



THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure de Paris

**Efficient and Scalable Methods for the Acquisition,
Retrieval, and Understanding of Open Tabular Data**

Soutenue par

Antoine Gauquier

Le 23 Septembre 2026

École doctorale n°386

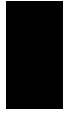
**Sciences Mathématiques de
Paris Centre**

Spécialité

Informatique

Composition du jury :

Bernd Amann Sorbonne Université	<i>Rapporteur</i>
Sihem Amer-Yahia CNRS et LIG	<i>Rapporteuse</i>
Fatiha Saïs Université Paris-Saclay	<i>Examinatrice</i>
Marc Spaniol Université de Caen Normandie	<i>Examineur</i>
Fabian Suchanek Télécom Paris–IPP	<i>Examineur</i>
Ioana Manolescu Inria et IPP	<i>Directrice de thèse</i>
Pierre Senellart École normale supérieure–PSL	<i>Directeur de thèse</i>



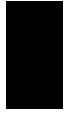
Abstract

Statistical data published by governments, statistical institutes, and international organizations are a public good: they inform public debate, support research, and provide the foundation for journalistic fact-checking. Yet their abundance is also what makes it hard to use. Statistics are scattered across hundreds of websites, published in heterogeneous formats, and organized for human readers rather than machines, so that the data, though available, are rarely usable in practice. Making it usable is not a single problem but a chain of them, from finding the data on the Web, to recovering its structure, to retrieving the few tables that answer a given question. This thesis addresses several links of that chain, with a particular focus on statistical tables, a multidimensional and comparatively neglected category of tabular data.

We make four contributions. First, for data acquisition, we formalize focused crawling within a website as a graph exploration problem, prove its NP-hardness, and propose SB-CLASSIFIER, a reinforcement-learning crawler that learns which hyperlinks lead to target files and retrieves a large fraction of them while visiting only a small part of the website. Second, for retrieval, we define Statistical Table Discovery and present STAR, a method that indexes the temporal, spatial, and remaining content of tables and queries separately, outperforming general-purpose table discovery systems while scaling to large corpora without a GPU. Third, for structure recovery, we propose an end-to-end spreadsheet understanding pipeline for cell-type classification and table extraction that runs on a single machine without a GPU, yet remains competitive with much heavier Transformer- and LLM-based systems. Fourth, for annotation, we build TABAGREE, an agreement-aware platform that produces reliable labels for these tasks at scale and treats inter-annotator disagreement as an interpretable signal. We also release two public benchmarks built with this platform, on statistical table discovery and on spreadsheet understanding.

Across these contributions, our methods are efficient on modest hardware, robust to the heterogeneity of real data, and scalable to the volumes reached by Web-scale statistical corpora, showing that open statistical data can be acquired, understood, and retrieved efficiently and at scale.

Open Statistical Data ★ Tabular Data ★ Focused Web Crawling ★ Table Discovery ★
Spreadsheet Understanding ★ Data Annotation ★ Scalable Data Management



Résumé

Les données statistiques publiées par les gouvernements, les instituts statistiques et les organisations internationales constituent un bien commun : elles éclairent le débat public, soutiennent la recherche et servent de fondement au fact-checking. Leur abondance rend toutefois leur exploitation difficile. Dispersées sur des centaines de sites Web, publiées dans des formats hétérogènes et organisées pour des lecteurs humains, elles sont souvent peu exploitables par les machines. Les rendre exploitables implique de résoudre une chaîne de problèmes : localiser les données sur le Web, en reconstruire la structure, puis retrouver les tableaux pertinents pour répondre à une question donnée. Cette thèse traite plusieurs maillons de cette chaîne en se concentrant sur les tableaux statistiques, une catégorie de données tabulaires à la fois multidimensionnelle et relativement peu étudiée.

Nous présentons quatre contributions. Premièrement, pour l’acquisition des données, nous formalisons le crawl ciblé d’un site Web comme un problème d’exploration de graphe, en démontrons sa NP-difficulté et proposons SB-CLASSIFIER, un crawler guidé par l’apprentissage par renforcement. Celui-ci apprend quels hyperliens mènent aux fichiers cibles et en récupère une grande partie tout en ne visitant qu’une faible portion du site. Deuxièmement, pour la recherche de tableaux, nous définissons le problème de découverte de tableaux statistiques et présentons STAR, une méthode qui indexe séparément le contenu temporel, spatial et linguistique des tableaux et des requêtes. STAR surpasse les systèmes génériques de découverte de tableaux tout en passant à l’échelle sur de grands corpus sans GPU. Troisièmement, pour la reconstruction de structure, nous proposons une chaîne de compréhension des feuilles de tableur couvrant la classification du type des cellules et l’extraction de tableaux. Elle s’exécute sur une seule machine sans GPU tout en demeurant compétitive avec des systèmes bien plus coûteux, fondés sur des transformeurs ou des LLM. Quatrièmement, pour l’annotation, nous développons TABAGREE, une plateforme permettant de produire à grande échelle des annotations fiables pour ces tâches et d’exploiter le désaccord entre annotateurs comme un signal interprétable. Nous publions deux jeux de données construits avec TABAGREE, dédiés à la découverte de tableaux statistiques et à la compréhension des feuilles de tableur.

Nos méthodes combinent efficacité, sobriété matérielle et passage à l’échelle. Robustes à l’hétérogénéité des données du monde réel, elles rendent possible l’acquisition, la compréhension et la recherche efficaces de données statistiques ouvertes à grande échelle.

Données statistiques ouvertes ★ Données tabulaires ★ Crawl ciblé du Web ★ Découverte de tableaux ★ Compréhension des tableurs ★ Annotation de données ★ Gestion de données à grande échelle



Remerciements

Mes premiers remerciements, et sans doute les plus chaleureux, vont à mes deux co-directeurs de thèse, Pierre Senellart et Ioana Manolescu. Je vous remercie tout d'abord pour la confiance que vous m'avez accordée en me prenant chacun à votre tour en stage, puis en m'accompagnant pendant ces trois années de thèse. Vous avez été, à bien des égards, mes premiers guides dans le monde de la recherche, et je vous dois une grande partie de ce que j'ai appris : la démarche de recherche scientifique, la conduite d'expériences, la rédaction d'articles, et bien d'autres choses encore. Au-delà de ces compétences, vous m'avez transmis une véritable conception du métier de chercheur : la rigueur scientifique, l'honnêteté intellectuelle, la passion de la transmission, l'importance de la communication et de la vulgarisation, la curiosité, le souci de la reproductibilité, et tant d'autres valeurs qui me paraissent aujourd'hui indissociables de ce métier.

Ces trois années sous votre encadrement m'ont rarement donné le sentiment d'être simplement « dirigé ». J'y ai davantage trouvé un accompagnement par des collègues et des mentors, toujours disponibles pour écouter et discuter de mes idées, m'aider à les mettre en valeur lorsqu'elles étaient pertinentes, mais aussi à les remettre en question et à explorer de meilleures pistes lorsque c'était nécessaire, toujours avec bienveillance et factualité. Vous m'avez appris à prendre du recul sur mon propre travail, à défendre mes idées lorsqu'elles le méritaient, mais aussi à en reconnaître les limites et à savoir les abandonner lorsqu'une meilleure voie se présentait. Pierre, Ioana, merci donc pour tout ce que vous m'avez appris, mais aussi pour la confiance et l'autonomie que vous m'avez accordées. Je suis certain que les compétences, méthodes et valeurs que vous m'avez transmises me seront précieuses dans la suite de ma carrière scientifique.

Je remercie ensuite les deux rapporteurs de ce manuscrit, Bernd Amann et Sihem Amer-Yahia, pour leurs retours constructifs et leurs remarques précises. Merci également à Fatiha Saïs d'avoir accepté d'être examinatrice de ma thèse, ainsi qu'à Marc Spaniol et Fabian Suchanek pour leur participation à mon comité de suivi individuel et leurs retours constructifs, et pour avoir accepté d'être examinateurs de cette thèse.

Je souhaite également remercier mes co-auteurs, avec qui j'ai eu la chance d'explorer des sujets de recherche aussi variés que passionnants. Ces collaborations m'ont permis de découvrir de nouvelles problématiques, mais aussi des méthodes et des façons de travailler différentes, contribuant ainsi à diversifier ma vision de la recherche et à enrichir mes compétences techniques au-delà du cœur de cette thèse. Je tiens à remercier plusieurs personnes. Nelly Barret, tout d'abord, avec qui j'ai fait mes premiers pas dans la recherche scientifique et découvert plus particulièrement les problématiques de la communauté

des bases de données, notamment autour de l'exploration de données hétérogènes et du passage à l'échelle. Shrey Mishra, avec qui j'ai enrichi mes compétences en machine learning et deep learning, et découvert de nouvelles problématiques autour de l'extraction d'informations à partir de documents, de la multimodalité et de l'efficacité. Enfin, mon travail avec Pratik Karmakar m'a ouvert à des problématiques plus théoriques en bases de données et m'a appris à raisonner différemment, en découvrant notamment les liens entre des problématiques très pratiques, comme la provenance des données, et des raisonnements plus abstraits issus des probabilités et de la théorie des jeux.

Merci également aux deux équipes Valda et CEDAR pour leur accueil pendant ces trois années. Je pense tout spécialement aux doctorants et post-doctorants avec qui j'ai eu le plaisir d'échanger plus ou moins régulièrement : Marijan, Shrey, Nelly, Shufan, Przemek, Madhulika, Félix, Lucas, Manos, Pratik, Angelo, Georgios, Mélanie, et bien d'autres encore. Merci également au service administratif du département d'informatique de l'ENS, et plus précisément Lise-Marie pour sa disponibilité et sa perpétuelle efficacité. Au-delà de la science, les discussions informelles, les séminaires et l'environnement de ces équipes ont aussi largement contribué à rendre ces trois années aussi riches qu'agréables.

J'en profite également pour remercier celles et ceux qui ont contribué, plus indirectement, à la réussite de cette thèse à travers mes activités en dehors de la recherche. Mes pensées vont tout d'abord aux étudiants à qui j'ai eu la chance d'enseigner pendant ces trois années. Je remercie les différentes promotions des deux CPES de PSL, ainsi que Paul Sevestre et Léo Boullot, que j'ai eu le plaisir d'encadrer en stage. Vous tous avez été mes premiers élèves et m'avez permis de prendre confiance dans ce nouveau rôle, tout en contribuant à forger ma manière d'enseigner. Mes remerciements vont également au personnel du Centre de Service de la Donnée et de l'IA de la Marine, qui m'a accueilli dans le cadre de mes périodes de réserve militaire. Je remercie vivement le CC Bimbault, le CC Bouiller et le CF Giraud pour leur confiance et pour m'avoir permis de contribuer à la valorisation de la gouvernance des données de la Marine nationale. Cette expérience m'a appris à transformer des travaux de recherche en systèmes réellement utilisables, à travailler avec des utilisateurs non experts et dans des environnements contraints, mais aussi à prendre des initiatives, à innover et à valoriser mon travail.

Mes derniers remerciements vont à ma famille, et plus spécialement à mes parents, mon frère, mes beaux-parents, mes grands-parents, ma marraine et mon parrain. Merci pour votre soutien indéfectible durant cette thèse, mais plus généralement tout au long de mon parcours scolaire. Merci d'avoir toujours cru en moi, de m'avoir poussé à donner le meilleur de moi-même, et de m'avoir transmis cette passion et cette curiosité qui m'animent. Je vous dois également beaucoup des valeurs qui m'ont permis d'arriver jusqu'ici : le goût de l'effort, la valeur du travail et l'importance de la persévérance pour atteindre ses objectifs. Pour terminer, je veux remercier tout particulièrement ma compagne, Claire. Merci de m'avoir accompagné pendant toutes ces années, d'avoir célébré avec moi les victoires qui ont ponctué mon parcours, mais aussi de m'avoir épaulé dans les moments les plus difficiles. Merci infiniment pour tout ce que tu m'as apporté et m'apportes encore un peu plus chaque jour, et pour avoir supporté, avec une patience remarquable, toutes ces heures d'explications et de réflexions souvent obscures à tes yeux.



Contents

Abstract	i
Résumé	iii
Remerciements	v
1 Introduction en Français	1
1.1 Introduction	1
1.2 Plan du manuscrit	6
1.3 Publications	7
1.4 Systèmes et prototypes de recherche	8
2 Introduction	11
2.1 Introduction	11
2.2 Manuscript Outline	15
2.3 Publications	17
2.4 Systems and Research Prototypes	19
3 Background	21
3.1 Tabular Data and Relational Databases	22
3.2 Entity-Oriented Tables on the Web	22
3.3 Spreadsheets	23
3.4 Querying Tables in Natural Language	24
3.5 Acquiring Tabular Data from the Web	25
3.6 Statistical Tables and their Societal Importance	26
4 Efficient Focused Web Crawling for Scalable Data Acquisition	29
4.1 Introduction	29
4.2 Graph-Based Modeling of Web Data Acquisition	31
4.2.1 Graph Crawling Problem and Intractability	31
4.2.2 Web Graph Modeling	33
4.3 Reinforcement Learning-Based Crawling	35
4.3.1 Environment: States and Actions	35
4.3.2 Grouping Links into Actions	36

4.3.3	Estimating Rewards with URL Classifier	39
4.3.4	Crawling Algorithm	41
4.4	Experimental Evaluation	43
4.4.1	Websites and Data Sources	43
4.4.2	Limitations of Search Engine-Based Dataset Retrieval	45
4.4.3	Baselines	46
4.4.4	Experimental Setup	48
4.4.5	Performance Comparison with Baselines	49
4.4.6	Hyperparameter Study	54
4.4.7	Effectiveness of Sleeping Bandit Learning	55
4.4.8	URL Classification Design and Quality	65
4.4.9	Multimedia Filtering	70
4.4.10	Early-Stopping Mechanism	71
4.5	Related Work	72
4.5.1	Web Crawlers	72
4.5.2	DOM Tag Path Representations	76
4.5.3	Multi-Armed Bandits Algorithms	76
4.6	Summary	76
5	Efficient and Scalable Search for Statistics	77
5.1	Introduction	77
5.2	Statistical Table Representation and Problem Setup	79
5.2.1	Statistical Table Data Model	79
5.2.2	Statistical Table Discovery Problem	80
5.3	Statistical Table Indexing	81
5.3.1	Linguistic Content Indexing	82
5.3.2	Separate and Exact Spatial and Temporal Indexing	84
5.4	Statistical Table Retrieval	86
5.4.1	Retrieval Procedure	86
5.4.2	Relaxation Mechanism	88
5.4.3	Best-Effort Table Question Answering	89
5.5	Architecture and Configuration	90
5.5.1	Model Modularity and Multilingual Capabilities	90
5.5.2	Indexing and Retrieval Configuration	91
5.6	Experimental Evaluation	93
5.6.1	Dataset of Statistical Tables	93
5.6.2	Systems	96
5.6.3	Generation of Evaluation Questions	98
5.6.4	Evaluation Metrics	100
5.6.5	Results and Analysis	101
5.6.6	Experiment Conclusions	107
5.7	Related Work	107
5.7.1	Table Discovery	108

5.7.2	Table Question Answering	108
5.7.3	End-to-end Question Answering	109
5.7.4	Statistical Table Discovery	109
5.8	Summary	110
6	Scalable Spreadsheet Table Understanding	111
6.1	Introduction	111
6.2	Related Work	113
6.2.1	Cell-Type Classification in Spreadsheets	113
6.2.2	Table Extraction in Spreadsheets	114
6.2.3	End-to-end Spreadsheet Understanding	115
6.2.4	Datasets for Spreadsheet Understanding	115
6.3	End-To-End Spreadsheet Table Understanding	116
6.3.1	Cell-Type Classification	116
6.3.2	Table Extraction	121
6.4	The StatSheets Dataset	126
6.4.1	Dataset Construction	126
6.4.2	Annotation Protocol and Dataset Statistics	127
6.5	Experimental Evaluation	128
6.5.1	Cell-Type Classification	129
6.5.2	Table Extraction	134
6.5.3	Experiment Conclusions	146
6.6	Summary	147
7	TabAgree: Spreadsheet Table Annotation	149
7.1	Introduction	149
7.2	Positioning with Respect to Existing Datasets and Tools	150
7.3	The TabAgree Annotation Platform	152
7.3.1	Table Relevance Annotation	153
7.3.2	Table Structure Annotation	154
7.3.3	Annotation Statistics and Inter-Annotator Agreement	155
7.4	Interactive Demonstration and System Walkthrough	156
7.5	Summary	157
8	Conclusions and Perspectives	159
8.1	Summary of Contributions	159
8.2	Perspectives	161
9	Other Scientific Contributions	165
9.1	Heterogeneous Graph Exploration and Entity Path Finding	165
9.2	Multimodal Information Extraction from Scientific Documents	166
9.3	Uncertainty Reasoning and Cooperative Game Theory	167

Introduction en Français

Cette thèse traite de l'acquisition, de la compréhension et de la recherche de données statistiques sur le Web. Ce chapitre introductif est organisé comme suit. La partie 1.1 présente les problèmes abordés par cette thèse et donne un aperçu de ses contributions. La partie 1.2 décrit le manuscrit chapitre par chapitre. La partie 1.3 liste les publications et présentations associées, et la partie 1.4 décrit les systèmes open-source et les prototypes de recherche publiés dans le cadre de ce travail.

1.1 Introduction

Les données statistiques sont omniprésentes dans la vie publique. Les gouvernements, les instituts nationaux de statistique et les organisations internationales publient de vastes quantités de chiffres sur la population, l'emploi, la santé, les prix, l'éducation ou l'environnement, et ces chiffres constituent un bien public. Les décideurs s'appuient sur eux pour éclairer leurs décisions et alimenter les débats, les chercheurs en sciences sociales les utilisent pour analyser les politiques et détecter des tendances, et les rédactions journalistiques s'en servent pour vérifier l'exactitude des affirmations publiques et comparer des pays ou des périodes. Lorsqu'une affirmation telle que « les décès en Belgique ont augmenté en 2021 » est formulée dans un débat public, il existe généralement, quelque part dans un jeu de données ouvert, une valeur permettant de la confirmer ou de la réfuter. La difficulté ne réside que rarement dans l'absence de données; elle tient plutôt au fait qu'il est difficile de trouver la bonne valeur dans une source fiable, dans le bon tableau, à l'endroit et au moment appropriés.

Cette difficulté mérite une attention particulière, car l'écart entre des données *disponibles* et des données *utilisables* a des conséquences. Les citoyens ont régulièrement une perception erronée de faits élémentaires sur les sociétés dans lesquelles ils vivent, parfois par plus d'un ordre de grandeur : dans une large enquête [144], des répondants aux États-Unis ont ainsi largement surestimé la taille de petits groupes démographiques, estimant par exemple la part de la population transgenre à 21 % contre une valeur réelle proche de 0,6 %, et la part des musulmans à 27 % contre environ 1 %. Les mêmes répondants ont sous-estimé la taille de grands groupes, tels que la proportion d'adultes chrétiens (58 % estimée contre 70 %) ou titulaires d'au moins un diplôme du secondaire (65 % contre 89 %). Cette déformation systématique illustre à quel point l'intuition collective peut s'éloigner de la réalité documentée, et un débat public plus sain dépend

en partie de la facilité d'accès et de vérification des chiffres sous-jacents. Les enjeux sont encore plus élevés face à la désinformation organisée : peu après l'invasion de l'Ukraine en 2022, le service français *VIGINUM* a documenté une vaste campagne persistante de manipulation de l'information (connue sous le nom de RRN, ou « Doppelganger »), qui enregistrait des noms de domaine imitant étroitement ceux de médias et d'institutions publiques français, dont un faux site du ministère de l'Europe et des Affaires Étrangères utilisé pour publier un communiqué de presse fabriqué, et les amplifiait via un réseau de faux comptes sur les réseaux sociaux [170]. À l'ère où les réseaux sociaux constituent un vecteur de propagation privilégié, la capacité à confronter les affirmations publiques à des statistiques fiables, et ainsi à contribuer à lutter contre la désinformation, n'est plus seulement une préoccupation académique. Pourtant, l'abondance même des données statistiques est ce qui rend leur exploitation difficile. Les statistiques sont dispersées sur des centaines, voire des milliers de sites Web, chacun ayant ses propres conventions d'organisation et de publication. Elles se présentent sous une grande variété de formats, allant des fichiers CSV et tableurs aux dialectes XML tels que SDMX [155] et aux tableaux intégrés dans des documents PDF. Surtout, elles sont publiées pour des lecteurs humains plutôt que pour des machines : un tableau statistique est conçu pour être *lu*, son sens étant porté par les titres, en-têtes, cellules fusionnées et l'organisation spatiale, plutôt que par un schéma explicite.

Rendre ces données exploitables en pratique n'est donc pas un problème unique, mais une *chaîne* de problèmes. Avant qu'une valeur puisse être récupérée, il faut d'abord trouver le tableau qui la contient quelque part sur le Web. Avant de pouvoir interroger ce tableau, il faut en reconstruire la structure (quelles cellules sont des en-têtes, lesquelles sont des données, où un tableau se termine et un autre commence) à partir d'une mise en page conçue pour l'œil humain, comme c'est typiquement le cas dans les tableurs. Une fois un corpus de tels tableaux disponible, répondre à une requête utilisateur revient à retrouver les quelques tableaux pertinents parmi potentiellement des centaines de milliers. Chacune de ces étapes constitue un obstacle en soi, et chacune a été étudiée, à divers degrés, par la communauté de la gestion de données. Ce qui a reçu beaucoup moins d'attention est le cas spécifique des tableaux *statistiques*, ainsi que l'intégration de ces étapes dans un pipeline unifié transformant des données brutes du Web en réponses à des requêtes utilisateurs. Cette thèse étudie plusieurs composantes de ce pipeline.

La distinction entre tableaux statistiques et tableaux relationnels ou orientés entités du Web (qui ont fait l'objet de nombreux travaux) est un point central de cette thèse et est examinée en détail dans le chapitre 3. En résumé, les tableaux statistiques présentent des caractéristiques très différentes de ceux pour lesquels la plupart des méthodes existantes ont été conçues. Contrairement à un tableau orienté entités, où chaque ligne décrit un objet et chaque colonne l'un de ses attributs, un tableau statistique est typiquement le résultat de l'agrégation d'une mesure selon plusieurs *dimensions*, dans l'esprit des cubes de données [100], ce qui le rend nettement plus difficile à modéliser et à traiter. Leurs cellules sont principalement numériques et portent peu d'information isolément : la valeur 11.3, par exemple, est difficile à interpréter sans savoir qu'elle correspond à un taux de mortalité, en Belgique, en 2021. La majeure partie de la sémantique est donc portée non

par les valeurs elles-mêmes mais par les *dimensions* du tableau. Parmi ces dimensions, le temps et la localisation géographique sont particulièrement importants, car ils sont présents dans la majorité des jeux de données statistiques et apparaissent fréquemment dans les requêtes des utilisateurs. Les tableaux statistiques sont également souvent volumineux. En tant qu'agrégats multidimensionnels, ils peuvent contenir des millions de lignes une fois sérialisés, à une échelle où les méthodes conçues pour des tableaux relationnels relativement petits deviennent difficilement applicables efficacement, soit parce que les coûts d'indexation augmentent fortement, soit parce que le traitement des requêtes devient prohibitif. Ces caractéristiques ont des implications importantes pour la conception de systèmes de recherche et de question-réponse, et influencent toutes les étapes du pipeline. Les méthodes qui n'en tiennent pas compte exploitent généralement mal la structure et la sémantique des données statistiques.

Parmi les formats dans lesquels les tableaux statistiques sont publiés, cette thèse se concentre sur les tableaux, dans la mesure où ils représentent l'essentiel des données statistiques librement disponibles. L'extraction de tableaux depuis d'autres sources, telles que les documents PDF, est un problème connexe mais distinct [161]. De même, détecter quelles affirmations méritent d'être vérifiées en priorité est un problème de traitement automatique du langage naturel, traité dans des travaux distincts et hors du périmètre de cette thèse.

Le travail de cette thèse est donc organisé autour de quatre composantes, chacune correspondant à un maillon de la chaîne, et le reste de cette partie les introduit dans l'ordre dans lequel les données les traversent.

Acquisition des données. Rien ne peut être structuré ou interrogé avant d'être trouvé, et la localisation de fichiers statistiques sur le Web est en elle-même non triviale. Certaines organisations exposent une API permettant d'accéder à leurs jeux de données, mais écrire du code spécifique pour chacune d'elles est extrêmement chronophage et devient rapidement obsolète dès qu'une API change. Par ailleurs, beaucoup d'autres ne proposent aucune interface de ce type, et les moteurs de recherche généraux n'indexent qu'une fraction limitée et opaque de ce qui est publié. L'alternative évidente, le crawling exhaustif d'un site, est prohibitive : les contraintes d'éthique du crawling imposent d'espacer les requêtes, de sorte que l'exploration d'un site d'un million de pages prend déjà plus d'une semaine rien que pour respecter ces délais. Nous étudions donc le *focused crawling*, où l'objectif est de récupérer autant de fichiers *cibles* (jeux de données) que possible sur un site donné en émettant le moins de requêtes possible. Le chapitre 4 formalise ce problème comme une exploration de graphe, montre qu'il est NP-difficile, et propose SB-CLASSIFIER, un crawler basé sur l'apprentissage par renforcement. Son idée centrale est que la structure d'un site Web est elle-même une source d'information : les liens présents sur des chemins de balises similaires au sein d'une page tendent à conduire à des contenus similaires, de sorte qu'un crawler peut apprendre, au cours de son exploration, quels chemins valent la peine d'être suivis pour atteindre les cibles. Évalué sur des sites totalisant plus de 22 millions de pages, SB-CLASSIFIER récupère, sur les plus grands sites, 90% des fichiers cibles en ne visitant que 20% des pages totales, sans connaissance préalable de la structure des sites et sans infrastructure parallèle de moteur de recherche.

Récupération de la structure. Trouver les fichiers cibles n’est utile que si leur contenu peut être interprété par une machine, et c’est ici que la nature orientée humain des données statistiques devient un obstacle. Les tableurs réels ne présentent que rarement un tableau isolé : une même feuille mélange souvent plusieurs zones de données avec des en-têtes hiérarchiques, des titres, des notes de bas de page et d’autres éléments auxiliaires, et les conventions de mise en forme varient selon les éditeurs, les langues et les logiciels. Reconstruire des tableaux interprétables automatiquement à partir de tels fichiers est nettement plus difficile que de parser, par exemple, un tableau HTML ou un export de base de données. Le chapitre 6 aborde ce problème avec un pipeline de bout en bout qui décompose la tâche en deux sous-problèmes : la classification du rôle sémantique de chaque cellule (en-tête, donnée, titre, etc.), puis le regroupement des cellules d’en-têtes et de données en tableaux cohérents. Un classifieur non linéaire prédit d’abord le rôle de chaque cellule à partir d’un ensemble riche de caractéristiques de contenu, de mise en forme et de position spatiale, puis un champ aléatoire conditionnel impose ensuite une cohérence spatiale entre les prédictions voisines. Un algorithme déterministe extrait enfin les limites des tableaux à partir de la grille obtenue, identifiant zéro, un ou plusieurs tableaux présents dans la feuille. Cette approche atteint ou dépasse la précision de méthodes bien plus lourdes basées sur des transformeurs ou des LLM, pour une fraction de leur coût et sans GPU. Afin de permettre une évaluation réaliste, nous publions également STATSHEETS, un benchmark de tableurs statistiques réels, volontairement variés en source, langue, sujet, taille et format, annotés pour les deux tâches.

Recherche des bonnes tables. Une fois les données trouvées et leur structure reconstruite, il s’agit alors de déterminer quel(s) tableau(x), parmi un corpus potentiellement très large, répond(ent) à un besoin exprimé en langage naturel. Il s’agit d’une variante du problème de *table discovery*, mais, comme discuté précédemment, les méthodes existantes supposent des tableaux relationnels orientés entités et ne sont pas adaptées aux données statistiques. Le chapitre 5 définit la découverte des tableaux statistiques (*Statistical Table Discovery*) comme un problème à part entière et propose STAR (Space- and Time-Aware Statistical Retrieval). L’idée clé est que les dimensions temporelles et spatiales sont presque toujours présentes dans les tableaux statistiques et apparaissent fréquemment dans les requêtes des utilisateurs. En conséquence, STAR indexe chaque table séparément selon ses dimensions temporelles, spatiales et sémantiques restantes, extrait les mêmes éléments depuis la requête, et les fait correspondre indépendamment, de sorte qu’une question portant sur un lieu et une période donnés est associée aux bons tableaux même lorsque la formulation diffère de celle utilisée dans le tableau lui-même. Les valeurs numériques, qui constituent la majorité des cellules, répondent aux questions sans pour autant en faire partie et n’ont donc pas besoin d’être indexées. Cela permet de réduire fortement la taille des index et d’accélérer le traitement des requêtes, même sur des corpus où les méthodes concurrentes saturent la mémoire ou deviennent très lentes. STAR permet ainsi de retrouver les tableaux statistiques pertinents avec une meilleure précision que les méthodes généralistes de *table discovery*. Contrairement aux approches de question-réponse fondées sur de grands modèles de langue, il reste rapide et précis à mesure que le corpus s’accroît, sans recourir à un LLM ni nécessiter de GPU. Il a par

ailleurs été intégré dans le système STATCHECK, utilisé en pratique par des journalistes de *Radio France* pour le journalisme de données statistiques et le fact-checking sur de larges collections de sources statistiques de confiance.

Annotation tout au long du processus. Deux de ces composantes, la récupération de structure et la recherche, reposent sur des données annotées, à la fois pour l’entraînement des modèles et pour leur évaluation. Produire de telles annotations est en soi une tâche non triviale. Les tableurs ont des mises en page complexes, et déterminer si un tableau répond à une question implique souvent des jugements subtils, conduisant à des désaccords fréquents entre annotateurs sur un même exemple. Des étiquettes fiables sont pourtant rares, car les outils d’annotation existants supportent rarement les tableurs entièrement formatés, la collecte à grande échelle ou la présence de plusieurs annotateurs simultanément. Plutôt que de considérer ces désaccords comme du bruit, le chapitre 7 les exploite comme un signal mesurable, et introduit TABAGREE, une plateforme d’annotation pour l’étiquetage à grande échelle avec plusieurs annotateurs. Elle supporte à la fois l’annotation au niveau de la cellule et au niveau structurel, réalisée dans un véritable éditeur de tableur afin de préserver les indices de mise en forme distinguant en-têtes, données et titres, même pour des fichiers de plusieurs millions de cellules, ainsi que l’évaluation de la pertinence question–tableau. Pour ces deux types d’annotations, elle calcule l’accord inter-annotateurs à l’aide de mesures simples et interprétables, fournissant une indication de la difficulté de la tâche et de la fiabilité des étiquettes, tout en facilitant la construction de nouveaux benchmarks.

Un pipeline de bout en bout. Pris séparément, chacune de ces composantes traite un problème d’intérêt pour la communauté de la gestion de données. Pris ensemble, ils forment un pipeline allant de l’acquisition de données statistiques sur le Web jusqu’à la récupération de tableaux répondant à une question d’un utilisateur. La Figure 2.1 donne un aperçu de ce pipeline et montre comment les chapitres s’articulent : un crawler ciblé collecte des tableurs depuis des organisations publiques fiables; la compréhension de tableurs reconstruit les tableaux et le rôle de chaque cellule; et la recherche de tableaux statistiques indexe le corpus résultant et récupère les tableaux les plus pertinents pour une requête en langage naturel. La plateforme d’annotation soutient les tâches sur lesquelles reposent l’entraînement et l’évaluation des composantes apprises. Tout au long du travail, les mêmes préoccupations ont guidé la conception : l’efficacité sur du matériel modeste, la robustesse à l’hétérogénéité des données réelles, le passage à l’échelle vers les volumes du Web, et, autant que faire se peut, l’interprétabilité des méthodes. Ce pipeline, construit à partir des contributions des chapitres suivants, constitue le cadre global de cette thèse.

Plusieurs de ces contributions dépassent délibérément le cadre des données statistiques. Le crawler n’est pas limité aux statistiques : il cible des fichiers de données de toute nature au sein d’un site, selon une notion de cible définie par l’utilisateur. De même, le pipeline de compréhension de tableurs classe les cellules et détecte les tableaux à partir de caractéristiques communes à toute feuille à contenu tabulaire, plutôt qu’à partir de caractéristiques spécifiques aux statistiques, de sorte que sa sortie peut servir d’entrée à toute tâche en aval nécessitant des représentations propres et structurées. Un autre résultat de ce travail est l’ensemble des données annotées produites au fil de la thèse et

publiées via TABAGREE, afin de permettre à la communauté d’entraîner et d’évaluer de futurs systèmes sur ces tâches.

1.2 Plan du manuscrit

Le reste du manuscrit est organisé comme suit.

Chapitre 3 : Contexte. Ce chapitre replace le travail de la thèse dans le contexte plus large de la littérature en gestion de données. En partant du modèle relationnel, il retrace le chemin parcouru par la communauté au cours des vingt dernières années : des tables HTML orientées entités aux tableurs, de la recherche par mots-clés à l’interrogation en langage naturel sur les tableaux, et des corpus statiques au crawling Web à grande échelle. Il conclut en introduisant les tableaux statistiques comme une catégorie distincte de données structurées, dont l’acquisition, la compréhension et la recherche posent des défis qui ne sont que partiellement traités par les travaux existants.

Chapitre 4 : Crawling Web ciblé et efficace pour l’acquisition de données à grande échelle. Ce chapitre formalise le *focused crawling* pour les jeux de données statistiques comme un problème d’exploration de graphe, établit sa NP-difficulté, et présente le crawler SB-CLASSIFIER. Il inclut une étude expérimentale extensive sur un ensemble d’organisations publiques de confiance dont les sites totalisent plus de 22 millions de pages distinctes, en comparant SB-CLASSIFIER à des baselines de crawling ciblé.

Chapitre 5 : Recherche efficace de statistiques passant à l’échelle. Ce chapitre définit le Statistical Table Discovery et présente STAR (Space- and Time-Aware Statistical Retrieval). STAR est évalué par rapport à des systèmes généralistes de table discovery sur un jeu de données publié dans ce travail, comprenant 7 605 tableaux *Eurostat*, deux ensembles de 100 questions en langage naturel, et près de 5 000 paires (question, tableau) annotées manuellement, constituant le premier benchmark de ce type construit sur des données ouvertes réelles et variées.

Chapitre 6 : Compréhension efficace des tableurs passant à l’échelle. Ce chapitre présente un pipeline de bout en bout pour la classification des types de cellules et l’extraction de tableaux dans les tableurs, combinant un classifieur non linéaire, un champ aléatoire conditionnel, et une étape d’extraction déterministe, que nous comparons à des méthodes basées sur transformeurs et LLM. Il introduit également STATSHEETS, un benchmark publiquement disponible de 737 feuilles annotées issues de 14 organisations publiques, couvrant 818 tableaux et plus de sept millions de cellules annotées.

Chapitre 7 : TabAgree : Annotation de tableaux de tableurs. Ce chapitre présente TABAGREE, une plateforme d’annotation multi-annotateurs pour les tableaux de tableurs. Elle supporte deux workflows : l’annotation au niveau des cellules et de la structure des tableaux, réalisés dans un véritable éditeur de tableur afin de préserver la mise en forme, et l’évaluation de la pertinence question–tableau. Dans les deux cas, elle rapporte l’accord inter-annotateurs à l’aide de métriques simples et interprétables,

fournissant un signal direct de difficulté des tâches et réduisant l’effort nécessaire pour construire de nouveaux benchmarks pour ces tâches.

Chapitre 8 : Conclusions et perspectives. Ce chapitre résume les quatre contributions principales et revient sur les propriétés transversales qui les relient. Il ouvre ensuite un ensemble de perspectives, à la fois spécifiques à chaque composant et transversales, notamment la combinaison des méthodes proposées en un pipeline de bout en bout fonctionnant depuis l’acquisition de données statistiques sur le Web jusqu’à la réponse aux questions des utilisateurs sur ces données.

Chapitre 9 : Autres contributions scientifiques. Le chapitre final présente des contributions de recherche réalisées durant le doctorat dans le cadre de projets collaboratifs en dehors du périmètre principal de la thèse. Trois directions sont couvertes : l’exploration de graphes hétérogènes et la recherche de chemins d’entités dans le système CONNECTIONLENS, l’extraction d’information multimodale à partir de documents scientifiques, et la quantification de l’incertitude via la théorie des jeux coopératifs. Ces travaux ont donné lieu à des publications dans des conférences et journaux internationaux à comité de relecture par les pairs et reflètent l’activité scientifique plus largement menée durant le doctorat.

1.3 Publications

Les travaux présentés dans cette thèse ont été acceptés pour publication et présentation dans des conférences et journaux nationaux et internationaux à comité de relecture par les pairs. Les publications sont organisées par chapitre. Les contributions BDA correspondent à des présentations informelles lors de la conférence française annuelle de bases de données, soit présentant des travaux déjà publiés dans des conférences ou journaux internationaux, soit présentant des recherches en cours ou soumises pour évaluation. Ces présentations n’ont pas donné lieu à des publications supplémentaires et ne se recoupent donc pas avec leurs versions internationales correspondantes.

J’ai également participé à plusieurs projets de recherche collaboratifs en dehors du périmètre principal de la thèse, ayant conduit à des publications dans des conférences et journaux internationaux : les travaux associés sont présentés dans le chapitre 9.

Publications pour le chapitre 4

1. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Reinforcement Learning–Based Focused Web Crawling for Robust and Scalable Data Acquisition, 2026. [**En cours de relecture**] [80]
2. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawling for Scalable Web Data Acquisition. In *Proceedings of the 29th International Conference on Extending Database Technology, EDBT 2026*, pages 372–385. [79]

3. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawler for Scalable Web Data Acquisition. Presented in the *40th Annual French Conference on Databases, BDA 2024*. [69]
4. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient and Focused Web Crawling for Statistical Data Sources Retrieval. Presented in the *2nd Workshop “Journées Infox sur Seine”*, 2024. [70]

Publications pour le chapitre 5

5. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. STAR: Efficient, Scalable, and Modular Retrieval of Statistical Tables. *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, 2026. [77]
6. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*, pages 1113–1126. [76]
7. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. Presented in the *41st Annual French Conference on Databases, BDA 2025*. [71]

Publications pour le chapitre 6

8. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management, CIKM 2026*. [81]
9. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. Presented in the *42nd Annual French Conference on Databases, BDA 2026*. [78]

Publications pour le chapitre 7

10. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. TABAGREE: An Agreement-Aware Platform for Reliable Spreadsheet Table Annotation (démonstration), 2026. [En cours de relecture]. [82]

1.4 Systèmes et prototypes de recherche

Les travaux présentés dans cette thèse ont donné lieu à plusieurs systèmes open-source et prototypes de recherche, chacun accompagné de code et/ou de données rendus publics afin de permettre leur reproductibilité ainsi que des futurs travaux.

Chapitre 4 : Crawler ciblé et adaptation de TRES. Le kit de reproductibilité des expériences du chapitre 4 [74] est disponible sur [GitHub](#). Le dépôt fournit trois implémentations de crawlers : des crawlers locaux exécutés sur des répliques de sites stockées dans des bases SQLite (incluant SB-CLASSIFIER et toutes les baselines), un outil online-to-local permettant de répliquer un site Web dans une base SQLite locale, et des crawlers semi-online basés sur le framework *Scrapy* combinant accès local et requêtes HTTP en ligne. Des scripts permettant de lancer les expériences et de reproduire toutes les figures sont inclus. Par ailleurs, notre adaptation du topical crawler TRES [121] à la tâche d’acquisition de datasets est disponible [75] : elle étend le système original avec un modèle de classification sauvegardé et chargé au format `.keras` par souci de compatibilité avec les bibliothèques modernes d’apprentissage profond.

Chapitre 5 : Système STAR–StatCheck et kit de reproductibilité. Le système STAR–STATCHECK est disponible en open-source sur [GitLab](#) [73]. Il s’agit d’une implémentation Python du pipeline complet de Statistical Table Discovery, avec une documentation MkDocs couvrant l’installation, l’indexation et les requêtes. Le kit de reproductibilité [72] est disponible sur [GitHub](#) et contient : les deux ensembles de questions d’évaluation, les top-10 des résultats de tous les systèmes comparés (BM25, STAR avec PEARL et SBERT, SOLO, PNEUMA, BIRDIE), des paires (question, tableau) annotées manuellement pour le calcul des scores de pertinence, ainsi que des scripts Python pour calculer les métriques. Les deux jeux de données (2 000 tables *Eurostat* les plus légères et le jeu complet) sont archivés sur [Zenodo](#)¹.

Chapitre 6 : Jeu de données StatSheets, pipelines de code et implémentation de SpreadsheetLLM. Le kit de reproductibilité du chapitre 6 est disponible publiquement sur [Zenodo](#) [83]. Il inclut le jeu de données STATSHEETS, avec 737 feuilles annotées pour les tâches de classification de cellules et d’extraction de tableaux. Il contient également le code complet pour tous les modèles de classification de cellules (RF, LIGHTGBM, CRF-LINEAR, CRF-RF, CRF-LIGHTGBM), ainsi que le code de l’algorithme déterministe d’extraction de tableaux appliqué aux grilles de cellules prédites (TE(CELL-TYPE CLASSIFIER)). Enfin, le dépôt inclut notre implémentation du framework SPREADSHEETLLM [59] (pipelines de fine-tuning et d’inférence pour un modèle MISTRAL-7B avec adaptateurs LoRA), qui constitue à notre connaissance la seule implémentation publique de cette méthode pour l’extraction de tableaux.

Chapitre 7 : Plateforme d’annotation TabAgree. Le code source complet de la plateforme TABAGREE est disponible sur [GitLab](#) [84], avec les instructions de déploiement et un lien vers une vidéo de démonstration. Le dépôt inclut des données d’exemple prêtes à l’emploi pour les deux tâches d’annotation (pertinence question–tableau et annotation structurelle cellule/tableau), permettant un déploiement immédiat sans préparation préalable des données.

¹<https://doi.org/10.5281/zenodo.15681383>

Introduction

This thesis addresses the acquisition, understanding, and retrieval of statistical data on the Web. This introductory chapter is organized as follows. Section 2.1 presents the problems this thesis addresses and gives an overview of its contributions. Section 2.2 outlines the manuscript chapter by chapter. Section 2.3 lists the associated publications and presentations, and Section 2.4 describes the open-source systems and research prototypes released with this work.

2.1 Introduction

Statistical data is everywhere in public life. Governments, national statistical institutes, and international organizations publish vast quantities of figures on population, employment, health, prices, education, or the environment, and these figures are a public good. Officials rely on them to inform decisions and support debates, social scientists use them to analyze policies and detect trends, and newsrooms use them to check the accuracy of public claims and to compare countries or periods. When a statement such as “deaths in Belgium rose in 2021” is made in a public debate, somewhere in an open dataset there is usually a figure that confirms or refutes it. The difficulty is rarely that the data does not exist; it is that finding the right figure in a trustworthy source, in the right table, at the right location and time, is hard.

This difficulty deserves careful attention, because the gap between data being *available* and data being *usable* has consequences. Citizens routinely misjudge basic facts about the societies they live in, sometimes by more than an order of magnitude: in a large survey [144], respondents in the United States vastly overestimated the size of small demographic groups, placing for instance the transgender share of the population at 21% against a true figure near 0.6%, and the Muslim share at 27% against about 1%. The same respondents underestimated the size of large groups, such as the share of adults who are Christian (58% estimated against 70%) or who hold at least a high-school degree (65% against 89%). Such systematic misperception illustrates how far public intuition can drift from documented reality, and a healthier public debate depends in part on making the underlying numbers easy to reach and to check. The stakes are higher still in the face of organized disinformation: shortly after the 2022 invasion of Ukraine, the French state service *VIGINUM* documented a large, persistent information-manipulation campaign (known as RRN, or “Doppelganger”), that registered domain names closely

imitating those of French media and public institutions, including a counterfeit of the Ministry for Europe and Foreign Affairs used to publish a fabricated press release, and amplified them through a network of inauthentic social-media accounts [170]. In an era where social networks are a privileged vector of propagation, the ability to confront public claims with reliable statistics, and thereby to help fight the spread of misinformation, is no longer only an academic concern. Yet the very abundance of statistical data is also what makes it hard to exploit. Statistics are scattered across hundreds, if not thousands, of websites, each with its own organization and publication habits. They come in a wide variety of formats, from CSV and spreadsheet files to XML dialects such as SDMX [155] and tables embedded in PDF documents. And, crucially, they are published for human readers rather than for machines: a statistical table is laid out to be *looked at*, with its meaning carried by titles, headers, merged cells, and spatial arrangement rather than declared in an explicit schema.

Making this data usable in practice is therefore not a single problem but a *chain* of them. Before a figure can be retrieved, the table that contains it has to be found somewhere on the Web. Before the table can be queried, its structure, which cells are headers, which are data, where one table ends and another begins, has to be recovered from a layout meant for the human eye, as is typically the case in the spreadsheets through which tabular data is so often published. Once a corpus of such tables is available, answering a user’s question means retrieving the few tables, out of possibly hundreds of thousands, that are relevant to it. Each of these steps is an obstacle on its own, and each has been studied, to varying degrees, by the data management community. What has received much less attention is the specific case of *statistical* tables, and the problem of integrating these steps into a unified pipeline that transforms raw data from the Web into answers to user queries. This thesis investigates several components of this pipeline.

The distinction between statistical tables and the relational or entity-oriented Web tables that have been the focus of much prior work is a central consideration throughout this thesis and is examined in detail in the next chapter. In summary, statistical tables exhibit characteristics that differ substantially from those of the tables for which most existing methods have been developed. Unlike an entity-oriented table, where each row describes one object and each column one of its attributes, a statistical table is typically the result of aggregating a measure along several *dimensions*, in the spirit of data cubes [100], which makes it substantially harder to model and to handle. Their cells are predominantly numerical and convey little information in isolation: the value 11.3, for example, is difficult to interpret without knowing that it corresponds to a death rate, in Belgium, in 2021. Most of the semantic information is therefore carried not by the values themselves but by the table’s dimensions. Among these, time and geographic location are particularly important, as they are present in most statistical datasets and frequently appear in user queries. Statistical tables also tend to be large. Such aggregates may contain millions of rows once serialized, a scale at which methods designed for relatively small relational tables become difficult to apply efficiently, whether because indexing costs increase substantially or because query processing becomes prohibitively slow. These characteristics have important implications for the design of retrieval and

question-answering systems, and they influence every stage of the processing pipeline. Methods that do not account for them generally fail to exploit the structure and semantics of statistical data effectively.

Among the formats in which statistical tables are published, this thesis focuses on spreadsheets, since they account for most openly available statistical data; extracting tables from other sources, such as PDF documents, is a related but separate problem [161]. Similarly, detecting which claims are worth fact-checking in the first place is a natural-language processing problem, addressed in separate work and outside the scope of this thesis.

The work of this thesis is therefore organized around four components, each addressing one link of the chain, and the rest of this section introduces them in the order in which data flows through them.

Acquiring the data. Nothing can be structured or retrieved before it is found, and finding statistical files on the Web is itself non-trivial. Some organizations expose an API to retrieve their datasets, but writing specific code for each one is heavily time-consuming and breaks whenever an API changes; many others offer no such interface at all, and general search engines index only a small, opaque fraction of what is published. The obvious alternative, crawling a website exhaustively, is prohibitively expensive: crawling ethics require waiting between requests, so that acquiring a website of a single million pages can take more than a week of waiting time alone. We therefore study *focused* crawling, where the goal is to retrieve as many *target* files (datasets) as possible within a website while issuing as few requests as possible. Chapter 4 formalizes this as a graph exploration problem, shows that solving it optimally is NP-hard, and proposes SB-CLASSIFIER, a reinforcement-learning-based focused crawler. Its central idea is that the structure of a website is itself a guide: hyperlinks found on similar *tag paths* within a page tend to lead to similar content, so a crawler can learn, as it explores, which paths are worth following to acquire the targets. Evaluated on websites totaling more than 22 million pages, SB-CLASSIFIER retrieves, on the larger websites, 90% of the target files while visiting only 20% of the pages, without any prior knowledge of a website’s layout and without the parallel infrastructure of a search engine.

Recovering the structure. Finding the target files is only useful if their contents can be read by a machine, and this is where the human-oriented nature of statistical data becomes an obstacle. Real-world spreadsheets rarely present a table in isolation; one sheet often mixes several data regions with hierarchical headers, titles, footnotes, and other auxiliary elements, and the conventions for laying them out vary across publishers, languages, and software. Recovering machine-interpretable tables from such files is substantially harder than parsing, for example, an HTML table or a database export. Chapter 6 addresses this with an end-to-end pipeline that splits the problem into two tasks: classifying the semantic role of each cell (header, data, title...), and grouping the header and data cells into the tables they form. A non-linear classifier first predicts a role for each cell from a rich combination of content, formatting, and spatial features; a conditional random field then enforces spatial coherence between neighboring predictions. A deterministic algorithm finally extracts the table boundaries from the resulting grid,

identifying zero, one or several tables that lie in the spreadsheet’s sheet. This approach matches or exceeds the accuracy of much heavier Transformer- and LLM-based methods at a fraction of their cost, and without requiring a GPU. To enable evaluation under realistic conditions, we also release *STATSHEETS*, a benchmark of real statistical spreadsheets, deliberately varied in source, language, topic, size, and format, annotated for both tasks.

Retrieving the right table(s). With data found and its structure recovered, the question becomes which table, out of a potentially very large corpus, answers a given information need expressed in natural language. This is a variant of the *table discovery* problem, but, as discussed above, the standard methods assume entity-oriented relational tables and do not fit statistical data well. Chapter 5 defines Statistical Table Discovery as a problem on its own, and addresses it with STAR (Space- and Time-Aware Statistical Retrieval). The key observation is that time and space dimensions are present in most statistical tables and are frequently specified in user queries. Accordingly, STAR indexes each table along its temporal, spatial, and remaining semantic elements separately, extracts the same elements from a query, and matches them independently, so that a question about a given location and time is matched against the right tables even when the wording differs between the question and the tables. Numerical fact values, which constitute most of the cells, answer questions without being part of them and therefore do not require indexing. This keeps indexes small and allows fast query processing, even on corpora where competing systems exhaust memory or run much more slowly. STAR therefore retrieves relevant statistical tables more accurately than general-purpose table discovery methods. Unlike question-answering approaches built around large language models, it also stays fast and accurate as the corpus grows while using no LLM and requiring no GPU. STAR has been integrated into the *STATCHECK* system, used by journalists at *Radio France* for fact-checking claims based on INSEE and Eurostat statistical datasets.

Annotating, throughout. Two of these components, structure recovery and retrieval, rely on labeled data, both for training models and for evaluating their performance. Producing such labels is itself a non-trivial task. Spreadsheets have complex layouts, and judging whether a table answers a question often involves subtle, and sometimes expert, decisions, leading to frequent disagreement among annotators working on the same item. Reliable labels are nonetheless scarce, since existing annotation tools rarely support fully formatted spreadsheets, large-scale collection, or multiple annotators at once. Rather than treat this disagreement as noise, Chapter 7 uses it as a signal worth measuring and introduces *TABAGREE*, an annotation platform for large-scale, multi-annotator labeling. It supports both cell-level and table-structure annotation, performed within a real spreadsheet editor so that the formatting cues distinguishing headers, data, and titles are preserved, even for files with millions of cells, as well as question–table relevance assessment. For both types of annotations, it computes inter-annotator agreement through simple and interpretable measures, providing an indication of task difficulty and label reliability, and facilitating the construction of new benchmarks for these tasks.

A pipeline, end to end. Taken separately, each component addresses a problem of interest to the data management community. Taken together, they form a pipeline that goes from the acquisition of statistical data on the Web all the way to the retrieval of tables that answer a user’s question. Figure 2.1 provides an overview of this pipeline and shows how the chapters fit together: a focused crawler collects spreadsheets from a set of trusted public organizations; spreadsheet understanding recovers the tables and the role of each cell; and statistical table discovery indexes the resulting corpus and retrieves the most relevant tables for a natural-language query. The annotation platform supports the tasks on which the learned components are trained and evaluated. Throughout, the same concerns guided the design: efficiency on modest hardware, robustness to the heterogeneity of real data, scalability to the volumes that Web-scale statistical corpora actually reach, and, as far as possible, interpretable methods. This pipeline, assembled from the contributions of the following chapters, provides the overall framework of this thesis.

Several of these contributions deliberately reach beyond statistical data. The crawler is not tied to statistics: it targets data files of any kind within a website, under a notion of target that the user defines. Likewise, the spreadsheet-understanding pipeline classifies cells and detects tables from features common to any sheet with tabular content, rather than from features specific to statistics, so its output can serve as input to any downstream task that needs clean, structured representations. A further outcome of this work is the annotated data produced along the way and released through TABAGREE, meant to let the community train and evaluate future systems on these tasks.

2.2 Manuscript Outline

The rest of this manuscript is organized as follows.

Chapter 3: Background. This chapter situates the work of the thesis within the broader context of the data management literature. Starting from the relational model, it traces the path the community has taken over the past two decades: from entity-oriented HTML tables to spreadsheets, from keyword search to natural language querying over tables, and from static corpora to Web-scale focused crawling. It ends by introducing statistical tables as a distinctive class of structured data whose acquisition, understanding, and retrieval pose challenges that remain only partially addressed by existing work.

Chapter 4: Efficient Focused Web Crawling for Scalable Data Acquisition. This chapter formalizes focused crawling for statistical datasets as a graph exploration problem, establishes its NP-hardness, and presents the SB-CLASSIFIER crawler. It includes an extensive experimental study on a set of trusted public organizations whose websites total more than 22 million distinct pages, comparing SB-CLASSIFIER against focused baselines.

Chapter 5: Efficient and Scalable Search for Statistics. This chapter defines Statistical Table Discovery and presents STAR (Space- and Time-Aware Statistical Retrieval). It is evaluated against general-purpose table discovery systems on a dataset,

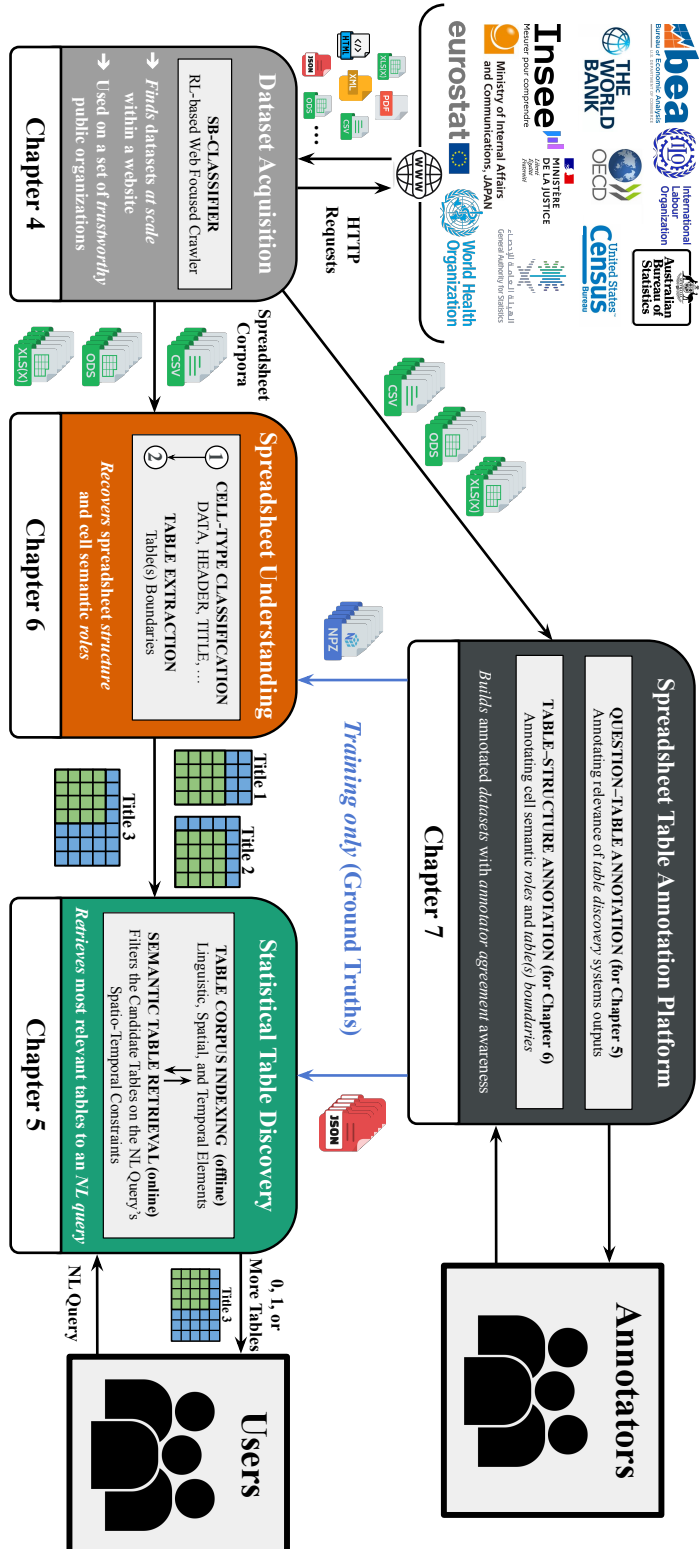


Figure 2.1: Overview of the thesis pipeline for acquiring, understanding, and retrieving open statistical data. A focused crawler (Chapter 4) acquires spreadsheets from trusted public organizations; spreadsheet understanding (Chapter 6) recovers cell roles and table boundaries; and statistical table discovery (Chapter 5) indexes the resulting tables and retrieves those relevant to a user’s natural-language query. The annotation platform TABAGREE (Chapter 7) builds the annotated datasets on which these tasks are trained and evaluated.

released with this work, of 7605 *Eurostat* tables, two sets of 100 natural-language questions, and nearly 5000 manually annotated (question, table) pairs, the first such benchmark built on real, varied Open Data.

Chapter 6: Scalable Spreadsheet Table Understanding. This chapter presents an end-to-end pipeline for cell-type classification and table extraction in spreadsheets, combining a non-linear classifier, a conditional random field, and a deterministic extraction step, which we compare against Transformer- and LLM-based methods. It also introduces STATSHEETS, a publicly available benchmark of 737 annotated sheets drawn from 14 public organizations, covering 818 tables and more than seven million labeled cells.

Chapter 7: TabAgree: Spreadsheet Table Annotation. This chapter presents TABAGREE, a multi-annotator annotation platform for spreadsheet tables. It supports two workflows: cell- and table-structure annotation, carried out inside a real spreadsheet editor so that formatting is preserved, and question-table relevance assessment. In both, it reports inter-annotator agreement through simple, interpretable metrics, providing a direct signal of task difficulty and lowering the effort of building new benchmarks for these tasks.

Chapter 8: Conclusions and Perspectives. This chapter summarizes the four core contributions and revisits the transversal properties that connect them. It then opens a set of future directions, both specific to each component and transversal, including the prospect of combining the proposed methods into a single end-to-end pipeline that runs from acquiring statistical data on the Web all the way to answering the questions users ask over it.

Chapter 9: Other Scientific Contributions. The final chapter presents research contributions made during the PhD in collaborative projects outside of the main scope of the thesis. Three directions are covered: heterogeneous graph exploration and entity path finding in the CONNECTIONLENS system, multimodal information extraction from scientific documents, and uncertainty quantification through cooperative game theory. These works led to publications in international peer-reviewed conferences and journals, and reflect the broader scientific activity carried out during the doctorate.

2.3 Publications

The work presented in this thesis has been accepted for publication and presentation in national and international peer-reviewed conferences and journals. The publications are organized by chapter. The BDA contributions correspond to informal presentations at the annual French database conference, either reporting work already published in international venues or presenting ongoing research and work under review. These presentations did not result in additional publications and therefore do not overlap with their corresponding international versions.

I also took part in several collaborative research projects outside the scope of the main thesis topic that led to publications in international peer-reviewed venues; the associated works are presented in Chapter 9.

Publications for Chapter 4

1. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Reinforcement Learning–Based Focused Web Crawling for Robust and Scalable Data Acquisition, 2026. [Under Review] [80]
2. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawling for Scalable Web Data Acquisition. In *Proceedings of the 29th International Conference on Extending Database Technology, EDBT 2026*, pages 372–385. [79]
3. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawler for Scalable Web Data Acquisition. Presented in the *40th Annual French Conference on Databases, BDA 2024*. [69]
4. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient and Focused Web Crawling for Statistical Data Sources Retrieval. Presented in the *2nd Workshop “Journées Infox sur Seine”*, 2024. [70]

Publications for Chapter 5

5. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. STAR: Efficient, Scalable, and Modular Retrieval of Statistical Tables. *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, 2026. [77]
6. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*, pages 1113–1126. [76]
7. Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. Presented in the *41st Annual French Conference on Databases, BDA 2025*. [71]

Publications for Chapter 6

8. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management, CIKM 2026*. [81]
9. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. Presented in the *42nd Annual French Conference on Databases, BDA 2026*. [78]

Publications for Chapter 7

10. Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. TABAGREE: An Agreement-Aware Platform for Reliable Spreadsheet Table Annotation (demonstration), 2026. [Under Review]. [82]

2.4 Systems and Research Prototypes

The work presented in this thesis has produced several open-source systems and research prototypes, each accompanied by code and/or data made publicly available to support reproducibility and further research.

Chapter 4: Focused crawler and TRES adaptation. The experiment reproducibility kit for Chapter 4 [74] is available on [GitHub](#). The repository provides three crawler implementations: *local crawlers* that run on website replicas stored as SQLite databases (including SB-CLASSIFIER and all baselines), an *online-to-local* tool that replicates a live website into a local SQLite database, and *semi-online crawlers* built on the [Scrapy](#) framework that combine local database lookups with live HTTP requests. Scripts to run experiments and reproduce all figures are included. Separately, our adaptation of the TRES [121] topical focused crawler to the target-retrieval task is available [75]; it extends the original TRES system with a classifier model saved and loaded in `.keras` format for compatibility with recent deep learning libraries.

Chapter 5: STAR–StatCheck system and experiment reproducibility kit. The STAR–STATCHECK system is available as open-source code on [GitLab](#) [73]. It is a Python implementation of the full Statistical Table Discovery pipeline, with MkDocs-based documentation covering installation, indexing, and querying. The experiment reproducibility kit [72] is available on [GitHub](#) and contains: the two evaluation question sets, the top-10 retrieval outputs of all compared systems (BM25, STAR with PEARL and SBERT, SOLO, PNEUMA, BIRDIE), manually annotated (question, table) pairs for computing the relevance scores, and Python scripts to compute both metrics. The two datasets (2 000 lightest [Eurostat](#) tables and the full dataset) are archived on [Zenodo](#)¹.

Chapter 6: StatSheets dataset, code pipelines, and SpreadsheetLLM implementation. The experiment reproducibility kit for Chapter 6 is publicly available on [Zenodo](#) [83]. It includes the STATSHEETS dataset, with 737 annotated sheets for both cell-type classification and table extraction tasks. It also contains the full code for all cell-type classification models (RF, LIGHTGBM, CRF-LINEAR, CRF-RF, CRF-LIGHTGBM), as well as the code for the deterministic table extraction algorithm operating on predicted cell-type grids (TE(CELL-TYPE CLASSIFIER)). Finally, the repository includes our implementation of the SPREADSHEETLLM [59] framework (fine-tuning and inference pipelines for a MISTRAL-7B model using LoRA adapters), which to our knowledge is the only publicly available implementation of this method for table extraction.

¹<https://doi.org/10.5281/zenodo.15681383>

Chapter 7: TabAgree annotation platform. The full source code for the TABAGREE platform is available on [GitLab](#) [84], along with deployment instructions and a demonstration video link. The repository includes ready-to-use example data for both annotation tasks (question–table relevance and cell/table-structure labeling), allowing immediate deployment without requiring prior data preparation.

Background

Tabular data is one of the oldest and most common ways of organizing information. Long before relational database systems, census reports, scientific measurements, and financial reports were laid out as rows and columns. The reason is simple: a grid lets a human reader find a value, compare it to its neighbors, and estimate at a glance how a quantity varies along one or more dimensions. This apparent simplicity, however, hides a difficulty that comes from the *dual* nature of tables: a table is at once a machine-processable data structure and a human-readable display format, and its organization often reflects visual and editorial choices rather than logical ones.

This chapter provides the background needed to situate the contributions, and is organized as follows. Its structure follows the path that the data management community has itself taken over the past two decades. We start from the relational model and the preconditions it requires (Section 3.1), and then move to the tables found on the open Web. We first look at entity-oriented tables extracted from HTML pages, which were a natural first step for the community, since they behave much like the relational tables its tools were designed for (Section 3.2). We then turn to spreadsheets, which are much harder to process, and which relational systems cannot use directly without first recovering their structure (Section 3.3). Next, we look at how such tables, once available, can be searched for, or queried, via natural language. The data management and natural language processing communities have approached this from different angles, and a recurring tension between what a method can express and how well it scales runs through this work (Section 3.4). Since all of this requires having large collections of tables in the first place, we review how such collections are built, either as static corpora or through focused crawling (Section 3.5). Together, these lines of work point to three related needs: *acquiring*, *understanding*, and *querying* tabular data. They also leave one type of table aside, despite its practical value: *statistical tables*, whose aggregated structure and societal importance make them the main object of this thesis (Section 3.6).

Throughout this chapter, our goal is to give background rather than a complete survey; the positioning of each contribution with respect to prior work is delegated to the related-work section of each chapter.

3.1 Tabular Data and Relational Databases

The relational model, introduced by Codd [47], represents data as relations, that is, sets of tuples over named, typed attributes, and manipulates them through a small algebra that is independent of any physical storage layout. A few years later, this vision was made practical for end users by the SEQUEL language of Chamberlin and Boyce [33], later standardized as SQL, which lets one express complex retrieval, joins, and aggregation declaratively, leaving the system free to choose how to run the query. The combination proved very successful: it supports a large industry and remains, decades later, the reference against which other data management paradigms are compared. On top of it, a large body of work has addressed multidimensional analysis through data warehouses and OLAP cubes [100], temporal data [101], and standardized exchange formats for structured statistics such as SDMX [155].

This power, however, relies on a strong precondition: the data must already reside in a database, under a schema that someone has designed, and queries must be written in a formal language. In other words, the relational model assumes that the work of modeling, ingesting, and curating the data has already been done. This work is often costly, and has traditionally fallen to database specialists, for the aforementioned reasons. Most of the world’s tabular data does not satisfy this precondition. With the advent and explosive growth of the Web, a large volume of structured and semi-structured data became openly available, yet almost none of it lived inside a database management system. To make such “data in the wild” usable, the data management community worked to make it queryable with the tools it had developed over the previous decades.

3.2 Entity-Oriented Tables on the Web

A natural first target was the large population of HTML tables found in Web pages. Many of these are *entity-oriented* (or entity-centric) tables, in which each row describes one entity, such as an album, a city, or a company, and the columns give its attributes. Each such table behaves like a small relational table that simply lacks explicit metadata. For a community already used to the relational model, these were the most approachable tables on the Web: their row-per-entity structure matches the relational view of data, and the difficulties that arise when combining them, such as reconciling columns that use different vocabulary or recognizing when two tables refer to the same real-world object, were close to the problems the community had been studying for years. Early efforts in this direction also exploited the less regular structure of HTML *lists*: Elmeleegy et al. [62] proposed a domain-independent, unsupervised technique to split list items into fields and recover multi-column relations from them, using a corpus of HTML tables to guide the alignment of fields. The most influential study of this line of work is WEBTABLES [28], which extracted billions of HTML tables from a general-purpose Web crawl and, through statistical classification, isolated a corpus of more than a hundred million high-quality relational tables, by far the largest of its kind at the time.

A central motivation for collecting such tables was the construction of open *knowledge bases* (KBs), that is, large, structured repositories of facts about entities and their relationships. Projects such as DBPEDIA [8] and YAGO [163] built cross-domain KBs from Web sources, and the broader effort on large-scale knowledge collection and curation [127, 179] relied heavily on tabular Web data to populate and extend them. Turning a heterogeneous collection of tables into a single, coherent KB raises its own difficulties. The main one is the alignment of entities and attributes across tables that use different names, granularities, and conventions for the same real-world objects; this connects table acquisition to the long-standing problems of *schema matching* [150] and *entity resolution* [61]. Taken together, this body of work established a first pillar: the automatic *acquisition* and *understanding* of entity-oriented tables, and their use as input for downstream applications.

3.3 Spreadsheets

As the focus broadened beyond HTML, spreadsheets emerged as a second source of tabular data, much harder to process. A spreadsheet is a two-dimensional grid of cells, edited through tools such as *LibreOffice Calc*, *Microsoft Excel* or *Google Sheets*. It is one of the most widely used data authoring formats among users who are not computer scientists, and is common not only in businesses but also among public administrations, NGOs, and statistical agencies that publish open data. This ease of use is also the source of the difficulty. An entity-oriented table has a fixed regularity, with one row per entity and one column per attribute. A spreadsheet, in contrast, gives complete freedom in how data is laid out on the grid: a single sheet may hold several tables, mix titles, multi-level headers, notes, and aggregates, use blank rows as visual separators, and encode meaning through formatting alone. This freedom creates ambiguity at every level: identifying attributes, delimiting tuples, and even recovering the role of each cell. It is also why relational systems cannot simply “ingest” spreadsheets, since the structure that a database expects to be declared is here implicit, and must first be recovered. Two tasks are therefore needed before any relational-style processing can take place: classifying the *role* of each cell (e.g., header, data, title) and *detecting the tables* present on a sheet, that is, locating their boundaries.

Methods for these tasks have followed the general trajectory of machine learning itself. Early approaches were rule- and structure-based, recovering table regions from the connectivity of non-empty cells, as in the WEBSMATCH line of work [48]. A second generation introduced learning: sequence and graphical models, such as the *Conditional Random Field*-based (CRF-based) extraction of Chen and Cafarella [37] and the layout-inference classifiers of Koci et al. [113], improved robustness over hand-crafted rules. *Deep Learning* (DL) then pushed performance further: TABLESENSE, for instance, models table detection as an object-detection problem over the cell grid using convolutional networks [58]. More recently, *Large Language Models* (LLMs) have been used as well, as in SPREADSHEETLLM, which proposes a compact encoding so that sheets fit within an LLM’s context window [59]. Each step has improved accuracy, which is essential to

handle the full diversity of real-world spreadsheets. However, the growing complexity and computational cost of these models is hard to reconcile with a key feature of the setting: the very large volume of data found in the wild. The challenge is therefore to find a balance between predictive quality and scalability. This tension directly motivates our work on spreadsheet understanding (Chapter 6).

3.4 Querying Tables in Natural Language

Acquiring and understanding tables, whatever their form, is only useful if one can subsequently *query* them. Early work on querying Web tables and the KBs built from them relied on lexical matching, ranking tables by keyword and term overlap against their headers, body, and surrounding context. WEBTABLES already supported keyword search over its table corpus [28]; Pimplikar and Sarawagi [148] answered queries expressed as column keywords by matching terms against table fields; and WIKITABLES likewise explored keyword search over tables mined from *Wikipedia* [22], with ranking based on term-frequency signals. Semantic similarity was then introduced to go beyond exact term matches [185], opening the way to learned representations. The question of querying in *Natural Language* (NL) arose naturally, encouraged by progress in the *Natural Language Processing* (NLP) community on language representation. Successive generations of language models, and later large language models, were applied to the problem, and a very large body of work now addresses natural-language access to tables.

The NLP community has largely approached the task at query time: given a question and one table, or a small set of tables supplied alongside it, it identifies the relevant cells or generates an answer, as in TAPAS [94] and dense table retrieval [95]. This framing is natural from an NLP perspective, where the goal is to understand language over a fixed piece of content. It does not, however, address the concerns that are central to data management, since it assumes the relevant table is already in hand. A method that reads each table again for every incoming question does not scale to corpora of thousands or millions of tables, and even indexing reaches its limits when individual tables contain millions of rows and hundreds of columns. Scalability, in both the number of tables and their size, is not a design goal of these methods.

The data management community has therefore taken up the problem, and two complementary lines of work have become particularly active. The first is *Table Question Answering* (TQA): given one table or a small set of tables, return an answer to a natural-language question. Some methods locate the answering cells within a table; others use LLMs to produce a natural-language answer [164, 177, 182, 186, 187]. Beyond this basic case lie harder variants: some questions require taking the union of several tables [140], some require an aggregation within a table, and some have no exact answer in the available data but could be given an approximate one when similar data exists [45]. A closely related problem, often presented as a way to perform TQA, is the translation of natural language to SQL (NL2SQL). Since the community already knows how to query relational tables in SQL, a reliable translator would turn TQA into a solved problem;

this is in particular how aggregation is typically handled. NL2SQL remains an active and unsolved research topic [40, 110, 123].

The second line of work is *Table Discovery* (TD): given a natural-language question or a keyword query, find, within a large corpus, the subset of tables that best match it semantically. This is the retrieval step that TQA assumes when no relevant table is given in advance. Here too, methods have evolved from lexical, frequency-based matching toward neural language representations and, recently, LLMs to facilitate discovery [11, 38, 90, 145, 159, 175]. A recurring obstacle is that tables in the wild often lack the metadata on which discovery depends, and LLM-based methods depend on it especially. Recent work addresses this directly: AUTODDG for instance automatically generates textual descriptions of tabular data to improve findability when metadata is missing or inconsistent [184], while benchmarks and studies of the retrieval step measure how strongly metadata quality affects downstream performance [65, 96, 102]. This work increasingly takes place in the context of *data lakes*, large and loosely governed collections of datasets, where the discovery of joinable and unionable tables has become a research topic on its own [43, 55, 64]. This line of work directly motivates our work on scalable search over heterogeneous corpora of statistical tables (Chapter 5).

3.5 Acquiring Tabular Data from the Web

None of the above has any value unless one can first access large collections of data that reflect the complexity and diversity of the real world. This holds for understanding entity tables or spreadsheets, for answering questions, and for discovering tables over large corpora. Two broad strategies have been used to assemble such collections: building static corpora, and acquiring data directly from the Web.

On the corpus side, the most prominent efforts concern HTML tables. The *Web Data Commons project* released large public corpora of relational Web tables extracted from *Common Crawl* [122], the latter being a freely available, periodically refreshed archive of billions of crawled Web pages [49]. Together, they provided for the first time a common ground for evaluating Web-table systems, since most earlier corpora were owned by search-engine companies and not available to researchers. More targeted corpora have been built for specific tasks, such as FETAQA, whose tables are drawn from *Wikipedia* [139]. These corpora are, however, limited to HTML tables. For spreadsheets, the picture is sparser. WEBSHEET, a Web-crawled corpus of more than four million sheets, was announced together with TABLESENSE [58] but, for compliance reasons, was never released and thus cannot be used by the community. Smaller resources have filled the gap, such as DECO, a few thousand annotated business spreadsheets for cell classification and table detection [116], and FUSE, a corpus of about 250 thousand spreadsheets extracted from *Common Crawl* [17]. A general limitation of such smaller corpora is that they are each specific to a particular domain. More generally, any fixed (static) corpus, while useful for research or benchmarking, may become obsolete or limited for practical use. This motivates the need for tools that can re-build a corpus by gathering up-to-date data from high-quality sources.

This motivates the second strategy: acquiring data from the Web on demand. *Focused crawlers* are the most relevant line of work here. Introduced by Chakrabarti et al. [32], a focused crawler steers the exploration of the Web toward pages about a given topic, using cues such as the textual content of pages and the topical locality of hyperlinks [54, 56]. This is directly useful for building thematic corpora of tables. More recent work has refined the idea with reinforcement learning [91, 121], extended it to the discovery of structured data [130] and of domain-specific content [147], and, closest to our concerns, to the collection of *datasets* on a given theme [183]. Other work has studied adaptive, structure-based crawling more generally [63, 87, 156]. This line of work directly motivates our crawler for robust and scalable data acquisition (Chapter 4).

In summary, making tabular data usable requires addressing three related needs: *acquiring* it from the Web, *ingesting and understanding* its structure, and *querying* it. These three needs structure the present thesis.

3.6 Statistical Tables and their Societal Importance

As discussed in the introduction, statistical tables remain a comparatively underexplored category, even though the data they carry grounds public debate and is increasingly targeted by organized disinformation. A first body of work, originating in NLP, treats fact verification as a labeling task over textual evidence: the FEVER dataset and benchmark, in which claims must be classified as supported, refuted, or lacking sufficient information against *Wikipedia*, has been particularly influential here [168]. Before verification itself comes the problem of deciding which sentences are worth checking, that is, claim detection and check-worthiness estimation, for which dedicated benchmarks have been proposed [6]. A second line of work, closer to data management, verifies claims not against free text but against *structured* data, and in particular against official statistics. Early systems searched for the statistic relevant to a textual claim within a database of such figures [30, 173]. Later systems turned this into a scalable, mixed-initiative process, as in SCRUTINIZER, which organized large-scale, data-driven verification with a human in the loop [107, 108]. Other systems matched claims directly against tabular sources [153]. These efforts share a common premise: verification is only as good as the data one can use, which presupposes the ability to acquire, understand, and query statistical data at scale.

This thesis is rooted in one such effort, the STATCHECK project, a long-running collaboration between database researchers and fact-checking journalists at *Radio France*, the French national broadcasting institution. Its goal is to automatically verify claims circulating in public discourse and on social media against trusted statistical sources [13, 14]. The project makes the three needs identified above concrete and interdependent. It must first *acquire* reference data from a set of known but highly heterogeneous trusted sources, namely national and international organizations that publish statistics in disparate formats and structures; this is precisely the data-acquisition problem we address in Chapter 4. It must then *detect* the check-worthy claims found in social-media streams and textual content, a task that is largely a natural language processing problem and the subject of separate work [15]. Finally, it must *retrieve* the relevant statistics and

compare them against the claim. These statistics are often multidimensional aggregates whose dimensions are organized in hierarchies, that is, the comparatively neglected class of tables discussed above, which raises the search and understanding problems we take up in Chapters 5 and 6. The early `STATCHECK` retrieval prototypes [16] made clear that existing table search and question-answering methods do not scale to the size and structural complexity of real statistical corpora, a limitation that directly motivates the contributions of this thesis.

Efficient Focused Web Crawling for Scalable Data Acquisition

4.1 Introduction

Among the openly accessible data published on the Web, *statistics datasets* (**SDs**, in short) form a distinct and highly valuable category. As discussed in Chapter 3, they differ significantly from the entity-oriented Web tables the community has most studied: their content is mostly numeric, they are multidimensional aggregates, and they are published as standalone files in a variety of formats (CSV, spreadsheets, XML dialects such as SDMX [155], etc.).

Finding *all* the SDs published by an organization is often desirable: to study them as a whole, to compare across organizations, or to populate a corpus for fact-checking. A labor historian may want every SD published by the *ILO*; a journalist may need all *Eurostat* SDs on poverty. Yet current methods to do so are either lacking or inefficient. Some organizations expose an API, e.g., *Eurostat*, but writing custom code for each is cumbersome and brittle, and many websites hosting numerous SDs, e.g., the *ILO*, offer none. Search engines and data portals expose only a small, opaquely selected fraction of existing SDs (Section 4.4.2). A naive, exhaustive crawl is impractical on large websites: beyond its time and space cost, *crawling ethics* impose a wait (typically one second) between requests, so that acquiring a website of one million pages takes eleven days for the waits alone.

We therefore study a *focused crawl*: acquiring within a given website as many **targets** as possible while **minimizing the resources consumed**, e.g., HTTP requests or transferred volume, feasibly on a single machine rather than on search-engine-scale infrastructure. Motivated by SD retrieval, we define targets as *data files* (CSV, spreadsheet, etc.), though the notion generalizes to any target definition. Existing focused crawlers [32, 56, 63, 87, 91, 121, 130] often rely on assumptions that do not hold here, such as the topical locality of pages within a website [54]. We compare against representative ones, including a generic FOCUSED baseline [32, 56], the topic-driven TRES [121], and a non-topical TP-OFF baseline adapted from textual-diversity crawling [63]. We thus

address the problem of **retrieving SDs within a website as efficiently as possible, without prior knowledge of its structure or content.**

Contributions

In this chapter, our contributions are as follows.

- (1) We formalize our graph crawling problem and prove that optimally solving it is NP-hard (Section 4.2).
- (2) We propose SB-CLASSIFIER in Section 4.3, a novel RL-based focused crawler relying on two crucial hypotheses:
 - (i) *Links found on similar tag paths within HTML pages (a tag path [66, 131] goes from the page root to a hyperlink tag such as <a>) lead to similar content*
 - (ii) *We can learn, from the tag path structure, which tag paths are likely to lead to links towards targets.*
- (3) We evaluate SB-CLASSIFIER through extensive experiments on diverse websites totaling 22.2 million pages. Our approach outperforms all baselines; on large websites, it **retrieves 90% of the targets while accessing only 20% of the pages** (Section 4.4). We complement this with a manual characterization of the websites and an analysis of where targets are typically located within pages (Section 4.4.1).
- (4) We conduct a thorough experimental analysis:
 - (i) We validate that the RL component effectively learns to distinguish tag paths by their expected reward (Section 4.4.7);
 - (ii) We present the MIME-type and URL-extension filtering mechanism and ablate its contribution to crawl performance (Section 4.4.9);
 - (iii) We provide a comprehensive hyperparameter study across all evaluated websites (Section 4.4.6);
 - (iv) We evaluate the URL classifier in isolation and assess how its quality affects overall crawl efficiency (Section 4.4.8);
 - (v) We present and visualize the early-stopping mechanism (Section 4.4.10).

This chapter is organized as follows. Section 4.2 formalizes the crawling problem as graph exploration and establishes its intractability. Section 4.3 presents the SB-CLASSIFIER approach. Section 4.4 reports the experimental evaluation. Section 4.5 surveys related work, and Section 4.6 concludes. A *GitHub* repository [74] provides code to reproduce all experiments.

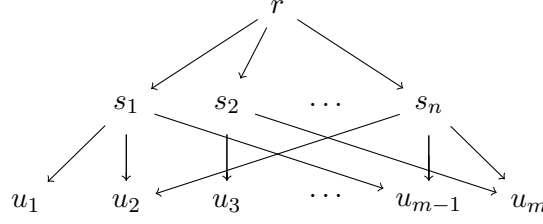


Figure 4.1: Graphical summarization of the graph G_{sc}

4.2 Graph-Based Modeling of Web Data Acquisition

We formalize our SD acquisition as a *graph crawling problem*. We show that the decision version is NP-hard, even if the website is known before the crawl (which in practice is not) in Section 4.2.1. Then we detail the mapping of our problem into the graph crawling framework, including a crucial choice of labels for the graph edges, in Section 4.2.2.

4.2.1 Graph Crawling Problem and Intractability

We model a website as a rooted, node-weighted, edge-labeled directed graph. Each node represents a webpage, and each edge is a hypertext link leading from one to another. We fix a countable set $\mathcal{L}$ of labels, e.g., finite sequences of character strings.

Definition 1 (*Website Graph*)

A website graph G is a tuple $(V, E, r, \omega, \lambda)$, with: V a finite set of nodes (representing webpages); $E \subseteq V^2$ a set of edges (representing hyperlinks); $r \in V$ the root of the graph (the input webpage); $\omega : V \rightarrow \mathbb{R}^+$ a cost function assigning a positive weight to every node (the cost of retrieving that page); $\lambda : E \rightarrow \mathcal{L}$ a labeling function, assigning a label to each link found in a page.

On such a graph, we define a *crawl* and its *cost*:

Definition 2 (*Crawl*)

A crawl of a website graph G is an r -rooted subtree $T = (V', E')$ of (V, E) . Its total cost, $\omega(T)$, is defined as $\sum_{u \in V'} \omega(u)$.

The graph crawling problem is formalized as follows:

Problem 3 (*Graph Crawling Problem*)

Given a website graph $G = (V, E, r, \omega, \lambda)$ and a subset V^* of targets in V , the graph crawling problem is to find a crawl $T = (V', E')$ of G with $V^* \subseteq V'$, of minimal total cost.

We define the *frontier* of a crawl $\{u \in V \setminus V' \mid (v, u) \in E, v \in V'\}$: nodes that have not been crawled but are pointed to from nodes that have been. Figure 4.2 shows a

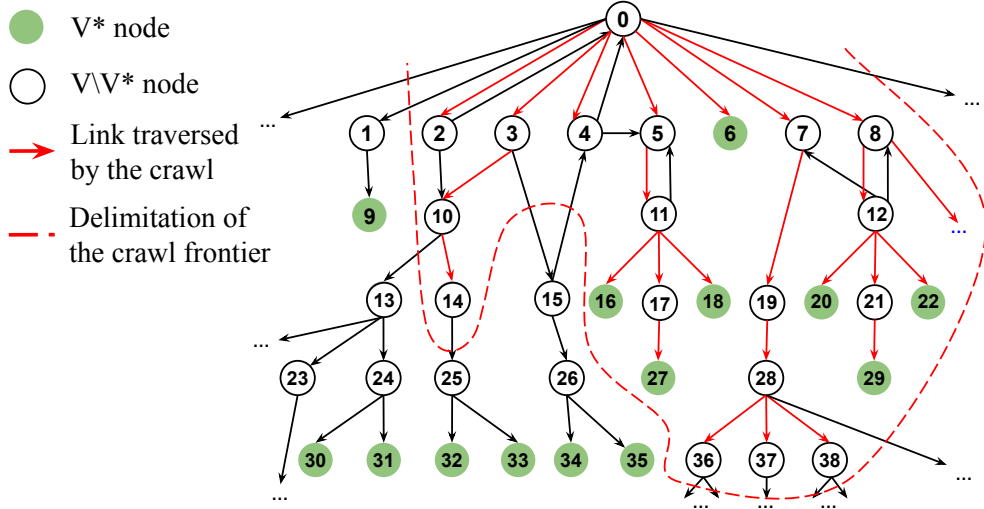


Figure 4.2: Sample website, crawl, and frontier

website graph as well as a possible crawl and its frontier. Even assuming the graph is fully known in advance, the graph crawling problem is intractable:

Proposition 4

Given a website graph $G = (V, E, r, \omega, \lambda)$, a subset $V^* \subseteq V$ and some $B \in \mathbb{R}^+$, determining whether there exists a crawl $T = (V', E')$ of G such that $V^* \subseteq V'$ and $\omega(T) \leq B$ is NP-complete; hardness holds even when ω is a constant function. This holds even when nodes in V^* do not have any out-links in G .

Proof: To show NP-completeness, we must show that the problem belongs to NP and is NP-hard. Let us start with the upper bound. Given a graph $G = (V, E, r, \omega, \lambda)$, we guess a subgraph $T = (V', E')$ of G (which is a polynomial-sized guess). In polynomial time, we check whether T is an r -rooted tree (i.e., whether it is connected, includes r , r has indegree 0 and other nodes indegree 1), we check that V' contains all nodes of V^* , and we check that $\omega(T) \leq B$. We accept if and only if these conditions are all satisfied. This yields a nondeterministic polynomial-time algorithm, meaning the problem is in NP.

We now move to the lower bound. Our crawling problem can be seen as a directed variant of the well-known NP-complete *Steiner Tree* [67] problem. NP-hardness of the directed Steiner tree problem is mentioned in the literature (see, e.g., [178]), but as it is not formally shown there, we prefer for completeness of the presentation reducing from the set cover problem, a classic NP-hard problem [67]. We denote $\mathcal{U} = \{u_1, \dots, u_m\}$ a set of m elements called the universe. We also define a collection $\mathcal{S} = \{s_1, \dots, s_n\}$ of n non-empty subsets, each of them containing some elements of $\mathcal{U}$, such that $\bigcup_{s \in \mathcal{S}} s = \mathcal{U}$. In its decision version, the set cover problem consists in determining, given such a universe and collection, and a natural integer B , whether there exists a *cover* $\mathcal{C} \subseteq \{s_1, \dots, s_n\}$ such that $|\mathcal{C}| \leq B$ and $\bigcup_{s \in \mathcal{C}} s = \mathcal{U}$.

We now propose a polynomial-time many-one reduction of the set cover problem to an instance of the graph crawling problem. We create a website graph $G_{sc} = (V_{sc}, E_{sc}, r, \omega, \lambda)$

as follows. We set V_{sc} to be $\{u_1, \dots, u_m, r, s_1, \dots, s_n\}$, including representations for every element of the universe $\mathcal{U}$, every set of the collection $\mathcal{S}$, as well as a distinct root r (by abuse of notation, we do not distinguish between elements of $\mathcal{U}$, $\mathcal{S}$ and the way they are represented in V_{sc}). We define E_{sc} as $\{(r, s_i) \mid i \in \{1, \dots, n\}\} \cup \{(s_i, u) \mid u \in s_i, i \in \{1, \dots, n\}\}$. In other words, in G_{sc} from the origin (root) r , we model each element of $\mathcal{S}$ as a vertex, that can be reached following a dedicated (directed) edge. Finally, for each new vertex $s_i \in \mathcal{S}$, we have as many outgoing edges as there are elements of $\mathcal{U}$ in s_i . Finally, we set ω to be the constant function that assigns cost 1 to every vertex, and λ to be some constant function. The result is a graph in the form of a tree of depth 2, depicted in Figure 4.1. We fix V^* to be $\mathcal{U}$. Observe that nodes in $\mathcal{U}$ do not have any outgoing links. We state that there exists $\mathcal{C} \subseteq \{s_1, \dots, s_n\}$ such that $|\mathcal{C}| \leq B$ and $\bigcup_{i \in \mathcal{C}} s_i = \mathcal{U}$ if and only if there exists a crawl T_{sc} of G_{sc} containing all elements of V^* and of total cost $\omega(T_{sc}) \leq |\mathcal{U}| + B + 1$.

Let us explain why this reduction is polynomial-time. In set cover, the universe $\mathcal{U}$ can be described by the number m of its elements, with representation size $\Theta(\log m)$. Each set of $\mathcal{S}$ needs to list every element within this set, so a set s_i has representation size $\Theta(\log m \times |s_i|)$. Finally, B has representation size $\Theta(\log B)$. This yields a total input size of $\Theta((\log m) (\sum_{i=1}^n |s_i| + 1) + \log B)$. Note that $\sum_{i=1}^n |s_i| \geq \max(m, n)$ so this is $\Omega(\max(m, n) \log m + \log B)$. But then, the construction depicted in Figure 4.1 can clearly be done in time polynomial in m and n (namely, in $O(m \times n)$ in the worst case where every set of the collection contains every element). The reduction is therefore polynomial-time.

We now proceed to show equivalence between the initial problem known to be NP-hard (set cover) and the graph crawling instance presented above. First, suppose that there exists $\mathcal{C} \subseteq \{s_1, \dots, s_n\}$ such that $|\mathcal{C}| \leq B$ and $\bigcup_{i \in \mathcal{C}} s_i = \mathcal{U}$. Then consider the crawl T_{sc} of G_{sc} formed by including r , every element of $\mathcal{C}$ using the edge from r to that element, and every edge from an element of $\mathcal{C}$ to an element of $\mathcal{U}$. Since $\mathcal{C}$ is a cover, this includes all elements of $\mathcal{U}$. The total cost of this crawl $\omega(T_{sc}) = 1 + |\mathcal{C}| + |\mathcal{U}| \leq |\mathcal{U}| + B + 1$.

Now suppose that there exists a crawl T_{sc} of G_{sc} of total cost $\omega(T_{sc}) \leq |\mathcal{U}| + B + 1$. Note that by definition of ω , the cost is just the number of nodes in T_{sc} , and this crawl necessarily includes the root r as well as all vertices of $\mathcal{U}$. The remaining $\leq B$ vertices are therefore vertices of $\mathcal{S}$. We pose $\mathcal{C}$ to be those. Then $|\mathcal{C}| \leq B$ and since T_{sc} is a crawl, for every $u \in \mathcal{U}$, there exists at least one $s \in \mathcal{C}$ such that the edge (s, u) is in T_{sc} , meaning that $u \in s$. We indeed have $\mathcal{U} = \bigcup_{s \in \mathcal{C}} s$. ■

Since optimal methods are out of reach (websites may have millions of pages), we need heuristic methods with low practical costs.

4.2.2 Web Graph Modeling

We complete our modeling of Web data acquisition as an instance of the graph crawling problem by detailing the graph G , the target subset V^* , and the edge labeling function λ . The choice of λ will turn out to be crucial for the performance of our approach.

We identify pages by their URL and fix r , the website root, as the start of the crawl. Next, we need to identify which pages belong to the website graph. Lacking a commonly agreed definition of a website's boundary [4, 156], we use a pragmatic approach. A URL is considered to be part of the same website as the root r if its *hostname* (the part of the URL which is after the scheme but before the path, with a potential "www." prefix

excluded) *is a subdomain of the hostname of r* . V contains all such URLs. For instance, if r is <https://www.A.B.com/index.php>, the URLs <https://www.A.B.com/folder/content.php> and <https://www.C.A.B.com/page.html> are part of V , but <https://www.B.com/page.php> and <https://edbticdt2026.github.io/> are not.¹ An edge $(u, v) \in E$ exists if u links to v via HTML tags like `<a>`, `<area>`, `<iframe>`.

Since our goal is to find SDs, target pages are those whose *Multipurpose Internet Mail Extensions* (MIME) type is in a *user-defined* list (e.g., `text/csv`, `application/pdf` or `application/vnd.ms-excel`, etc.). Non-target types include `text/html`, `video/*`, `image/*`, etc. The MIME type can be obtained via HTTP HEAD requests; to generalize to other content than SDs, any other target MIME type set can be used. We provide below the complete list of the 38 MIME types used in our implementation to identify target resources. This selection was made in the context of implementing our crawler for the collection of SDs; its rationale is further discussed in Section 4.4.9.

application/vnd.openxmlformats-officedocument.presentationml.presentation		
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet		
application/vnd.openxmlformats-officedocument.wordprocessingml.document		
application/vnd.openxmlformats-officedocument.wordprocessingml.template		
application/vnd.oasis.opendocument.presentation		
application/vnd.oasis.opendocument.spreadsheet		
application/vnd.oasis.opendocument.text		
application/vnd.ms-excel.sheet.macroenabled.12		
application/x-7z-compressed	application/x-rar-compressed	text/comma-separated-values
text/x-comma-separated-values	application/x-zip-compressed	application/zip-compressed
application/csv	application/json	application/msword
application/octet-stream	application/pdf	application/rdf+xml
application/rss+xml	application/vnd.ms-excel	application/vnd.rar
application/x-csv	application/x-gtar	application/x-gzip
application/xml	application/x-pdf	application/x-tar
application/x-yaml	application/yaml	application/zip
text/plain	text/csv	text/json
text/x-csv	text/x-yaml	text/yaml

To each edge e , the edge labeling function λ associates as label $\lambda(e)$ the **tag path** derived from the HTML page’s DOM structure [180], the standard tree-based representation of an HTML page. The tag path includes the **full path of HTML tags from the root to the hyperlink tag** (`a`, `area`, `iframe`, etc.), along with class and id attributes describing additional structural and styling information. For example, a label might be “`html body div#main ul.datasets li a`”, where ‘#’ prefixes the HTML ID, and ‘.’ indicates a class. This labeling makes it very likely that *links labeled with similar tag paths lead to similar content types*, even across different webpages of the same website. Figure 4.3 zooms within an HTML page. It shows 5 tag paths (the tags are omitted for readability), leading to: a text paragraph, an image and 3 hyperlinks to other pages (1

¹The reason for the special handling of a “www.” prefix is that many (but not all...) websites use it as a prefix for the domain name of the Web server.

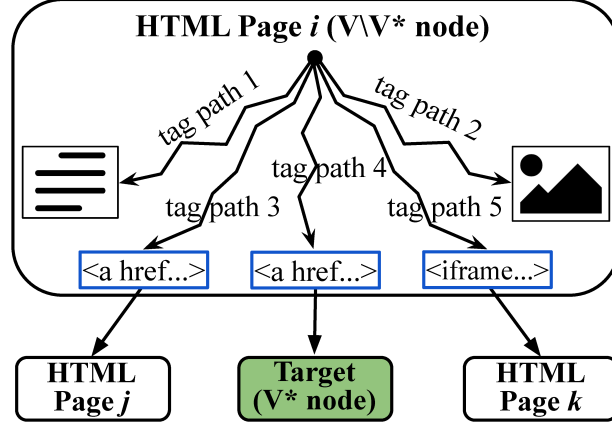


Figure 4.3: Tag paths in an HTML page

target, and 2 HTML pages). λ captures exactly the tag paths leading to each hyperlink. We consider two cost functions ω :

- (i) counting HTTP requests, $\omega(u) = 1$ for all $u \in V$;
- (ii) measuring the exchanged data volume, especially data volume received by the crawler, with $\omega(u)$ defined as the page size, which is unbounded in theory.

We also account for the cost c of *determining if a vertex is in V^** , typically via an HTTP HEAD request. If ω counts requests, then $c(u) = 1$; if ω measures volume, $c(u)$ is based on HEAD response size (much smaller than $\omega(u)$). Section 4.3.3 discusses a MIME type prediction method, resulting in a small, amortized c cost, which we can view as constant.

4.3 Reinforcement Learning-Based Crawling

Our goal in efficient crawling is to retrieve as many targets as possible at minimal cost, without prior knowledge of the website’s size or structure. We hypothesize that *an edge’s label λ (its tag path in the page containing the hyperlink) can be used to learn which edges lead to target-rich pages*. Obtaining and leveraging such insights require both *exploration* (to find promising paths) and *exploitation* (to retrieve targets), which we achieve via *reinforcement learning* (RL) [167]. Below, we present the environment (states and actions) of our crawling task (Section 4.3.1), and the learning algorithm (or *agent*) that evolves in it (Section 4.3.2). We describe the URL classifier we use to compute the agent’s rewards in Section 4.3.3, and the overall crawling algorithm in Section 4.3.4.

4.3.1 Environment: States and Actions

In RL, an *agent* evolves in an *environment*. It starts from an initial *state*, where it can choose among a set of *actions*. Each *action* leads to a new *state*, where other actions can be chosen. Each choice leads to a *reward*. The goal of the agent is to learn a *policy* (a

mapping from states to actions), that maximizes the total reward. This model is known as a *Markov Decision Process* (MDP) [21, 149]. For our crawler, one might consider using “the currently crawled webpage” as *state*. However, this is not suitable: the current webpage does not reflect previous choices, as it may be reachable via multiple paths. Also, an efficient crawler should not visit any webpage twice, whereas learning supposes multiple visits to a state to refine and then leverage information learned in that state. To keep the model simple and lightweight (discussed in Section 4.5), we use a **single-state model**, where the agent is perpetually in exactly one state. What matters then is only the set of *actions* that the agent can follow at each crawl step. Intuitively, an action is to navigate along a link. We delve into action details below.

4.3.2 Grouping Links into Actions

Following our hypothesis that *links with similar labels (tag paths) have similar values*, i.e., likelihood of leading to targets, we model an action as a *group of similar links*, with the semantics that taking the action amounts to crawling along one of these links.

Each link is labeled with its tag path, so we need a measure of similarity between tag paths. To that effect, we represent each tag path by a *numerical bag-of-words (BoW) vector* p , over the n -gram vocabulary built from all tag paths encountered so far. Note that n -grams preserve the order of HTML node names appearing in tag paths; in our context, this order is significant (as Section 4.4 confirms). Since the n -gram vocabulary is *dynamically* built during the crawl, BoW vectors for tag paths encountered at different times have different lengths. To still compute distances between them, we first *project each tag path into a fixed-dimensional vector* of size $D = 2^m$ for a chosen $m > 0$. We do this as follows:

1. We fix a *hash function* h such that:

$$h : x \mapsto \left\lfloor \frac{(\Pi \times x) \bmod 2^w}{2^{w-m}} \right\rfloor$$

which maps any integer x to a number between 0 and $D - 1$. Its parameters Π (a large prime number) and w are fixed, and chosen such that $w > m$.

2. Calling h on each position between 0 and $d - 1$, where d is the length of the BoW vector p , maps it to a new position between 0 and $D - 1$. This enables us to transform p into a D -dimensional p_D vector, where for every $0 \leq i < d$, $p_D[h(i)] = p[i]$.
3. Potential collisions of h (i.e., $i_1 \neq i_2$ with $h(i_1) = h(i_2)$) are resolved by setting $p_D[h(i_1)] = p_D[h(i_2)]$ to the *mean* of all elements of p , at positions which collide with i_1, i_2 . Positions $0 \leq j < D$ not hit by any i ($h(i) \neq j$ for all $0 \leq i < d$) are set to $p_D[j] = 0$. This happens, e.g., at the beginning of the crawl, when the set of HTML element names seen so far is small (thus d is small).

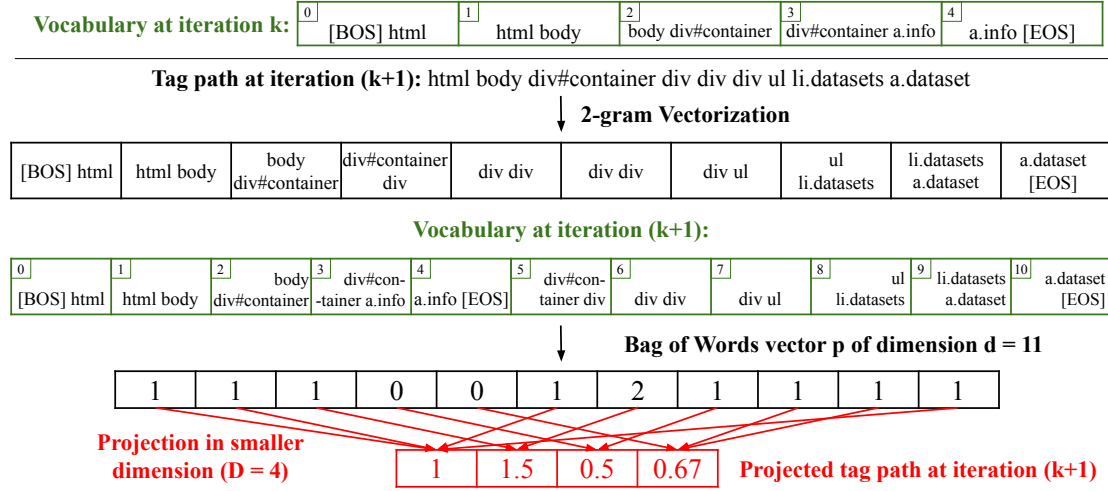


Figure 4.4: Mapping of a tag path into a fixed-size vector.

Figure 4.4 illustrates tag path projection for $D = 2^m = 4$, $w = 11$ and $\Pi = 766\,245\,317$. At iteration k , the 2-gram vocabulary contains $d_k = 5$ elements (including special tokens BOS and EOS denoting beginning and end of stream). At iteration $k + 1$, a new tag path is vectorized into a 2-gram vector of dimension 10, and the vocabulary is updated accordingly, totaling $d_{k+1} = 11$ elements. From it, we compute the d_{k+1} -dimensional BoW vector p_{k+1} . The hash function h maps each input dimension $i \in \{0, \dots, d_{k+1} - 1\}$ to $\{0, \dots, D - 1\}$, e.g.:

$$h(2) = \left\lfloor \frac{(766\,245\,317 \times 2) \bmod 2048}{512} \right\rfloor = 1.$$

Collisions can occur, e.g., $h(4) = h(8) = h(9) = 3$, so:

$$p_{D_{k+1}}[3] = \frac{p_{k+1}[4] + p_{k+1}[8] + p_{k+1}[9]}{3} \approx 0.67.$$

This yields a 4-dimensional $p_{D_{k+1}}$.

Our next task is to find, for each projected tag path p_D , the nearest action a_n among all known actions A . Conceptually, an action is an evolving cluster of similar tag paths with a centroid: the mean of all projected tag paths. For efficiency, we only store the centroid, which updates as the action's set of tag paths evolves.

Algorithm 1 computes the nearest action of each p_D if it is close enough; otherwise, it creates a new action. Action centroids are stored in a *Hierarchical Navigable Small Worlds* (HNSW) [128] index $\mathcal{I}$, chosen for its highly efficient updates of centroids as new tag paths join. While maintaining and querying this index might be costly in CPU time, it is negligible compared to crawl time (waiting between two requests, or webpage downloads). The index also estimates the *cosine similarity* between p_D and its closest centroid: if above a threshold θ , the tag path is added to the action; otherwise, a new

Algorithm 1: Finding the action for a hyperlink l , $\lambda(l) = p$

Input: Action set A , projected tag path p_D
Output: Action a

```

1  $a_n \leftarrow$  Approximate nearest neighbor (from index  $\mathcal{I}$ ) of  $p_D$ ;
2  $s_n \leftarrow$  Cosine similarity between  $a_n$  and  $p_D$ ;
3 if  $s_n \geq \theta$  then
4   | Update centroid of  $a_n$  in  $\mathcal{I}$  with respect to  $p_D$ ;
5 else
6   | Add a new entry  $a_{\text{new}}$  to  $\mathcal{I}$ , with value  $p_D$ ;
7   | Add  $a_{\text{new}}$  to  $A$ ;
8   |  $a_n \leftarrow a_{\text{new}}$ ;
9 return  $a_n$ 
    
```

Algorithm 2: Online URL classifier procedure

Input: U , a URL
Output: $l_U \in \{\text{"HTML"}, \text{"Target"}\}$, a class

```

1 if  $|X| \geq b$  then
2   |  $X_{\text{mat}} \leftarrow$  2-gram bag-of-words of each URL in  $X$ ;
3   |  $\mathcal{C}$  is incrementally trained on batch  $(X_{\text{mat}}, y)$ ;
4   |  $(X, y) \leftarrow (\emptyset, \emptyset)$ ;
5   |  $\text{initial\_training\_phase} \leftarrow \text{False}$ ;
6 if  $\text{initial\_training\_phase}$  then
7   |  $l_U \leftarrow$  MIME type class from HTTP HEAD on  $U$ ;
8   | Add  $(U, l_U)$  to  $(X, y)$  and return  $l_U$ ;
9 else
10  |  $U_{\text{vec}} \leftarrow$  2-gram bag-of-words vector of  $U$ ;
11  | return  $\mathcal{C}(U_{\text{vec}})$ 
    
```

action composed of only p_D is created. As our experiments show (Section 4.4), the choice of θ significantly impacts the agent performance. In extreme cases, $\theta = 0$ groups all tag paths into a single action, preventing learning as the agent selects paths randomly, whereas $\theta = 1$ creates a separate action for each path, so the agent only explores and never exploits. A useful threshold balances enough actions to separate interesting paths from others, while still enabling learning and exploitation.

The family of reinforcement learning models for single-state environments is called *Multi-Armed Bandits* (MABs) [167]. The standard, deterministic, MAB algorithm optimizing the exploration–exploitation trade-off is *Upper-Confidence Bound* (UCB) [7], which implements *optimism in the face of uncertainty*: at each step, a score is computed for each action, and the highest-scoring action is selected.

The MAB score of an action a at time $t+1$, denoted $s_{\text{MAB}}^{t+1}(a)$, combines an exploitation term (the mean of rewards obtained so far for a), and an exploration term, reflecting

how often a has been selected relative to the total crawl history:

$$s_{\text{MAB}}^{t+1}(a) = R_a^t + \alpha \sqrt{\frac{\log(t)}{N_t(a) + \varepsilon}}$$

where R_a^t is the mean reward for action a up to time t , α weights exploration vs. exploitation, $N_t(a)$ counts how many times a has been selected, and $\varepsilon > 0$ prevents division by zero when $N_t(a) = 0$.

Since each URL is visited only once, an action may become empty after all its URLs are visited. The *Awake Upper-Estimated Reward* (AUER) [112] adapts UCB to cope with actions becoming unavailable, or *sleeping*. Following the AUER model, our score becomes:

$$s_{\text{SB}}^{t+1}(a) = \mathbb{1}_a(t) \times \left(R_a^t + \alpha \sqrt{\frac{\log(t)}{N_t(a) + \varepsilon}} \right)$$

where $\mathbb{1}_a(t) = 1$ if action a has remaining unvisited links at time t , and 0 otherwise. After selecting a , our crawler *randomly chooses an unvisited link $l \in a$ to traverse next with equal probability*. How should we reward the crawling along $l \in a$? As our goal is to find *new targets*, the reward should reflect the number of *not-yet-discovered links to targets*, contained in the HTML page h reached via l : this “novelty” condition encourages the agent to focus on unseen targets. For instance, if h has 12 links, 5 leading to targets, and 2 already retrieved, the reward is 3. This simple choice performs well, as shown in Section 4.4.

Yet, a challenge remains: we must compute the reward of all links in h before following any, since each traversal incurs a crawling cost. We therefore need a fast *estimation*, applicable to *any* link, even if some are never followed. For this, we introduce an *online URL classifier* that estimates rewards by inspecting links in the newly acquired page h . We detail it in Section 4.3.3.

Last, we discuss the parameter α in our learning agent. UCB and AUER are optimal for $\alpha = 2\sqrt{2}$ [7, 112], but only under standard conditions where the rewards are normally distributed in $[0, 1]$. In our case, rewards (counts of new targets) are unbounded and typically do not follow a normal distribution (see Section 4.4.7), which is desirable: otherwise, it would mean that most HTML pages contain links to targets. If this held, we would have to visit the entire website, or at least a vast share, to retrieve most targets.

Setting up α to guarantee optimality in our case would require knowing the reward distribution, which is not possible. We therefore keep $\alpha = 2\sqrt{2}$, which gives good results in practice, across very varied reward distributions (as shown in Section 4.4.1 and further discussed in Section 4.4.7).

4.3.3 Estimating Rewards with URL Classifier

Determining the reward of an action taken by our agent (crawling a link) requires assessing a page’s MIME type from its URL. Doing HTTP HEAD requests is costly and requires spacing for crawling ethics, while using the URL extension (e.g., `.html`, `.csv`) fails for extensionless links such as <https://www.justice.gouv.fr/en/node/9961>, or others

within *ILO*². To be efficient and generic, we devise an online URL classifier³ to estimate rewards.

For our purposes, any URL belongs to one of three classes: “HTML”, “Target” or “Neither”. HTML pages are added to the crawl frontier, while a target contributes to the crawler’s reward. The “Neither” class includes non-HTML pages, URLs with non-target MIME types, or URLs lacking a MIME from the server. In our experience, over 92% of “Neither” cases correspond to HTTP **4xx** or **5xx** codes, designating errors on client or server.

We initially set our classifier to assign one of these three classes to each link found in a newly crawled page h . However, it proved unable to distinguish URLs leading to 4xx or 5xx errors from those that resolve fine. Intuitively, these are often very similar to accessible “HTML” or “Target” URLs. But *different classification errors have different impacts on crawling*:

1. Misclassifying a “Neither” URL as “HTML” or “Target” only incurs the moderate cost of following it (later, respectively, immediately), as it will likely throw an error. To reduce such visits, we apply filters based on user-defined MIME types and URL extensions (see Section 4.3.4).
2. Misclassifying “HTML” or “Target” as “Neither” *completely withdraws the page from the crawl*: if it is a target, we miss it; if it is HTML, we miss all URLs that might be found by following it. This can make the crawler miss huge parts of the website.

To avoid the latter, our classifier is trained only on the “HTML” and “Target” classes, even though some URLs are neither.

Algorithm 2 details the training and usage of our classifier $\mathcal{C}$. It is a logistic regression model [99], iteratively trained through epochs of *Stochastic Gradient Descent* (SGD) [25], with batch size b . It takes as input a vector of character-wise 2-grams of the URL. For instance, the URL <https://www.A.com/data/file.csv> is transformed into a list [ht, tt, tp, . . . , .c, cs, sv]. We associate to each pair of usual ASCII characters⁴ an ID, and encode each URL as a BoW over this vocabulary. The choice of model and features is discussed in Section 4.4.6.

As Algorithm 2 shows, in the first training epoch, we label b URLs via HTTP HEAD requests, with X the URL set and y their labels. Based on this, we compute rewards and assign URLs to the frontier $\mathcal{F}$ or targets V^* . After this epoch, we set `initial_training_phase` to false, and use the classifier to infer URL classes without additional HEAD requests. The classifier is further improved through *online training*: during execution, each HTTP GET issued by the crawler contributes an annotated (URL,

²E.g., <https://www.ilo.org/publications/elevating-potential-rural-youth-paths-decent-jobs-and-sustainable-futures-1>.

³Observe that here we classify *URLs*, whereas in the previous section we clustered *edge labels*, which, crucially to our method, are *tag paths* within HTML pages (Figure 4.3).

⁴ASCII digits, upper and lower case letters and main special characters.

Algorithm 3: Efficient target retrieval

Input: r the initial page to crawl

```

1  $(A, \mathcal{F}, T) \leftarrow (\emptyset, \emptyset, \emptyset);$ 
2  $(\beta, t) \leftarrow (0, 0);$ 
3  $u \leftarrow r;$ 
4 Add  $r$  to  $\mathcal{F};$ 
5 while  $|\mathcal{F}| > 0$  and  $\beta \leq B$  do
6   if  $|A| > 0$  then
7      $a_c \leftarrow \arg \max_{a \in A} \mathbb{1}_a(t) \left( R_{\text{mean}}(a) + \alpha \sqrt{\frac{\log(t)}{N_t(a) + \varepsilon}} \right);$ 
8      $u \leftarrow$  Select a link from  $\mathcal{F}$  mapped with action  $a_c$  uniformly at random,
       and  $N_t(a_c) \leftarrow N_t(a_c) + 1;$ 
9   else
10     $u \leftarrow$  Select a link from  $\mathcal{F}$  uniformly at random;
11   $\text{crawl\_next\_page}(u, a_c)$  (Algorithm 4);
```

class) pair, added to X and y for incremental training when the batch size b is reached. In this way, after the first epoch, we get labels at no extra cost.

Since we want to both *use* the classifier as soon as possible and *train* it often to improve its accuracy, we must set b relatively small. This online method adapts to potential changes in URL forms, e.g., when the crawl discovers new parts of the website where URLs are formatted differently. An offline method would not be able to adapt, even if trained on many examples initially.

4.3.4 Crawling Algorithm

We now present the overall crawling procedure (Algorithm 3). Initially, the tree T of crawled pages, the set of actions A , and the frontier $\mathcal{F}$ are empty. The budget β spent by the crawler and the crawling step t are set to 0. The crawler starts by visiting the original link r , continuing until all links are visited (the website is completely crawled) or the maximum budget B is reached. At each step, if A is non-empty, the learning agent chooses an action a_c following the Sleeping-Bandit (SB) procedure (Section 4.3.2) and chooses a link from $\mathcal{F}$ associated with a_c uniformly at random. Then, it crawls the page behind this link with Algorithm 4, and if its MIME type matches a predefined blacklist during download, its retrieval is immediately interrupted; this is typically used to avoid multimedia content.

Algorithm 4 starts by retrieving the HTTP GET response: status, MIME type from the header, and body (content) of the webpage. The request cost is added to the budget β . The status may fall into one of three categories:

1. Errors (on the client or server side) marked by **4xx** or **5xx** statuses, do not yield new links or targets.

Algorithm 4: Crawl next page

Input: u the URL to be crawled, a_c the associated action

```

1 Add URL  $u$  to  $T$ ;
2  $t \leftarrow t + 1$ ;
3 http_response, mime_type, body  $\leftarrow$  Result of the request on URL  $u$  with HTTP
  GET, and update  $\beta$  accordingly;
4 if request interrupted (banned MIME type) then return;
5 if http_response is 4xx or 5xx (error) then return;
6 else if http_response is 2xx (success) then
7     if mime_type  $\neq \emptyset$  then
8         if "HTML"  $\subset$  mime_type then
9             Add ( $u$ , "HTML") to  $(X, y)$ ;
10             $U_{\text{new}} \leftarrow$  All hyperlinks in body pointing to the same website as  $r$ ;
11        else if mime_type  $\in L$  then
12            Add ( $u$ , "Target") to  $(X, y)$ ;
13            Add target (the content of body) to  $V^*$ ;
14        return;
15 else if http_response is 3xx (redirection) then
16      $u \leftarrow$  Location from HTTP GET result on  $u$ ;
17     if  $u \notin T \cup \mathcal{F}$  then
18         crawl_next_page( $u$ );
19     return;
20 reward  $\leftarrow 0$ ;
21 for  $u_{\text{new}} \in U_{\text{new}}$  such that  $u_{\text{new}} \notin T \cup \mathcal{F}$  do
22     if has_blocklisted_extension( $u_{\text{new}}$ ) then continue;
23      $l_{u_{\text{new}}} \leftarrow$  class_of_url( $u_{\text{new}}$ ) (Algorithm 2);
24     Update  $\beta$  if HTTP HEAD request was made;
25     if  $l_{u_{\text{new}}} = \text{"HTML"}$  then
26          $a_n \leftarrow$  map_link_to_action( $A, u_{\text{new}}$ ) (Algorithm 1);
27         Add  $u_{\text{new}}$  to  $\mathcal{F}$ ;
28     else if  $l_{u_{\text{new}}} = \text{"Target"}$  then
29         crawl_next_page( $u_{\text{new}}$ );
30         reward  $\leftarrow$  reward + 1;
31 if  $u \neq r$  then
32      $R_{\text{mean}}(a_c) \leftarrow R_{\text{mean}}(a_c) + \frac{\text{reward} - R_{\text{mean}}(a_c)}{N_t(a_c)}$ ;
    
```

2. Redirections, indicated by **3xx** statuses, include an extra "Location" header with the redirection URL: if uncrawled, the crawler follows and processes it.
3. A **2xx** status means the server successfully responds with either an HTML page, a target (whose MIME type is in the list L of Section 4.2.2), or neither. In the

first two cases, X and y are updated. For HTML pages, new hyperlinks U_{new} are extracted from the page, keeping only in-website links.

Each new link not in $\mathcal{F}$ or T is classified using Algorithm 2, except when its extension matches a predefined blacklist established before the crawl.

If it points to an HTML page, it is mapped to an action using Algorithm 1. The set A is updated by either *changing an action’s centroid* or *adding a new action*, and the chosen action a_n is mapped to the link, which is then added to frontier $\mathcal{F}$. If the link leads to a target, we immediately retrieve it by recursively calling Algorithm 4, and the reward is updated. Finally, if the page crawled by the current call of Algorithm 4 is not the starting page, the mean reward for a_c is updated to reflect the last computed reward.

4.4 Experimental Evaluation

We now present our experimental evaluation: websites used (Section 4.4.1), poor search engine performance for this task (Section 4.4.2), baselines (Section 4.4.3), crawling setup (Section 4.4.4), comparison of our crawler to baselines (Section 4.4.5), and impact of hyperparameters (Section 4.4.6). Section 4.4.7 examines our Sleeping Bandit (SB) algorithm’s effectiveness in grouping links into coherent and reward-based groups; Section 4.4.8 analyzes the design and quality of the URL classifier; Section 4.4.9 ablates the multimedia filtering mechanism; and Section 4.4.10 presents the early stopping mechanism.

4.4.1 Websites and Data Sources

Table 4.1 lists the 18 websites used in our experiments. As our primary application arose from a collaboration with French journalists [14] from *Radio France*, six websites are French, public, and governmental: the *French National Assembly* (*as*), *French Local Communities* (*cl*), *French Council for Statistical Information* (*cn*), and the ministries of *Interior* (*in*), *Education* (*ed*), and *Justice* (*ju*). Each contains material on government branches, including SDs (our crawler’s targets). We also include eight multilingual websites: the UN’s *International Labour Organization* (*il*), *French Official Statistical Institute* (*is*), *Japan’s Ministry of Internal Affairs* (*jp*), *OECD* (*oe*), *Open Knowledge Foundation* (*ok*), *Qatar’s Official Statistical Service* (*qa*), the UN *World Health Organization* (*wh*), and the *World Bank* (*wo*). Finally, we add national websites in English: the *Australian Bureau of Statistics* (*ab*), *US National Center for Education Statistics* (*nc*), the *US Census* (*ce*), and the *US Bureau of Economic Analysis* (*be*). Websites are marked with a ✓ in the “Mlg.” column if they contain content in at least two languages, whether across different pages or within the same page. The websites range from a few thousand to over a million pages, excluding those resulting in 4xx or 5xx HTTP errors (see “#Available”, in thousands). A ✗ in “F.C.” (“Fully Crawled”) indicates incomplete crawls (details in Section 4.4.4).

The following metrics assess the *difficulty* and *interest* of a focused crawl. For partially crawled websites, metrics are computed on the BFS-visited subset (Section 4.4.4).

4.4. EXPERIMENTAL EVALUATION

Table 4.1: Main characteristics of websites (see detailed crawling methodology in Section 4.4.4)

Starting URL	Mlg.	F.C.	#Available	#Target	H2T	Target Size	Target Depth
<i>ab</i> https://www.abs.gov.au/	✗	✗	952.26	263.26	8.86	4.50 ($\pm$ 56.04)	8.94 ($\pm$ 2.56)
<i>as</i> https://www.assemblee-nationale.fr/	✗	✗	949.42	155.94	4.34	0.54 ($\pm$ 6.38)	5.84 ($\pm$ 1.07)
<i>be</i> https://www.bea.gov/	✗	✓	31.23	15.84	32.19	2.03 ($\pm$ 6.99)	5.73 ($\pm$ 3.21)
<i>ce</i> https://www.census.gov/	✗	✗	988.37	257.68	3.47	1.51 ($\pm$ 15.77)	4.23 ($\pm$ 0.48)
<i>cl</i> https://www.collectivites-locales.gouv.fr/	✗	✓	5.54	3.70	5.40	1.15 ($\pm$ 4.91)	2.80 ($\pm$ 0.82)
<i>cn</i> https://www.cnis.fr/	✗	✓	12.80	7.49	13.87	0.43 ($\pm$ 1.74)	4.26 ($\pm$ 1.59)
<i>ed</i> https://www.education.gouv.fr/	✗	✓	102.71	10.47	3.95	1.00 ($\pm$ 3.07)	11.89 ($\pm$ 13.22)
<i>il</i> https://www.ilo.org/	✓	✗	990.71	81.01	2.53	13.40 ($\pm$ 110.01)	4.26 ($\pm$ 1.28)
<i>in</i> https://www.interieur.gouv.fr/	✗	✓	922.46	22.98	1.54	1.12 ($\pm$ 3.06)	66.94 ($\pm$ 39.43)
<i>is</i> https://www.insee.fr/	✓	✓	285.55	168.88	41.34	3.13 ($\pm$ 21.43)	5.20 ($\pm$ 1.81)
<i>jp</i> https://www.soumu.go.jp/	✓	✗	993.87	328.83	6.30	0.80 ($\pm$ 4.49)	5.18 ($\pm$ 1.29)
<i>ju</i> https://www.justice.gouv.fr/	✗	✓	56.61	14.85	4.85	0.48 ($\pm$ 1.34)	86.91 ($\pm$ 86.30)
<i>nc</i> https://nces.ed.gov/	✗	✓	309.97	84.94	18.87	1.10 ($\pm$ 11.56)	3.63 ($\pm$ 1.66)
<i>oe</i> https://www.oecd.org/	✓	✓	222.58	45.04	15.61	2.31 ($\pm$ 23.37)	6.28 ($\pm$ 5.65)
<i>ok</i> https://okfn.org/	✓	✓	423.12	12.95	0.74	0.04 ($\pm$ 0.24)	2.64 ($\pm$ 2.89)
<i>qa</i> https://www.psa.gov.qa/	✓	✓	4.36	2.45	4.15	2.97 ($\pm$ 19.28)	3.03 ($\pm$ 0.61)
<i>uh</i> https://www.who.int/	✓	✗	351.86	55.59	14.19	1.26 ($\pm$ 11.14)	4.43 ($\pm$ 0.62)
<i>wo</i> https://www.worldbank.org/	✓	✗	223.67	23.10	2.38	2.80 ($\pm$ 27.16)	4.52 ($\pm$ 0.69)

“**#Target**” is the total number of identified targets in thousands, and the ratio $\frac{\text{\#Target}}{\text{\#Available}}$ measures *target density*; extremes are 66.78% (*cl*) and 2.49% (*in*). “**H2T**” (“HTML to Target”) is the percentage of HTML pages linking to at least one target, reflecting the *density of target-pointing pages* (e.g., *cl* has the highest density, but all targets are concentrated in 5.40% of HTML pages). “**Target Size**” gives the mean and standard deviation (s.d.) of target file sizes in megabytes, and “**Target Depth**” the mean and s.d. of depths, which correspond to the shortest link navigation from the root. *Mean depths vary greatly, from 3 to nearly 90*: highest values arise on portals requiring page-to-page navigation, e.g., *ju*. Table 4.1 shows our websites are *extremely diverse*: small or huge, varying in depth, target number and locations, and over 20 languages. This diversity demands *adaptable crawling strategies* to efficiently retrieve targets.

To further emphasize the diversity of crawling efforts, we manually analyzed the types of pages where targets are typically found. In *ju*, targets appear under various resource lists and data portals disseminated across the website. Some are huge (e.g., <https://www.justice.gouv.fr/documentation/bulletin-officiel>), others are smaller, e.g., <https://www.justice.gouv.fr/documentation/ressources/conventions-judiciaires-dinteret-public>. Some require multi-step navigation to reach deeper pages that contain one or a few target links. In *il*, most targets are found in small, dispersed data catalogs: some gathered in “*ILOSTAT*” while others are accessible only through further exploration. These targets are generally downloadable directly from the catalogs. In *wh*, the majority of targets require multi-step navigation through a hierarchy of subjects and sub-subjects, with only a subset leading to targets: efficient crawling thus requires quickly identifying interesting parts. In *nc*, all previously mentioned typologies are present: data portals (e.g., <https://nces.ed.gov/national-center-education-statistics-nces/products>), lists (e.g., <https://nces.ed.gov/surveys/spp/results.asp>), multi-step navigation.

4.4.2 Limitations of Search Engine-Based Dataset Retrieval

We attempted to retrieve SDs using search engines (SEs), in particular by testing three popular *Google* services: *classical search* (GS), *Google Datasets Search* (GDS), and the *Google Public Data Explorer* (GPDE) providing data from international organizations.

GS allows filtering by website (site:) and file type (filetype:), via its Web interface and *API*. However, SEs, including *Google*, do not fully index websites and GS caps results at 1 000, often truncating them. For instance, querying *ju* for PDFs yields only 302 results, although over 9 000 exist; similarly, only 240 ODS files are returned (out of 910), and on *in* 38 XLS (out of 1 546). Even worse: TSV is not recognized at all despite 11 097 files on *ju*. On *il*, GS lists 641 results instead of over 49 000, and no ZIP archives instead of at least 2 239. GDS trims results even further, e.g., 109 tabular files on *ju* (vs. 1 188); 93 datasets on *il* (vs. $\geq 170\,000$ found by our crawler); and 312 on *ce* (vs. 800 000), etc. GPDE is built for human readers, indexing from a closed, fixed set of providers, such as *Eurostat*, *wo*, *Statistics Canada*, and *oe*, but excludes key sites like UN’s *wh*, *il*, and US agencies (*be*, *nc*, *ce*). GPDE only allows manual exploration, plotting datasets guided by users’ filters but offering neither downloads nor links to originals: e.g., it displays 150

OECD statistics across 37 countries and 100 years, but only references a 272-page PDF (*OECD iLibrary*).

Overall, **SEs lack transparency and control over the retrieved SDs**, a critical limitation for our task. In contrast, our crawler retrieves all data files within a specified budget, with explainable decisions and explicit choice of target MIME types.

4.4.3 Baselines

We compare our best-performing SB-CLASSIFIER with an adapted version in which the URL classifier is replaced by an (unrealistic) perfect oracle (SB-ORACLE), as well as with several baseline methods. First, we consider four simple crawlers.

- (i) The RANDOM crawler selects the next hyperlink to visit uniformly at random from the crawl frontier.
- (ii) The *Breadth-First Search* (BFS) exhaustive crawler maintains the frontier as a *First-In-First-Out queue* data structure. It crawls all pages reachable by a path of length ℓ before any page reachable only by longer paths ($\ell' > \ell$).
- (iii) *Depth-First Search* (DFS) maintains the frontier in a *Last-In-First-Out stack*. It is rarely used for exhaustive crawling since it may fall into *robot traps*.
- (iv) The unrealistic *Omniscient Crawler* (OMNISCIENT) knows all target URLs (V^*) from the start and crawls them sequentially. Since an optimal crawler is intractable (Proposition 4), this omniscient crawler provides an (unreachable) upper bound on crawler efficiency.

The *Offline-Trained, Tag Path-Based Crawler* (TP-OFF) [63] is a closely related focused crawler originally designed to retrieve *diverse* textual content. We apply it *technically unchanged*, except for redefining its reward for target retrieval. As described in [63], it first crawls 3 000 pages with BFS and groups the tag paths leading to followed links as in Section 4.3.2. Page benefits are computed immediately, and tag path groups are stored in a priority queue ordered by average benefit. After these 3 000 pages, [63] *only considers links matching existing tag path groups* (ordered by priority), others are ignored. In our context, benefit computation requires knowing if links lead to targets. To enable this baseline, *we provide it with the true benefit* (number of targets behind a page’s links) as if given by an oracle, for the initial 3 000 pages. Beyond this point, newly formed tag path groups receive a fixed benefit of 0. This baseline, given an unfair advantage in its first stage, can be seen as an *ablated* version of ours, learning *offline* instead of *online* with our RL method.

The *Focused Crawler* (FOCUSED) represents early generic focused crawling approaches that rely on a classifier to estimate the likelihood that a newly discovered hyperlink leads to a target [32, 56]. The frontier is maintained as a *priority queue*, favoring links predicted to be relevant. It relies on a standard *logistic regression*, periodically retrained on crawled pages at no extra HTTP cost. Its features follow standard focused crawler practice: the (approximated) depth of the source page, a 2-gram BoW

representation of the URL (similarly to Section 4.3.3), and a 2-gram BoW representation of the link’s *anchor* text [32, 56]. *Topic-oriented* features are intentionally excluded (further discussed in Section 4.5). Overall, FOCUSED can be viewed as an *ablated* version of our crawler, as it neither exploits tag-path structure (Section 4.3.3) nor uses RL (Section 4.3.4).

TRES [121] is a topical, RL-based focused crawler using a Bi-LSTM classifier to assess HTML page relevance. Like our approach, it employs RL to learn where interesting content resides in a website. However, TRES defines interestingness through a user-specified topic, whereas we target SDs regardless of subject. As it targets topic-relevant HTML pages only, it requires input keywords, positive HTML examples, and is limited to one language. To include TRES as a baseline for SD retrieval, we made changes related to its focus, *without altering its core logic or technical components*. Specifically, we give it three *unfair advantages*:

- (i) TRES expects a list of topic-specific keywords to initialize its classifier’s training, hence we supply 74 hand-crafted terms likely to appear in links’ anchors to targets to initialize its classifier, which are listed below. A subset of these terms is language-dependent: namely, all entries that do not correspond to file extensions. We choose to express them in English, as a significant proportion of websites are written in English and, even when English is not the primary language (e.g., *jp* or *qa*), English content is frequently present. This design choice highlights the inherent language dependency of TRES.

pdf	archive	get dataset
xls	spreadsheet	data file
csv	table	read more
tar	list	resource list
zip	resource	get document
rar	annex	download publication
rdp	supplement	document archive
json	attachment	supporting materials
doc	proceedings	export data
xml	survey	dataset download
yaml	material	attached document
txt	output	official documents
tsv	content	browse files
ppt	statistics	download statistics
ods	article	download article
dta	paper	annual report
7z	metadata	white paper
tth	fact	technical report
file	download CSV	raw data
document	download PDF	metadata file
report	download XLS	open data
publication	download file	available for download
dataset	download document	technical documentation
data	access data	fact sheet
download	view report	

- (ii) We provide 1 000 HTML pages with links to targets from the evaluation websites as positive examples (based on prior crawls), as required to pre-train its classifier, giving it partial access to the solution.
- (iii) TRES must classify URLs as HTML or not to manage its frontier, as it only accepts HTML pages. Although costly in practice (see Section 4.3.3), we gave it an oracle doing this classification at no cost, giving it its chance while not affecting its original behavior.

Since its RL agent and classifier rely only on textual HTML features, TRES cannot directly detect targets. We therefore added two adjustments: links are added to the frontier as in [121], while others (which TRES would normally ignore) are visited immediately and targets counted if found; and its language filter was disabled, preventing coarse URL-based false exclusions. The modified code is available at [75].

4.4.4 Experimental Setup

Evaluating six baselines and our crawler under various hyperparameter settings with repeated crawls per website is infeasible due to time constraints (e.g., large files or slow connections) and crawling ethics. Thus, for each website, all baselines and our crawler run in a setting where *each first checks if the resource is already stored in a local database*. If so, we use it; otherwise, we fetch it via HTTP GET and the URL, HTTP status, headers, and response body are stored.

For evaluation purposes only, we stop crawling a website under any of the following conditions:

- (i) One crawler terminates before reaching one million pages, yielding a complete local replication: all others then rely solely on database look-ups.
- (ii) All crawlers reach one million visited webpages (possibly not the same pages across crawlers).
- (iii) A three-week time limit is reached, as crawling can be extremely slow for some websites (across 18 sites, 22.2 million distinct pages were retrieved).
- (iv) A crawler exceeds 1 minute between two requests, excluding network latency (only affecting TRES).

We conduct and present our experiments as follows:

- (i) We conduct hyperparameter studies and evaluate our URL classifier on fully-crawled websites with *fewer than one million pages*. Full replication enables parallel runs under multiple parameter settings and provides complete knowledge for both (unrealistic) SB-ORACLE crawler and TRES baseline (see Section 4.4.5).
- (ii) On websites where *all baselines reach one million webpages*, crawlers are compared on the subset of the website each of them visited. Note that their target sets, and

the retrieved data volume, may differ: a good crawler prioritizes *more target-rich* pages.

- (iii) On websites where *at least one baseline fails to reach one million pages* within the time limit, crawlers are compared on the smallest crawl size achieved. For example, if three different crawlers visit respectively 400 000, 200 000, and 800 000 pages, then results are computed on their (respective) first 200 000 pages.

We exclude retrieval times from our results, since these vary out of our control. We focus on the *number of requests and data volumes*; note that crawl time can be estimated from these, knowing the bandwidth and the ethics waiting time. To illustrate retrieval times, on the medium-sized website *ed*, SB-CLASSIFIER requires 3h 16min to collect 5 000 targets and 10h 52min to collect 10 000, whereas BFS requires 5h 13min and 48h 45min respectively ($1.6\times$ and $4.5\times$ more).

4.4.5 Performance Comparison with Baselines

Figures 4.5 and 4.6 compare our crawler’s performance with the baselines on the 18 diverse websites from Table 4.1. Default settings are $n = 2$ (n -grams), $\theta = 0.75$, $\alpha = 2\sqrt{2}$, $m = 12$ (path vector dimension), $w = 15$ (hash function), and $b = 10$ (batch size). MIME type and extension blocklists are chosen to filter out multimedia content (image, audio, video), usually large in volume. We thus exclude `image/*`, `audio/*`, and `video/*` MIME types, as well as the following URL extensions:

.3g2	.cgm	.flac	.jpg	.midi	.nef	.qt	.viv
.3ga	.cmx	.fli	.jpgm	.mj2	.npx	.ra	.wav
.3gp2	.cr2	.flv	.jpgv	.mjp2	.oga	.raf	.wax
.3gp	.crw	.fpx	.jpm	.mka	.ogg	.ram	.wbmp
.3gpa	.dcr	.fst	.k25	.mkv	.ogv	.ras	.weba
.3gpp2	.djv	.fvt	.kar	.mmr	.orf	.raw	.webm
.3gpp	.djvu	.g3	.kdc	.mov	.pbm	.rgb	.webp
.aac	.dng	.gif	.lvp	.movie	.pct	.rlc	.wm
.aacp	.dts	.h261	.m1v	.mp2	.pcx	.rmi	.wma
.adp	.dtshd	.h263	.m2a	.mp2a	.pef	.rmp	.wmv
.aff	.dwg	.h264	.m2v	.mp3	.pgm	.rw2	.wmx
.aif	.dxf	.heic	.m3a	.mp4	.pic	.rwl	.wvx
.aiff	.eol	.heif	.m3u	.mp4v	.opus	.snd	.x3f
.arw	.erf	.icns	.m4a	.mpa	.pjpg	.spx	.xbm
.asf	.f4v	.ico	.m4b	.mpe	.png	.sr2	.xif
.asx	.fbs	.ief	.m4p	.mpeg	.pnm	.srf	.xpm
.avi	.fh4	.jfi	.m4r	.mpg4	.ppm	.svg	.xwd
.avif	.fh5	.jfif	.m4u	.mpg	.psd	.svgz	
.avifs	.fh7	.jif	.m4v	.mpga	.ptx	.tif	
.bmp	.fh	.jpe	.mdi	.mrw	.pya	.tiff	
.btif	.fhc	.jpeg	.mid	.mxu	.pyv	.ts	
.ecelp4800	.ecelp7470	.ecelp9600	.jfif-tbnl				

4.4. EXPERIMENTAL EVALUATION

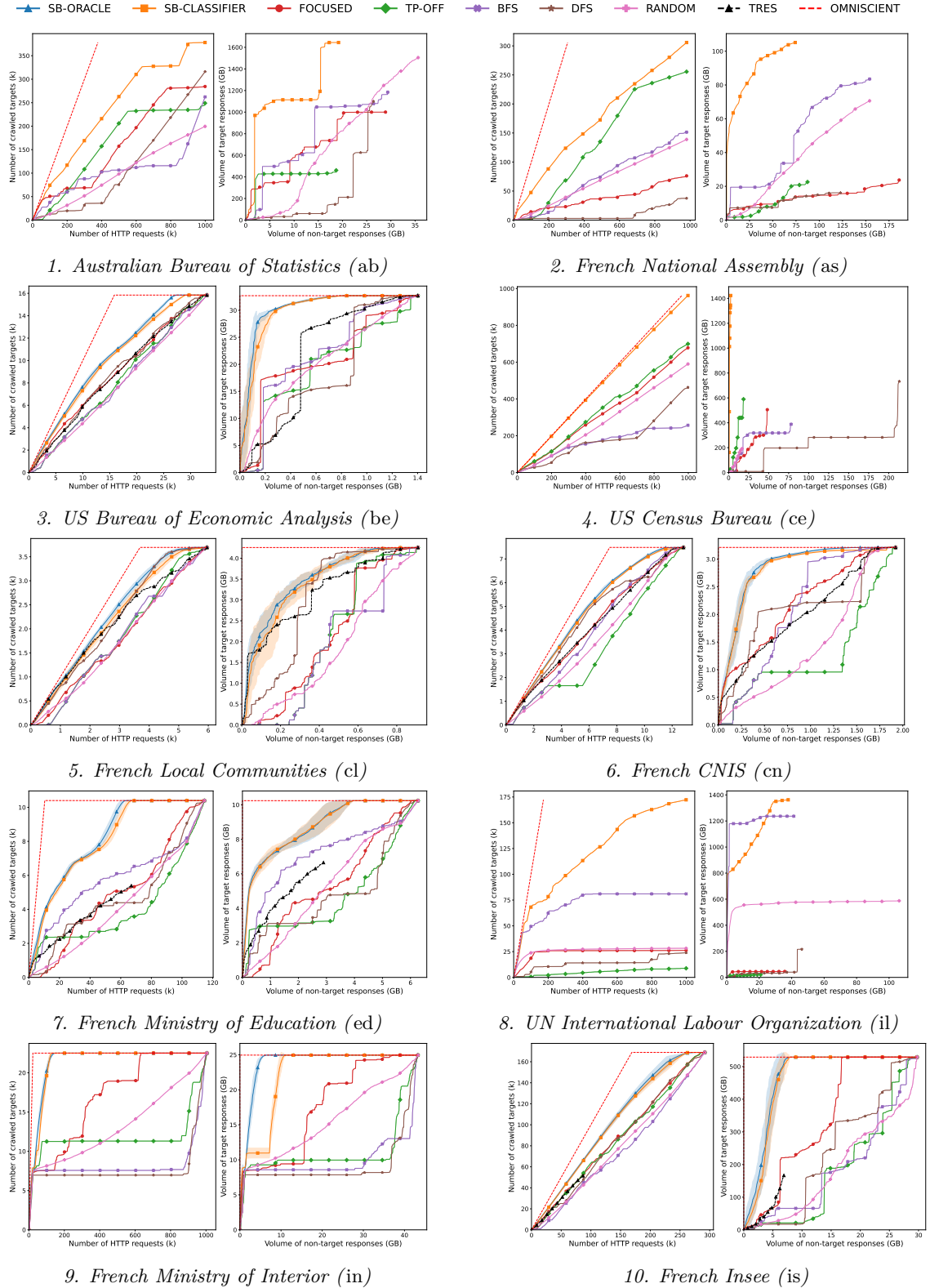


Figure 4.5: Comparison of different crawlers' performance for 10 of the websites presented in Table 4.1; for TRES, experiments are only shown for fully-crawled websites. SB-CLASSIFIER is our proposed approach.

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

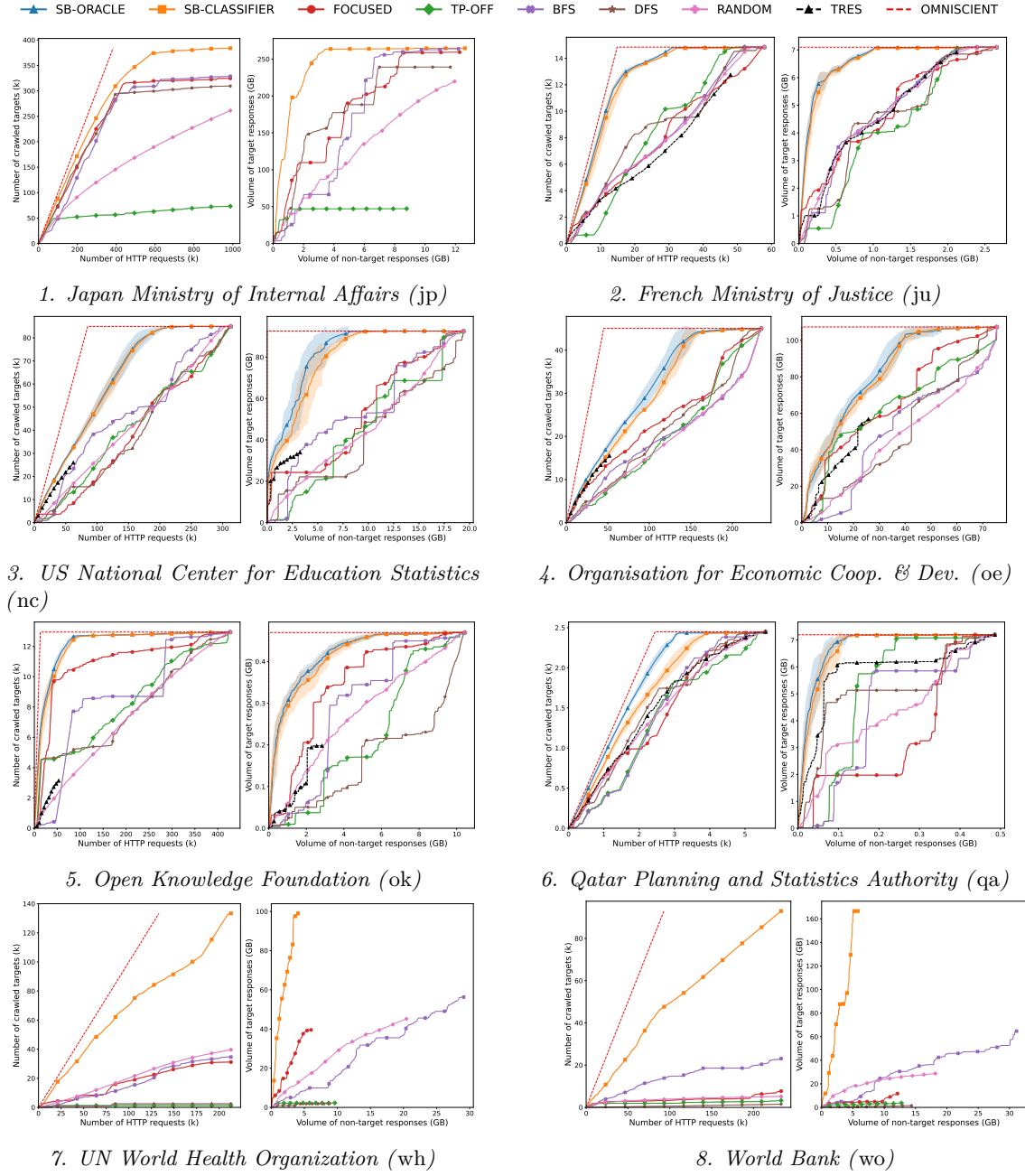


Figure 4.6: Comparison of different crawlers' performance for the 8 other websites (not in Figure 4.5); for TRES, experiments are only shown for fully-crawled websites

Table 4.2: Percentage of requests each crawler needs to retrieve 90% of the targets, for websites in Table 4.1. Below double horizontal rule, percentage of saved requests and lost targets due to the early-stopping mechanism (see Section 4.4.10).

Crawler	Website															
	ab	as	be	ce	cl	cn	ed	il	in	is	jip	ju	nc	oe	ok	qa
SB-ORACLE	NA	NA	72.6	NA	70.7	70.3	48.0	NA	12.8	73.8	NA	34.1	50.8	55.8	13.8	47.3
SB-CLASS.	31.2	35.1	75.7	23.5	74.4	70.9	51.5	14.2	11.9	76.0	37.7	35.8	51.6	59.2	15.5	57.7
FOCUSED	68.2	+∞	87.8	36.0	88.9	82.7	86.7	+∞	62.8	86.9	42.0	91.1	92.8	84.9	51.8	71.0
TP-OFF	96.4	50.3	86.2	34.7	81.8	88.2	95.6	+∞	99.7	88.0	+∞	74.4	93.0	88.7	76.2	88.6
BFS	97.4	90.8	89.1	73.5	87.5	80.0	94.6	33.2	99.3	92.7	45.2	80.8	81.8	96.5	66.8	70.6
DFS	83.7	+∞	85.2	74.9	70.6	84.6	90.5	+∞	99.7	87.7	45.6	80.2	93.7	88.7	80.5	74.4
RANDOM	+∞	98.2	92.4	44.5	89.2	85.1	95.0	+∞	99.0	92.7	+∞	83.2	87.9	96.8	85.0	77.8
Saved req.	34.4	0.0	0.0	0.0	0.0	0.0	27.4	0.0	82.6	2.2	39.0	18.8	20.4	0.0	73.1	0.0
Lost targets	13.5	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	2.5	0.4	0.1	0.0	2.0	0.0

Table 4.3: Fraction of non-target volume retrieved before reaching 90% of the total target volume, for websites in Table 4.1.

Crawler	Website															
	ab	as	be	ce	cl	cn	ed	il	in	is	jip	ju	nc	oe	ok	qa
SB-ORACLE	NA	NA	24.2	NA	56.3	24.6	49.2	NA	12.5	59.7	NA	22.9	29.5	48.0	33.2	30.2
SB-CLASS.	20.4	21.4	29.5	29.1	56.0	29.0	49.2	53.2	23.6	64.7	18.6	23.1	34.5	49.5	34.9	33.2
FOCUSED	+∞	+∞	85.2	97.0	76.3	74.7	86.4	+∞	67.3	73.8	66.8	72.2	84.9	72.7	49.8	80.3
TP-OFF	+∞	+∞	92.3	64.4	65.0	94.7	92.9	+∞	98.8	89.7	+∞	72.3	89.2	89.0	73.6	46.9
BFS	81.8	75.7	66.5	98.5	80.8	50.4	93.2	3.6	99.0	93.8	54.0	68.0	84.5	97.5	63.3	87.3
DFS	98.6	+∞	64.2	97.0	45.0	82.4	90.8	+∞	98.1	85.0	59.5	68.6	96.1	90.5	97.0	75.0
RANDOM	71.6	+∞	83.4	+∞	89.3	82.7	92.9	+∞	95.8	98.3	+∞	70.1	88.2	98.1	86.6	77.8

Note that URL extension filters are not useful on all websites (on some, URLs do not include any extension, see discussion at the beginning of Section 4.3.3). The impact of this filtering mechanism is discussed in Section 4.4.9.

In addition, for each crawler and all 18 websites, Table 4.2 (above the double rule) shows *the percentage of requests needed to retrieve 90% of targets*, and Table 4.3 *the percentage of non-target page volume retrieved before reaching 90% of total target volume*. Lower values indicate greater efficiency. For partially crawled sites, metrics are computed on pages visited by BFS. If a crawler never reaches the 90%, we show $+\infty$. The best-performing crawler per website is in bold, excluding SB-ORACLE (as unrealistic). For each website in Figure 4.5, two graphs show crawler performance: on the left, the *number of crawled targets vs. number of HTTP requests* (both GET and HEAD, in thousands); on the right, the *volume of target responses (GB) vs. volume of non-target ones*. In both graphs, a higher curve is better.

For *fully-crawled websites*, we run two variants of our crawler: SB-CLASSIFIER and SB-ORACLE (see Section 4.4.3). This assesses the impact of the URL classifier on crawl performance. Results are *averaged over 15 runs*, with shaded areas showing $\pm$ one s.d.; these are in most cases low, indicating stable behavior. Variability arises mainly from randomness in link selection within chosen actions, the stochastic SGD-trained classifier, and the single-state RL agent, whose design is crucial for stabilization (further discussed in Section 4.5).

For *partially crawled websites*, we report a single run of SB-CLASSIFIER, lacking perfect classifier information, and since multiple crawls are unfeasible (Section 4.4.4). All baselines but RANDOM and TRES are deterministic, thus one run suffices. TRES is only evaluated on the 10 smallest fully-crawled websites, as it requires oracle access and does not scale: beyond small websites (*be*, *cl*, *cn*, and *qa*), it exceeds the 1-minute-per-request threshold and is stopped before completing the crawl of larger websites. Crawling 100 000 additional pages at that point would require at the very least ten more weeks.

The plots and Tables 4.2 and 4.3 show that **our crawler with the URL classifier outperforms all baselines**, with one exception: on *cl*, DFS is slightly better near the end. On *il*, on volume only, BFS briefly surpasses ours, but overall our crawler retrieves twice as many targets. **For websites *as*, *ce*, *ed*, *il*, *in*, *ju*, *nc*, *wh*, and *wo*, SB-CLASSIFIER significantly reduces resource usage** (time or bandwidth) to reach a fixed number of targets or maximize targets under a budget. On *wo*, the best baseline (BFS) would require three months to match our targets (linear extrapolation), and FOCUSED nine months. On *wh*, the best baseline (RANDOM) would require three months, and the worst, (TP-OFF), six years. On *il*, the top-3 baselines find no targets in the last 650 000 iterations, whereas ours continues until the end. On other sites (*be*, *cn*, *cl*, *jp*, *qa*), while some baselines approach SB-CLASSIFIER performance, it remains superior overall, outperforming all baselines on a wide variety of websites, with respect to the size, depth, target distribution, etc. Even in less favorable scenarios where targets are widely distributed within the website, our approach is still beneficial. For instance, on *ab* and *as* there is on average one target for every two non-target resources, while on *jp* nearly one target exists per non-target; even in the extreme case of *ce*, where almost all resources

are targets, our approach still outperforms all baselines. This shows that our method not only adapts to varying target densities but also maintains robust performance on both extremely sparse and extremely dense target distributions. Small sites (*be*, *cl*, *cn*, *qa*) are traversed quickly, allowing less learning, yet our approach adapts effectively.

TP-OFF performs poorly, especially on *ed*, *il*, *jp*, *wh*, and *wo*. Learning from just 3 000 pages proves insufficient on large websites. On *cn*, it underperforms BFS after the initial 3 000 pages, despite only three times more pages remaining. Increasing the training set could help but would approach the simplicity of a standard BFS crawler.

FOCUSED is outperformed by ours on all websites, showing early focused crawlers are too simplistic to solve our problem efficiently. On 10 websites out of 18, it is beaten by BFS, DFS or RANDOM.

Despite being given three unfair advantages (hand-picked keywords, access to pages with target links, and an oracle for URL type prediction), TRES fails on 9 out of 10 websites, performing reasonably on *oe* alone (and only until it had to be stopped). More critically, TRES cannot scale to medium or large sites, even when run locally. Crawl iterations slow dramatically due to exhaustive feature-based evaluations during tree expansion. These results confirm that topical crawlers are fundamentally ill-suited for direct target retrieval, even when heavily adapted and unfairly favored.

Comparing our SB-CLASSIFIER with SB-ORACLE shows that our classifier is close to the (virtual) perfect oracle, which is validated in Section 4.4.9.

4.4.6 Hyperparameter Study

We present the impact of three key hyperparameters (α , n , and θ) on crawl performance. Table 4.4 reports the number of requests (resp. response volume for non-target pages) needed to retrieve 90% of the targets (resp. target volume). Figures 4.7 to 4.12 present exhaustive graphical results for α , n , and θ on the 11 fully-crawled websites. When varying one parameter, others remain at default values (Section 4.4.5). These hyperparameter studies are conducted without enabling the multimedia filtering component, in order to isolate the effect of each parameter on crawl performance. Additional parameters, including the projection dimension D (Section 4.3.2), showed no significant effect.

1. *Impact of α .* α controls the exploration–exploitation trade-off (Section 4.3.2). We test $\alpha \in \{0.1, 2\sqrt{2}, 30\}$ (Figures 4.7 and 4.8, top of Table 4.4), as optimality of $2\sqrt{2}$ is not guaranteed in our non-standard MAB setting. Larger α generally hurts performance; within the tested range, $\alpha = 2\sqrt{2}$ offers the best balance. Large α leads to excessive *exploration*, especially on low or moderate-reward websites, neglecting useful, already discovered actions. Examining the plots, $\alpha = 0.1$ occasionally outperforms $\alpha = 2\sqrt{2}$ (e.g., *oe*, *ok*) or achieves similar results (e.g., *cl*, *ed*, *in*, *is*). However, it consistently produces higher variability, visible in larger s.d. areas, both for cases with equivalent performance (e.g., *cl*, *ed*) and for those where it performs worse (e.g., *be*, *cn*, *ju*). Consequently, besides being generally less reliable than $\alpha = 2\sqrt{2}$, $\alpha = 0.1$ reduces crawler stability, which is critical since crawling is costly and multiple runs to secure a “good” outcome are not practical.

2. *Impact of n in n -grams for tag path representation.* For merging tag paths (Section 4.3.1), we test $n \in \{1, 2, 3\}$, where $n = 1$ represents paths as sets of HTML tags. Table 4.4 (middle) and Figures 4.9 and 4.10 show that $n = 2$ and $n = 3$ generally outperform $n = 1$ with the only exceptions being *oe* in the plots and, to a lesser extent, *in*, confirming that representing groups of tag paths provides a better learning basis than sets of individual tags. We select $n = 2$ because, although $n = 3$ yields similar performance ($n = 2$ performs better than $n = 3$ in Table 4.4 eight times, while $n = 3$ is better nine times), the plots give a more complete picture, showing that overall performance is generally equivalent, particularly in terms of number of requests, where the curves are almost always overlapping except on *oe*, *ok*, and *qa*. Moreover, choosing $n = 3$ incurs a higher computational cost due to exponential growth of the feature space.
3. *Impact of θ .* The similarity threshold θ controls tag path grouping (Section 4.3.1). We test $\theta \in \{0.55, 0.75, 0.95\}$ (Figures 4.11 and 4.12, and bottom of Table 4.4). High θ often performs poorly, especially when no similarity above 95% exists, as with websites that add unique IDs to tags. For instance, on *ed*, it caused an *Out-of-Memory* (OOM) error by creating as many actions as HTML pages. The best results occur with $\theta \in \{0.55, 0.75\}$, with $\theta = 0.75$ usually superior. When 0.55 performs better, 0.75 remains comparable (except occasionally for *oe* in the plots), but not conversely (e.g., on *ju*, $\theta = 0.75$ improves both requests and volume by 40%). Moreover, as with α , smaller values of θ generally increase variability, visible in larger s.d. areas. This is especially critical for websites *be*, *cl*, *ed*, and *ju*, further supporting the choice of $\theta = 0.75$.

4.4.7 Effectiveness of Sleeping Bandit Learning

We first study the reward associated by our algorithm to tag paths. Figure 4.14 shows the mean reward of the top ten path groups (logarithmic y -axis), for the ten websites in Figure 4.5. Top groups have high rewards: across websites, the best group averages 258, followed by 89, 74, 67, up to 41 for the 10th, showing that **our SB agent effectively identifies tag path groups leading to HTML pages with target links**. Although the first often dominates, several groups still yield substantial reward, hence restricting to the top one is insufficient. Comparing Figure 4.14 with the mean rewards per website (groups with non-zero rewards) in Table 4.5, the top 10 groups far exceed the overall mean (e.g., for *wo*, the 10th group scores 33.5 vs. 2.1 mean). This is also clear from the high s.d. values from Table 4.5 (over 20 times the mean for *wo*) indicating large differences across groups: rewards are not normally distributed but more closely resemble a power law. Table 4.5 also shows large cross-website reward disparities, preventing an optimal exploration–exploitation coefficient α . This supports our pragmatic choice of $\alpha = 2\sqrt{2}$, effective in practice, and further highlights strong heterogeneity in target concentration across websites, complementing Section 4.4.1.

Examining top-group tag paths often reveals clues about target-rich areas. In *nc*, a typical path includes `div.container.w-iap/div.iap-content`, linked to the *Inter-*

4.4. EXPERIMENTAL EVALUATION

Table 4.4: Percentage of requests that SB-ORACLE performs to retrieve 90% of the targets (left of |). Right of |: percentage of the volume of non-target pages retrieved, before retrieving 90% of the total target volume. Hyperparameter study on α (top), n (in n -grams, center), and θ (bottom), for the 11 fully-crawled websites.

Crawler	be	cl	cn	ed	in	is	ju	nc	oe	ok	qa											
$\alpha = 0.1$	86.3	26.2	75.9	42.3	74.3	35.5	53.7	54.1	9.8	10.2	77.1	66.2	37.1	35.0	51.6	26.2	55.6	34.4	14.3	33.2	67.7	32.1
$\alpha = 2\sqrt{2}$	84.7	24.2	76.4	56.3	71.8	24.6	53.0	49.2	11.1	11.0	74.2	58.9	35.0	22.9	51.4	29.5	59.2	48.0	10.3	19.0	68.9	33.9
$\alpha = 30$	83.8	36.7	79.6	58.9	75.3	32.4	66.2	41.5	11.6	11.8	80.9	66.4	43.3	28.8	67.3	29.5	68.8	72.9	36.7	71.3	71.8	30.4
$n = 1$	84.5	27.1	77.2	48.5	78.6	56.3	57.3	55.1	9.9	10.7	78.2	69.6	35.7	17.6	54.8	33.5	52.6	28.1	13.6	27.2	68.9	34.7
$n = 2$	84.7	24.2	76.4	56.3	71.8	24.6	53.0	49.2	11.1	11.0	74.2	58.9	35.0	22.9	51.4	29.5	59.2	48.0	10.3	19.0	68.9	33.9
$n = 3$	84.1	32.8	78.2	51.2	71.3	25.7	57.0	53.1	10.7	10.5	71.3	49.2	37.0	26.9	51.2	27.0	79.6	79.0	6.0	8.8	70.0	34.9
$\theta = 0.55$	81.2	42.0	76.8	50.5	76.6	41.9	56.5	53.1	8.2	9.4	78.7	65.5	80.6	65.4	56.1	35.5	52.4	30.9	12.5	25.7	67.8	26.0
$\theta = 0.75$	84.7	24.2	76.4	56.3	71.8	24.6	53.0	49.2	11.1	11.0	74.2	58.9	35.0	22.9	51.4	29.5	59.2	48.0	10.3	19.0	68.9	33.9
$\theta = 0.95$	82.4	47.7	84.3	72.1	73.1	44.7	OOM	OOM	9.8	11.0	71.0	54.9	73.3	66.5	57.3	33.2	90.2	87.2	12.4	19.0	68.3	25.9

Table 4.5: Mean and s.d. of non-zero rewards of the learning agent on each website

	ab	as	be	ce	cl	cn	ed	il	in	is	jp	ju	nc	oe	ok	qa	wh	wo
Mean	1.7	1.5	4.5	30.2	12.4	4.2	2.5	3.1	1.6	3.5	3.5	5.4	2.0	2.5	5.5	15.4	3.0	2.1
s.d.	16.8	5.35	20.9	290.3	2.8	8.9	7.1	53.9	4.2	11.1	17.4	10.5	8.7	9.3	13.9	18.8	22.0	43.5

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

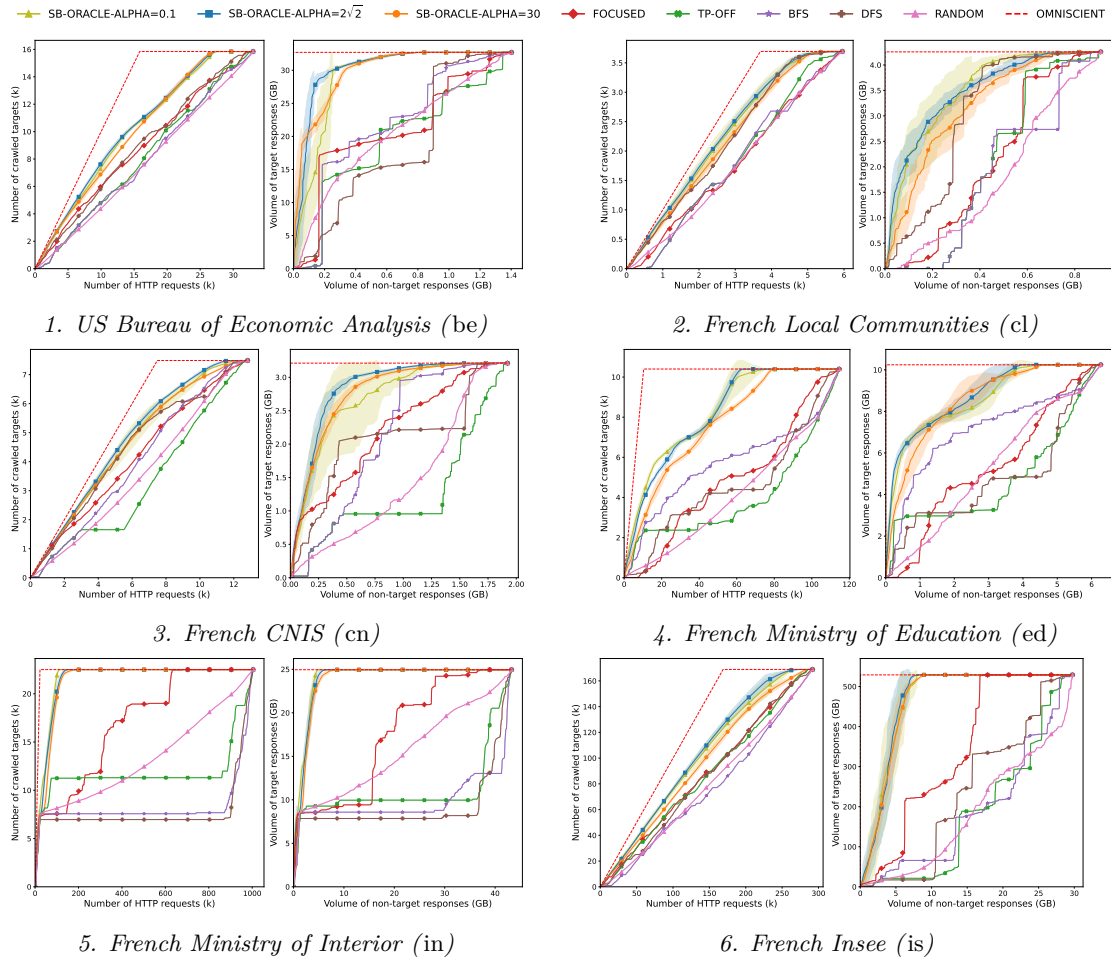


Figure 4.7: Crawler performance for hyperparameter study on coefficient α , for websites be, cl, cn, ed, in, and is

4.4. EXPERIMENTAL EVALUATION

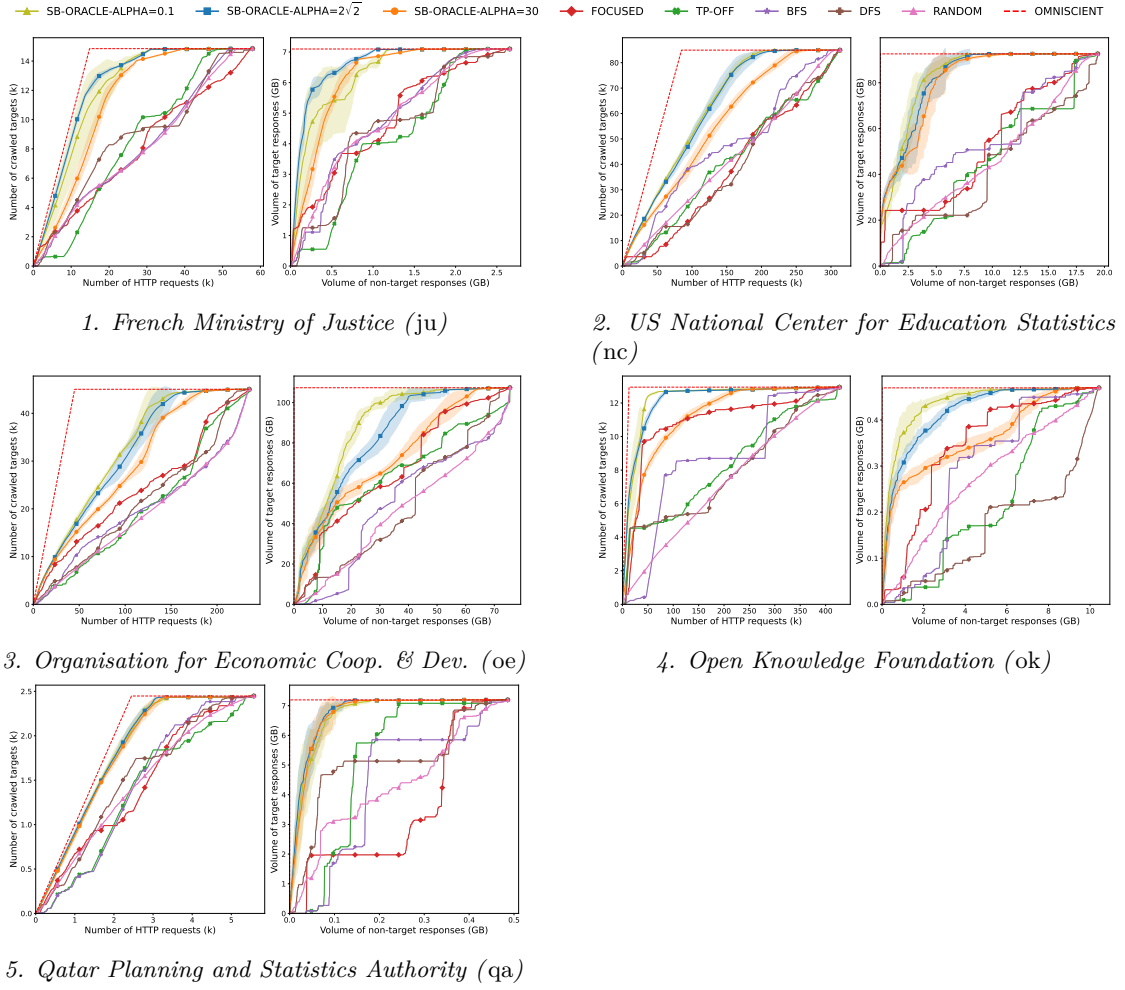


Figure 4.8: Crawler performance for hyperparameter study on coefficient α , for websites ju, nc, oe, ok, and qa

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

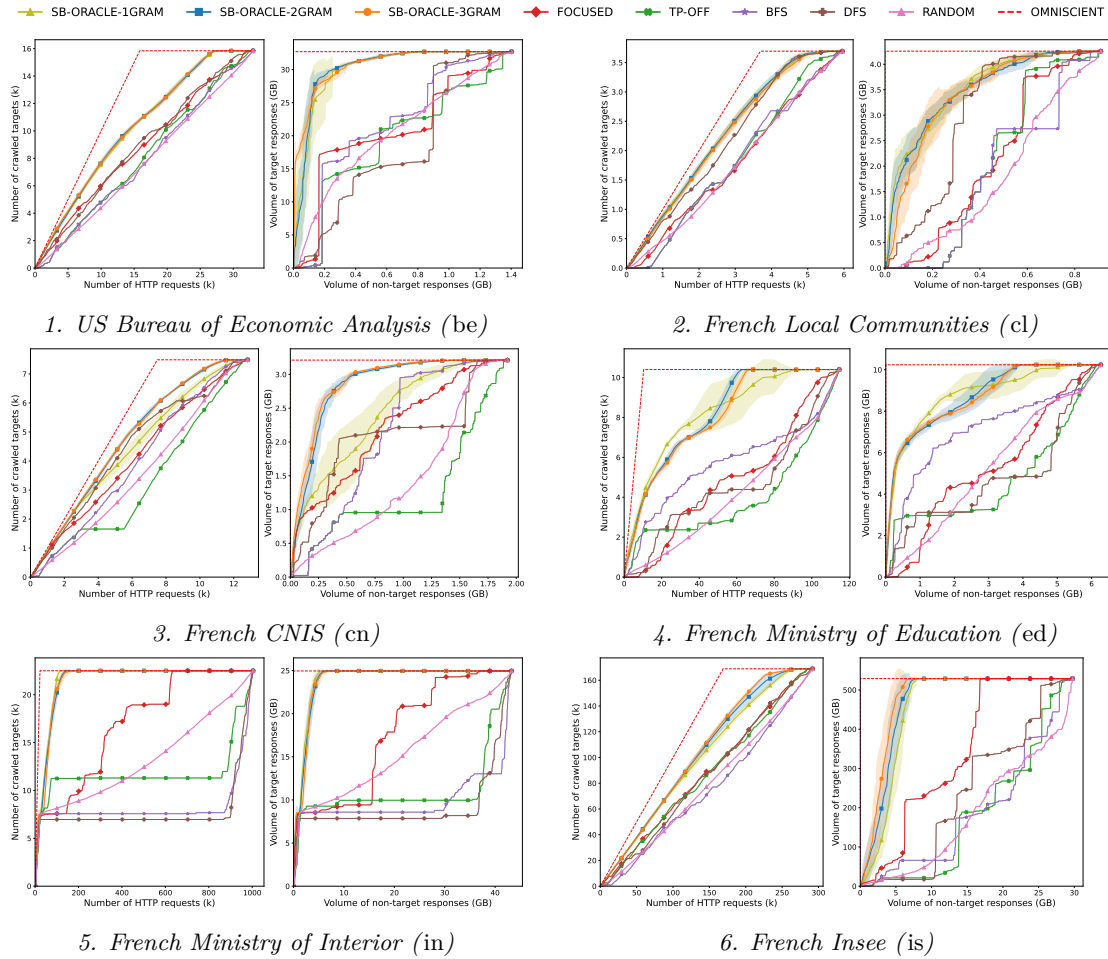


Figure 4.9: Crawler performance for hyperparameter study of the choice of n in n -grams used in tag path vector representation, for websites be, cl, cn, ed, in, and is

4.4. EXPERIMENTAL EVALUATION

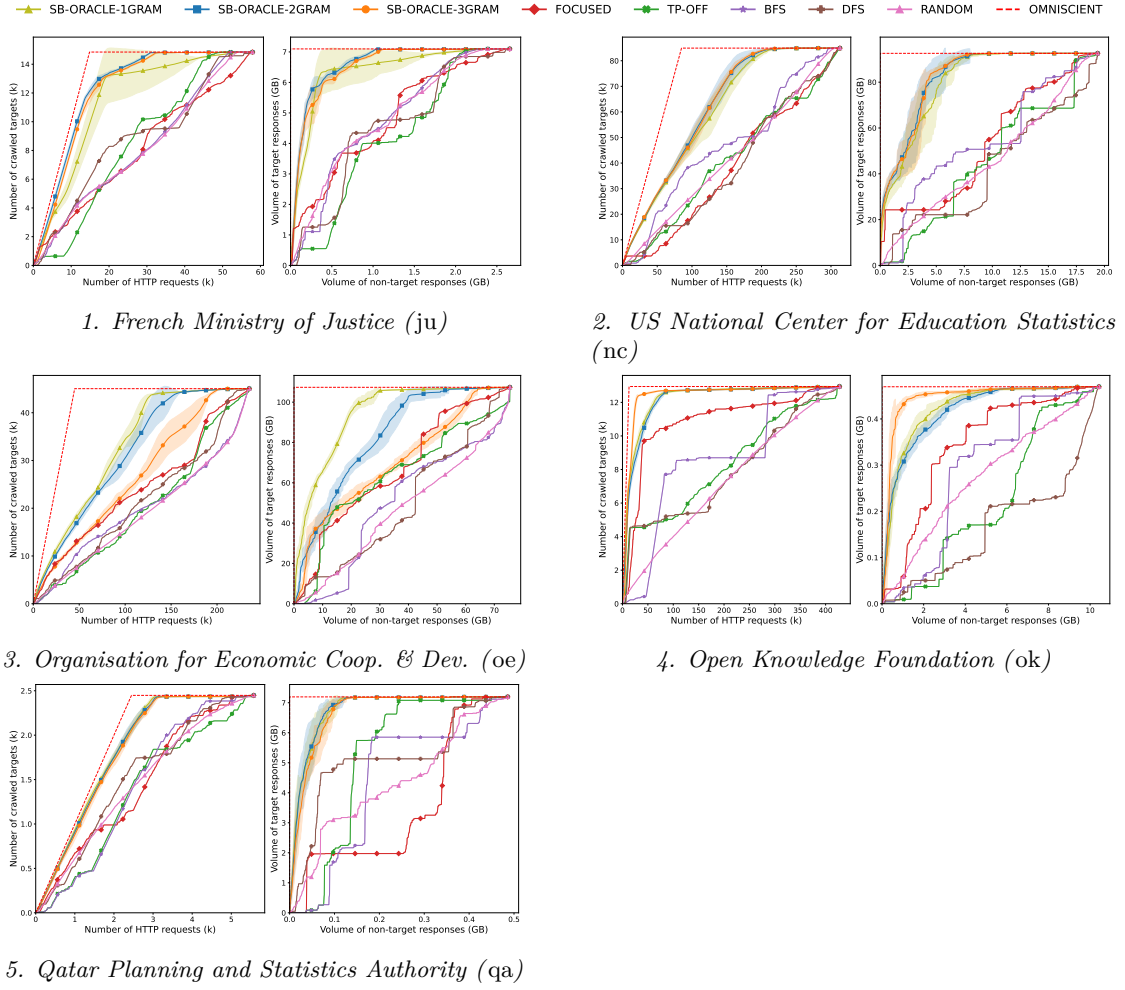


Figure 4.10: Crawler performance for hyperparameter study of the choice of n in n -grams used in tag path vector representation, for websites ju, nc, oe, ok, and qa

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

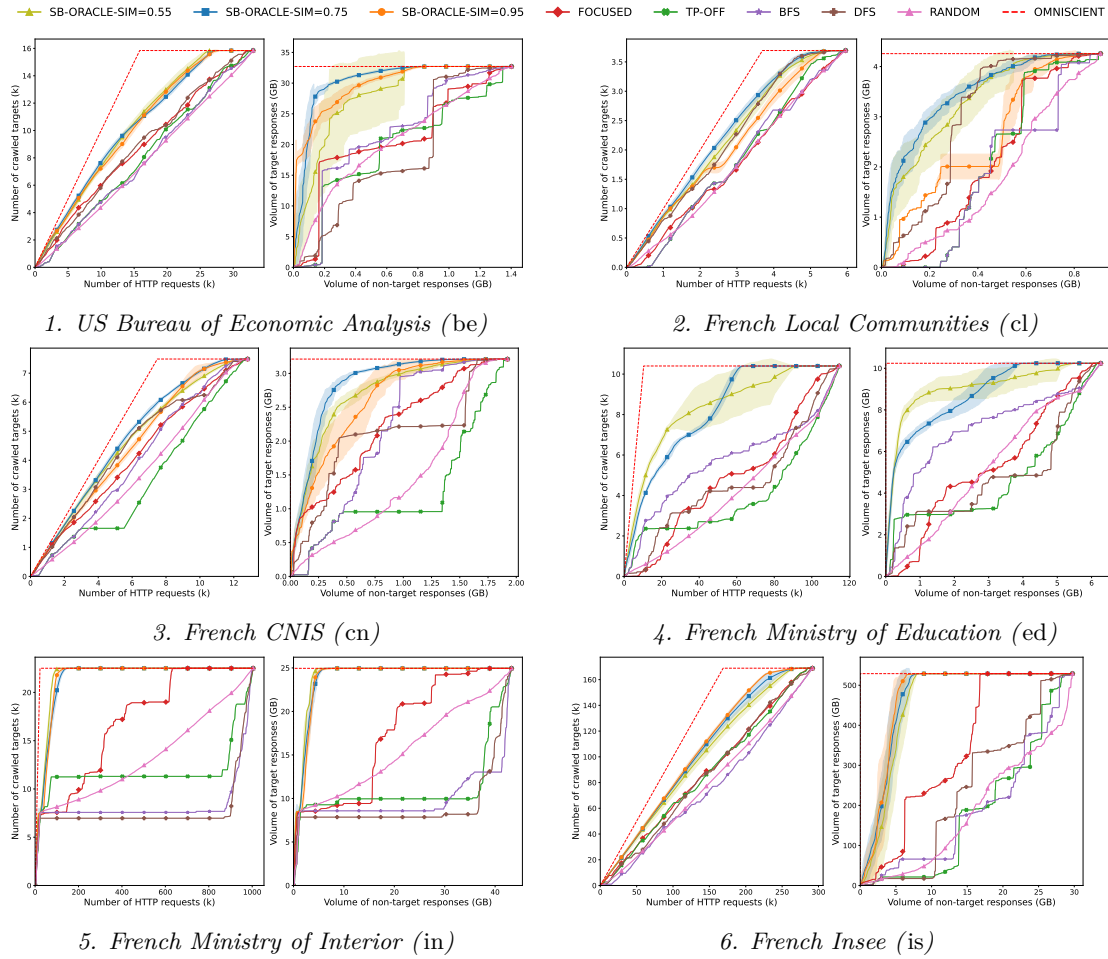


Figure 4.11: Crawler performance for hyperparameter study on similarity threshold θ , for sites be, cl, cn, ed, in, and is

4.4. EXPERIMENTAL EVALUATION

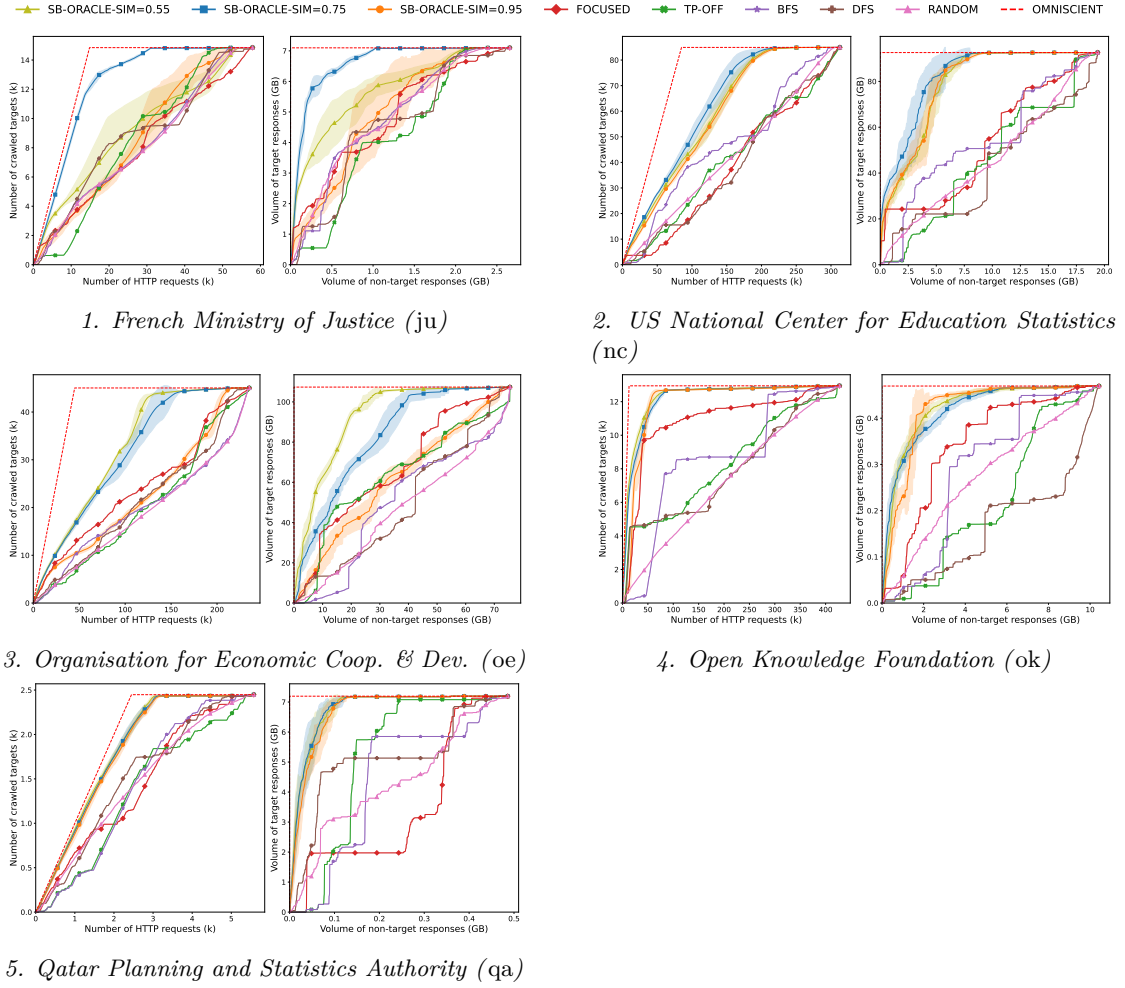


Figure 4.12: Crawler performance for hyperparameter study on similarity threshold θ , for websites ju, nc, oe, ok, and qa

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

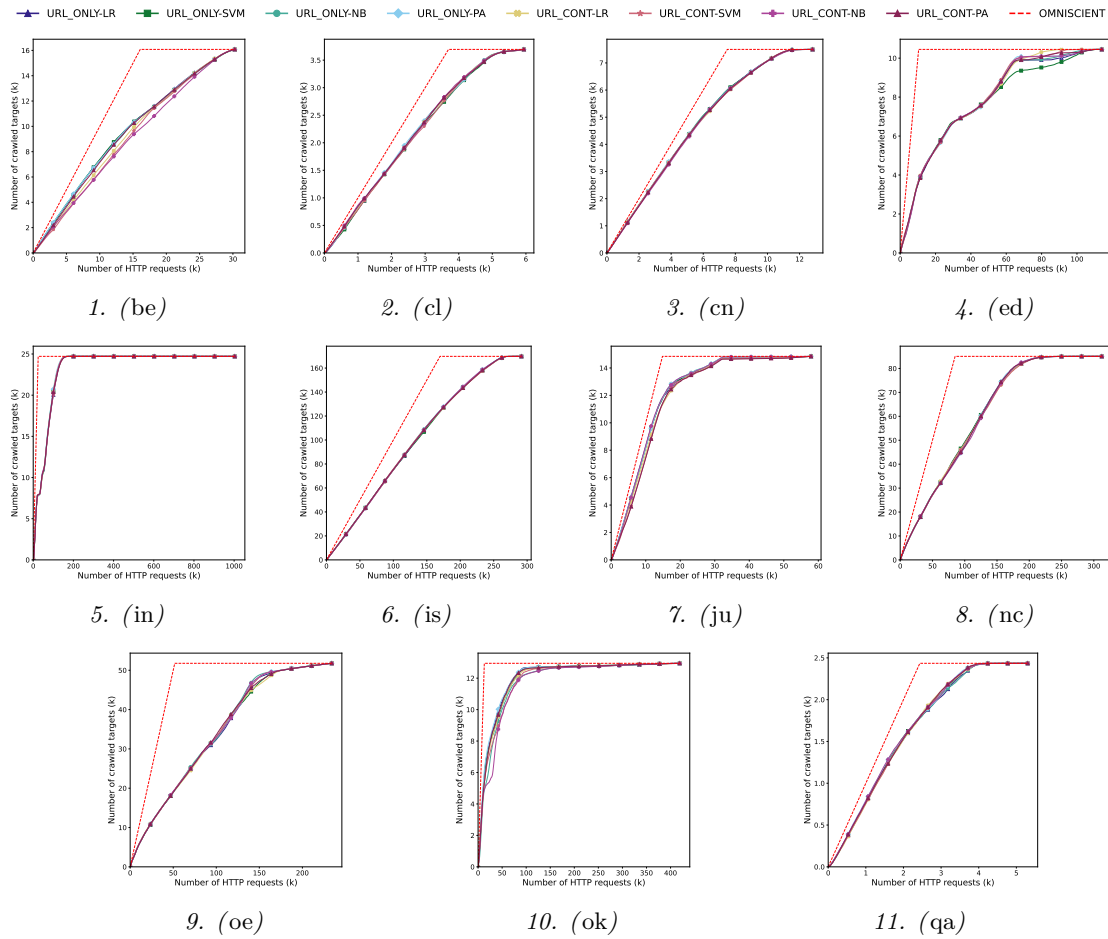


Figure 4.13: Crawler performance on fully-crawled websites for the different URL classifier models and feature sets

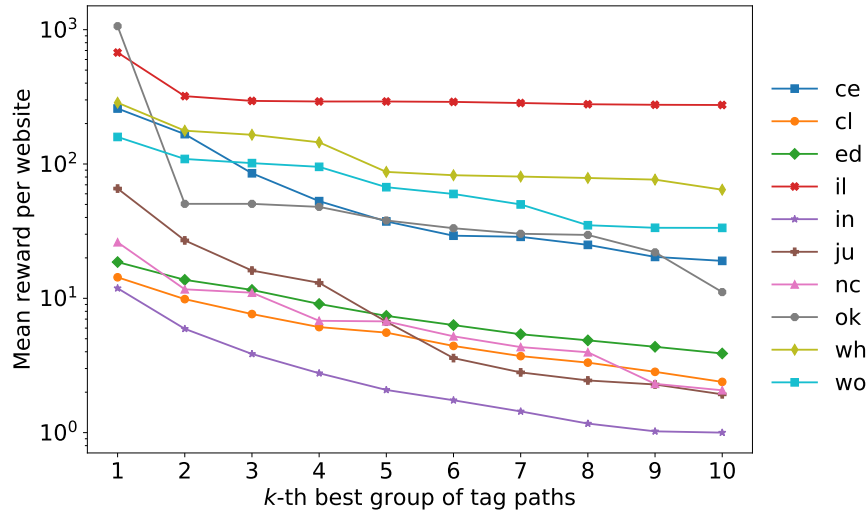


Figure 4.14: Mean rewards of the top-10 groups of tag paths for the ten selected websites from Figure 4.5 (log y-scale)

national Activities Program, a program focused on collecting international education statistics:

```
/html/body/div.nces/div.container.w-iap/div.iap-content/div.row/div.col-md-6/h4/a
```

In *wo*, paths include *collections-sief*, related to the *Strategic Impact Evaluation Fund*, a trust fund producing data and reports:

```
/html/body/div.wp-page-body.container.default-wrapperpage-collections.collections-sief/div.body-content-wrap.theme-nada-2/div.about-collection/div.repository-container/div.body/div/p/a
```

Some groups directly include target-retrieval keywords, e.g., *fr-link-download* in *ju*:

```
/html/body.path-node.page-node-type-ressource/div.dialog-off-canvas-main-canvas/div.layout-container/main#main-content/div.layout-content/div.region.region-content/div.block.block-system.block-system-main-block#block-open-theme-contentudelapageprincipale/article.mj-ressource.mj-break-word.node--promoted/div.fr-container.mj-ressource__content/div.fr-grid-row.fr-grid-row--gutters.mj-content__corps/div.fr-col-lg-8.fr-col-offset-lg-2.fr-col-12.content-container__paragraph/section.fr-downloads-group.fr-downloads-group--multiple-links/ul/li/a.fr-linkfr-link--download
```

or *status-publish* in *ok*:

```
/html/body.home.page-template.page-template-onecolumn-page.page-template-onecolumn-page-php.page.page-id-7/main.content/div.container/div.row/div.maincolumn-md-12/article.post.post-7.page.type-page.status-publish.hentry#post-7/div.entry-content/div#stcpDiv/div/strong/a
```


Other websites’ top paths are not human-interpretable yet still achieve high rewards. On *jp*:

```
/html/body.top/div#wrapper/div#groval_navi/ul#groval_menu/li.menu-item-has-children/ul.sub-menu/li/a
```

On *il*:

```
/html/body/div.container.s-lib-side-borders.pad-top-med#s-lg-side-nav-content/div.row/div.col-md-9/div.s-lg-tab-content/div.tab-pane.active#s-lg-guide-main/div.row.s-lg-row/div.col-md-12#s-lg-col-1/div.s-lg-col-boxes/div.s-lg-box-wrapper-17054143#s-lg-box-wrapper-17054143/div.s-lib-box-container#s-lg-box-14451639-container/div.s-lib-box.s-lib-box-std#s-lg-box-14451639/div#s-lg-box-collapse-14451639/div.s-lib-box-content/div.clearfix#s-lg-content-31285074/ul/li/a
```

On *ed*:

```
/html/body.cf-rtm/div.accueil-portail#page/div.container#main-ermes-container/main#main/div#readspeaker-container/div#portal/div.row/div.col-md-12.cms-inner-layout#layout-2/div.row/div.col-md-6.cms-inner-zone#zone-4/div#frame-224/div.frame.frame-portalcarouselwebframefactory.hidden-print/div.frame-standard.panel.panel-front.webframe-ermes-carousel/div.panel-body/div.ermes-frame-html#carousel-4AD04CE72B556A3CCF5DFCF7FB70964E/div.rsItem/div.modele_13.model-html/table/tbody/tr/td/a.vide
```

This shows that our agent detects useful patterns even without semantic meaning, adapting to diverse structures without prior knowledge and supporting an online, per-website learning approach. It further demonstrates how our approach is language-independent: the RL agent learns structural patterns in HTML pages regardless of semantic meaning, enabling efficient target retrieval even on non-English or multilingual websites.

Assessing the precision of our SD retrieval requires counting collected SDs in the targets. Detecting statistics tables in heterogeneous formats is costly: even state-of-the-art methods require roughly one second per PDF page [161]; spreadsheets pose similar challenges [171]. Performing this online or at scale remains an open problem. As an empirical peek into the SD retrieval precision, we manually analyze a random sample of 280 targets from 7 diverse websites in Table 4.6, counting the percentage of targets containing at least one statistics table (“SD Yield”) and the mean number of such SDs per target (“Mean # SDs / Target”). Results show that even on non-statistical websites (*ed*, *in*, *oe*, *wh*), a significant fraction of targets contain SDs; although many targets contain none, SDs concentrate on a subset, yielding, in most cases, more than two SDs per collected target.

4.4.8 URL Classification Design and Quality

All crawls with SB-CLASSIFIER so far use a logistic regression classifier (LR) with URL features (URL_ONLY). We present the confusion matrices of this classifier for each of the 11 fully-crawled websites in Tables 4.7 to 4.17, each averaged over 15 runs. We

4.4. EXPERIMENTAL EVALUATION

Table 4.6: SDs retrieval across sample targets

	<i>be</i>	<i>ed</i>	<i>is</i>	<i>in</i>	<i>nc</i>	<i>oe</i>	<i>wh</i>
SD Yield (%)	82	35	93	40	83	60	40
Mean # SDs / Target	9.1	2.8	2.9	2.1	2.1	4.9	1.4

Table 4.7: Confusion matrix of the URL classifier used in the SB algorithm on website *be* (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	11231	272	0
Target	732	15321	0
Neither	2436	2296	0

Table 4.8: Confusion matrix of the URL classifier used in the SB algorithm on website *cl* (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	1654	24	0
Target	154	3538	0
Neither	104	340	0

Table 4.9: Confusion matrix of the URL classifier used in the SB algorithm on website *cn* (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	5272	7	0
Target	64	7423	0
Neither	8	16	0

Table 4.10: Confusion matrix of the URL classifier used in the SB algorithm on website *ed* (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	85920	2610	0
Target	175	10180	0
Neither	8645	1348	0

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

Table 4.11: Confusion matrix of the URL classifier used in the SB algorithm on website in (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	883706	164	0
Target	117	22362	0
Neither	92544	3922	0

Table 4.12: Confusion matrix of the URL classifier used in the SB algorithm on website is (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	118805	113	0
Target	120	168762	0
Neither	1185	2292	0

Table 4.13: Confusion matrix of the URL classifier used in the SB algorithm on website ju (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	40657	143	0
Target	134	14711	0
Neither	404	1045	0

Table 4.14: Confusion matrix of the URL classifier used in the SB algorithm on website nc (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	217531	2193	0
Target	839	82187	0
Neither	3033	1222	0

Table 4.15: Confusion matrix of the URL classifier used in the SB algorithm on website oe (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
HTML	143412	11071	0
Target	1046	42150	0
Neither	3427	9994	0

Table 4.16: Confusion matrix of the URL classifier used in the SB algorithm on website ok (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
	HTML	Target	Neither
HTML	391802	3556	0
Target	669	12058	0
Neither	12484	2206	0

Table 4.17: Confusion matrix of the URL classifier used in the SB algorithm on website qa (on average, for 15 runs).

True \ Predicted	HTML	Target	Neither
	HTML	Target	Neither
HTML	1311	31	0
Target	77	2356	0
Neither	1320	247	0

Table 4.18: Confusion matrix of the URL classifier, on all fully-crawled websites (average over 15 runs, in percentages)

True \ Predicted	HTML	Target	Neither
	HTML	Target	Neither
HTML	58.04	1.37	0.00
Target	0.75	32.19	0.00
Neither	5.34	2.41	0.00

Table 4.19: Intra-site crawl metric (90% targets reached) and inter-site misclassification rates for URL classifier variants

Variant	be	cl	cn	ed	in	is	ju	nc	oe	ok	qa	MR
URL_ONLY-LR	82.1	75.1	71.3	53.2	11.7	76.1	36.5	52.6	60.7	15.9	62.3	2.62
URL_ONLY-SVM	82.7	75.7	71.8	63.6	11.3	76.0	37.4	52.2	63.5	16.7	61.5	2.99
URL_ONLY-NB	82.9	75.2	72.1	53.7	11.4	76.3	35.8	52.7	59.7	18.0	63.1	2.92
URL_ONLY-PA	82.3	74.4	71.7	53.3	11.1	75.8	36.7	51.6	60.5	15.9	60.9	2.56
URL_CONT-LR	82.2	74.4	71.9	54.3	11.3	76.4	37.8	52.9	64.7	16.8	60.0	5.93
URL_CONT-SVM	82.6	75.0	71.8	52.8	11.6	76.4	38.8	53.1	61.1	18.7	60.1	6.36
URL_CONT-NB	84.1	74.7	71.9	53.6	11.4	75.7	35.5	52.3	59.9	19.1	60.4	7.15
URL_CONT-PA	82.5	75.1	71.9	53.6	11.6	76.2	38.4	52.1	62.6	16.1	60.6	4.12

also summarize these results with a single confusion matrix averaged across all fully-crawled websites, shown in Table 4.18. Tables 4.7 to 4.17 are in number of pages, while Table 4.18 is in percentages (giving equal weight to each of the 11 websites). Classification errors are extremely marginal on “HTML” and “Target” URLs. Never classifying as “Neither” leads to some classification errors (discussed in Section 4.3.3), explaining the difference in requests between SB-CLASSIFIER and SB-ORACLE on the fully-crawled websites. However, since the classifier oracle is unfeasible in practice, SB-CLASSIFIER’s performance is satisfactory.

To study model and feature impacts, we also evaluate a linear SVM (SVM), multinomial Naive Bayes (NB), and a passive-aggressive classifier [50] (PA), restricting to lightweight online models and excluding deep approaches whose cost would shift the bottleneck from network latency to local CPU/GPU time. Each model is used with two feature sets: URL_ONLY and URL_CONT (URL plus anchor text, DOM path, and surrounding text), all encoded as character-level 2-gram BoW, yielding eight variants. Experiments run on fully-crawled websites, with 15 runs per configuration and per site. Table 4.19 reports, for each website and variant, the *intra-website* crawl metric from Table 4.2 (number of requests to reach 90% of targets); variants requiring at least 2% fewer requests than URL_ONLY-LR are in bold. The last column (“MR”) shows *inter-website* misclassification rates on HTML pages and targets from averaged confusion matrices. Figure 4.13 shows SB-CLASSIFIER crawl performance across configurations, for the 11 fully-crawled websites. Tables 4.20 to 4.27 report the inter-website averaged confusion matrices for all eight classifier variants, each obtained by averaging over the 11 fully-crawled websites and 15 runs per website. The “MR” metric is summarized from these confusion matrices as the average misclassification rate on HTML pages and targets. The URL_ONLY variants generally exhibit lower confusion between *HTML* and *Target* pages than their URL_CONT counterparts, suggesting that contextual features bring limited additional discriminative information. On 10 of 11 websites, no variant improves over URL_ONLY-LR by more than 2%; the sole exception is *qa*, where two variants gain at most 2.3%. This lack of consistent gains, also seen in “MR”, shows that neither richer features nor more complex models yield meaningful gains, justifying the choice of URL_ONLY-LR. This is further evident in the plots, where curves are mostly indistinguishable for websites *cn*, *in*, *is*, *ju*, *nc*, and *qa*.

Table 4.20: Confusion matrix of the URL classifier URL_ONLY-LR, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted			
	HTML (%)	Target (%)	Neither (%)
HTML	58.73	1.69	0.00
Target	0.77	32.73	0.00
Neither	4.50	1.58	0.00

Table 4.21: Confusion matrix of the URL classifier URL_ONLY-SVM, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	58.59	1.84	0.00
Target	0.98	32.53	0.00
Neither	4.48	1.60	0.00

Table 4.22: Confusion matrix of the URL classifier URL_ONLY-NB, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	58.44	1.97	0.00
Target	0.77	32.73	0.00
Neither	4.47	1.61	0.00

4.4.9 Multimedia Filtering

As described in Algorithm 4, our crawling strategy incorporates a filtering mechanism based on blocklists of MIME types and URL extensions. This mechanism enables the crawler to selectively exclude specific categories of content, thereby allowing it to focus on resource types that align with the user’s objectives, in addition to the predefined target definition. In our application, which aims to collect statistical datasets, we adopt a broad definition of targets (see Section 4.2.2 for the complete list). This choice serves two purposes. First, it demonstrates that the crawler can effectively adapt to a wide range of MIME types. Second, it reflects the practical reality that statistical datasets are distributed across diverse formats. While the most common representations consist of structured tabular formats (e.g., `xls`, `csv`, `ods`), datasets may also appear in other structured formats such as `xml`, `rdf`, or `json`. They may also be embedded in less structured documents (e.g., `pdf`) or packaged within compressed archives such as `zip` or `tar` files. Conversely, certain formats are highly unlikely to contain statistical datasets, or would make extraction particularly difficult. Moreover, such formats often occur in large quantities or involve substantial file sizes. Multimedia content clearly falls into this category. To assess the impact of this filtering, we conduct an ablation study comparing classifier performance with and without MIME type and URL extension blocklists (see Section 4.4.5 for full lists). Table 4.28 reports the differences in crawl performance between these two configurations.

In terms of request count (Table 4.2), gains are limited. Only *ab* and *be* improve, mostly via URL extension filtering, allowing saved requests to be replaced by deeper crawl exploration, explaining the increase in both target volume and number. Gains are larger when considering target volume relative to non-targets (Table 4.3), with notable

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

Table 4.23: Confusion matrix of the URL classifier URL_ONLY-PA, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	58.60	1.82	0.00
Target	0.58	32.92	0.00
Neither	4.50	1.57	0.00

Table 4.24: Confusion matrix of the URL classifier URL_CONT-LR, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	57.62	2.79	0.00
Target	2.78	30.73	0.00
Neither	4.58	1.49	0.00

improvements for *ab*, *be*, *ce*, *is*, *jp*, and *wo*. On fully crawled websites, filtering also reduces s.d. across runs, increasing stability. For non-fully crawled websites, filtering increases both the number and volume of targets by saving requests. Some sites show no request gain but positive volume gain, occurring when few very large files dominate, confirming that unfiltered multimedia (especially videos) can significantly slow the crawler.

4.4.10 Early-Stopping Mechanism

To prevent the crawler from continuing on websites with few or no remaining targets, we add an early-stopping mechanism. Every ν iterations, we compute a slope $\sigma = \frac{y_t - y_{t-\nu}}{\nu}$, representing the growth rate in the number y_t of targets at crawling step t . We then maintain an exponential moving average $\mu = \gamma \cdot \sigma + (1 - \gamma) \cdot \mu$, with γ the decay rate. If μ stays below a threshold ε for κ consecutive slopes (i.e., $\kappa \cdot \nu$ iterations), crawling stops. Parameters ($\nu = 1000$, $\varepsilon = 0.2$, $\gamma = 0.05$, $\kappa = 15$) provide a good balance between avoiding premature termination and efficiently stopping on exhausted sites. Table 4.2 (below the double rule) shows early-stopping results: the percentage of requests avoided and of targets missed. High values are better for the former, low for the latter. We observe three behaviors:

- (i) Medium to large sites where target discovery slows and stopping occurs effectively (*ab*, *ed*, *in*, *jp*, *ju*, *nc*, *ok*). Smaller sites take longer (in %) to stop due to the required $\kappa \cdot \nu$ iterations.
- (ii) Large websites with continuous target discovery where stopping conditions are never met before manual termination (*as*, *ce*, *il*, *oe*, *wh*, *wo*).

Table 4.25: Confusion matrix of the URL classifier URL_CONT-SVM, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	57.47	2.94	0.00
Target	3.03	30.48	0.00
Neither	4.61	1.46	0.00

Table 4.26: Confusion matrix of the URL classifier URL_CONT-NB, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	58.34	2.08	0.00
Target	4.64	28.87	0.00
Neither	4.58	1.49	0.00

- (iii) Small sites (*be*, *cl*, *cn*, *qa*) where the crawl ends before early stopping can trigger within $\kappa \cdot \nu$ iterations; this is not an issue, as these sites are quickly fully crawled.

Figure 4.15 shows a visualization of the early-stopping mechanism applied to two websites: *in* (left) and *ju* (right). Both fall under the first behavior described in Section 4.4.10, illustrating that smaller websites take longer (in %) to stop after the first signs of target convergence. For *in*, stopping occurs shortly after the point where a human would likely have cut, but for *ju*, the delay is more important. Despite early signs of convergence, stopping is postponed due to the fixed $\kappa \cdot \nu = 15\,000$ threshold, which represents a much larger fraction of the site for *ju* (60 000 pages) than for *in* (one million pages).

4.5 Related Work

We survey the families of Web crawlers most relevant to our setting, discussing how each relates to our approach (Section 4.5.1), then review prior uses of tag paths for Web navigation (Section 4.5.2), and discuss the MAB design that motivates our choice of the Sleeping Bandit formulation (Section 4.5.3).

4.5.1 Web Crawlers

We discuss several aspects and families of Web crawlers.

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

Table 4.27: Confusion matrix of the URL classifier URL_CONT-PA, inter-site average over the 11 fully-crawled websites (each averaged over 15 runs)

True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	57.96	2.45	0.00
Target	1.42	32.09	0.00
Neither	4.54	1.53	0.00

Table 4.28: Impact of multimedia filtering on crawl performance. “Request Gain”: percent increase in requests (as defined in Table 4.2); “Volume Gain”: percent increase in retrieved target volume (as defined in Table 4.3).

	<i>ab</i>	<i>as</i>	<i>be</i>	<i>ce</i>	<i>cl</i>	<i>cn</i>	<i>ed</i>	<i>il</i>	<i>in</i>	<i>is</i>	<i>jp</i>	<i>ju</i>	<i>nc</i>	<i>oe</i>	<i>ok</i>	<i>qa</i>	<i>wh</i>	<i>wo</i>
Request Gain	13.4	0.0	8.2	0.0	0.2	0.0	0.5	0.0	3.0	0.7	0.8	0.0	0.2	0.0	0.5	2.5	0.4	1.7
Volume Gain	42.7	0.0	12.3	12.4	1.6	0.2	0.7	2.3	3.8	18.8	21.1	4.8	3.7	0.0	4.0	5.8	4.7	19.7

Focused Crawlers

Focused crawlers prioritize some pages during the crawl, often based on predefined *topics*. Traditional focused crawlers Chakrabarti et al. [32] and Diligenti et al. [56] mainly train a classifier to predict the likelihood that a hyperlink leads to a relevant page; we adapted this approach to our target retrieval in FOCUSED. Modern focused crawlers typically employ RL to adapt the crawl: we review these RL-based approaches here. We distinguish two kinds of focused crawlers by their *focus*: most are *topical* crawlers [87, 91, 121, 183], which focus on a predefined *topic* via keywords or sample documents. *Non-topical* focused crawlers are rarer: besides this chapter (which focuses on *targets* of a given file type), examples include ANTHELION [130], which focuses on semantic annotations in webpages, and ACEBOT [63], which accumulates as much diverse textual content as possible. ANTHELION’s focus is more specific than ours and ACEBOT’s focus differs but we adapted it for target retrieval in TP-OFF. Topical crawlers are ill-suited for target retrieval as shown in Section 4.4.5, since they target HTML content and require language-specific elements defining the topic. Our SD application requires crawling sites with content about different domains (economics, education, health, etc.) and in multiple languages (Section 4.4.1). Another key difference is that assumptions of topical locality [54] (similar-topic pages are close) and crawl direction [91] (page scores increase near topic-relevant pages) do not typically hold in our case. Existing systems differ in the way they use learning: many choose to have a simple single-state representation [87, 130, 183] in the form of classical MABs, while some introduce a multiple-state MDP representation as in [91] or in [121] which directly extends on [91]. In order to support an MDP representation, Han et al. [91] and Kontogiannis et al. [121] have to define states that cannot just be the current state of the crawl but a summary thereof (as RL policy optimization requires visiting each state many times). For this, they use features characterizing the topic relevance of the last crawled page

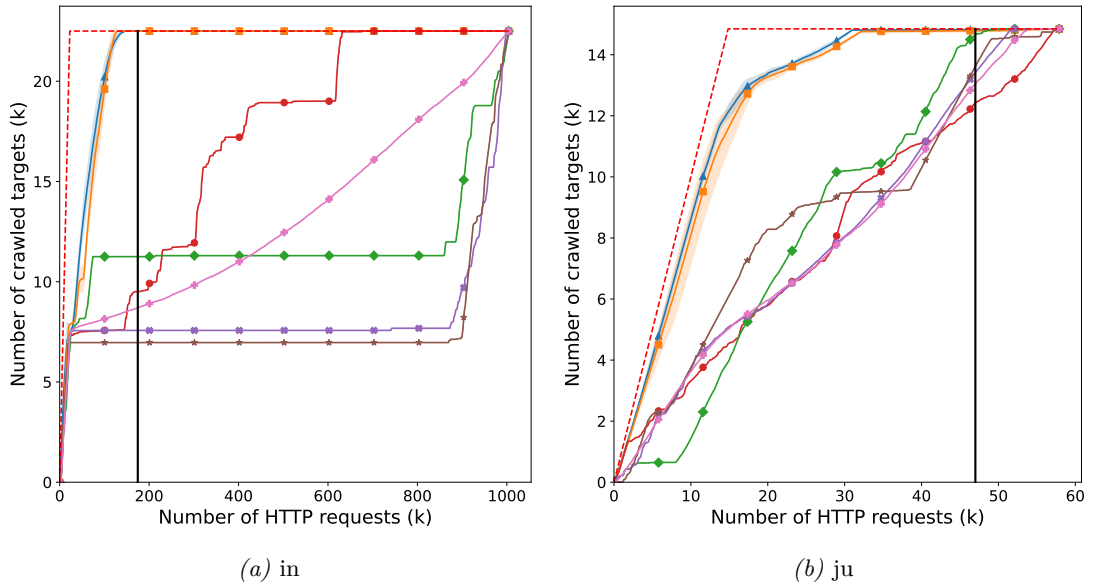


Figure 4.15: Early-stopping time (black rule) for in and ju

and its neighborhood. This does not have a direct equivalent in the setting of target retrieval (again, because of non-locality). We differ from prior work by choosing an SB representation, a mathematically well-founded way to both use a single state and model that not all actions are available at every step. It also allows faster convergence of action estimation than multiple-state approaches. Unlike our method, Faheem and Senellart [63] do not use RL but rely on offline training on part of the website (see Section 4.4.3). In most cases, including ours, actions correspond to selecting the next URL to retrieve. Two works differ: Meusel et al. [130] apply RL only to select *sites*, to crawl the next URL from, with URL choice handled separately, while Zhang et al. [183] additionally allow actions that query a SE. Finally, except for [63], our work differs from all cited approaches in targeting a focused crawl of a single given website: the most common approaches crawl across sites without strict scope constraints. This distinction is crucial for our application, which requires retrieving datasets exclusively from trusted sources. Table 4.29 summarizes the main characteristics of the focused crawlers we discuss, to help identify how our approach differs.

Distributed Crawlers

Distributed or *parallel* Web crawlers use multiple crawlers concurrently to speed up page retrieval. Key challenges lie in resource (especially, network) management. Cho and Garcia-Molina [42] introduced general architectures while Boldi et al. [23] proposed UBI-CRAWLER, a decentralized and distributed system using hashing for domain partitioning. Others [24, 34] focus on social network crawling. These works aim to efficiently crawl a

CHAPTER 4. EFFICIENT FOCUSED WEB CRAWLING FOR SCALABLE DATA ACQUISITION

Table 4.29: Characteristics of focused crawler works (*LI* = Language-Independent; *LeM* = Learning Model; *Code* = Code publicly available)

System	Focus	LeM	States	Actions	Scope	LI	Code
Gouriten et al. [87]	Topic	Bandit	—	URL selection	Across sites	✗	✗
ANTHELION [130]	Semantic annot.	Bandit	—	Site selection	Across sites	✓	✓
ACEBOT [63]	Text content	Offline	—	URL selection	Site-by-site	✓	✗
Han et al. [91]	Topic	MDP	Relevance features	URL selection	Across sites	✗	✗
DSDD [183]	Topic	Bandit	—	URL selection or SE use	Across sites	✗	✗
TRES [121]	Topic	MDP	Relevance features	URL selection	Across sites	✗	✓
FOCUSED	Targets	Online	—	URL selection	Site-by-site	✗	✓
SB-CLASSIFIER	Targets	Sleeping Bandit	Available tag paths	URL selection	Site-by-site	✓	✓

given set of pages, whereas we focus on minimizing that set. The two could be combined to improve crawling speed and reduce network load.

Crawlers Targeting Website Categories

Web crawlers targeting specific website types (e.g., forums, blogs, or CMS) leverage specific and common structural patterns within these sites. For instance, Guo et al. [88] and Cai et al. [29] focus on forums. While effective, these crawlers are less general, relying on assumptions about site structure. In contrast, our approach adapts to each website dynamically, and needs no assumptions (see Section 4.2.1).

Deep-Web Crawlers

Hidden- or deep-Web crawlers explore content not reachable via standard hyperlinks, often requiring user interactions such as filling out forms. A comparative study of state-of-the-art deep-Web crawlers appears in [93]. In contrast, our work focuses on acquiring specific targets from official websites. In our context, form-based interfaces serve mainly as filters, which are not useful for large-scale retrieval. Moreover, on many institutional websites (see Section 4.4.1), such data is accessible through standard navigation, not portals. Extending our approach to deep-Web crawling remains a natural direction for future work (see Section 4.6).

Incremental Crawlers

Incremental or revisit policy-based crawling views a website as an evolving set of pages and selectively revisits them to maximize updated content while minimizing unnecessary requests. Cho and Garcia-Molina [41] present core challenges, and Sigurdsson [160] introduces an incremental version of HERITRIX [137], used by the *Internet Archive*. More recent works propose learning-based revisit policies: [9, 53, 120] predict emergence of new outlinks from page features. Others use RL: Schulam and Muslea [154] show Thompson Sampling (TS) outperforms MAB alternatives and the PHAM CRAWLER [147], and Kolobov et al. [119] introduce a learning algorithm to compute optimal revisit policies. While our method is a single-shot crawl, we plan to build upon it to implement incremental strategies.

4.5.2 DOM Tag Path Representations

Tag paths, e.g., *root-to-link* paths in the DOM of each HTML page, have been exploited for Web crawling in prior works. Crescenzi et al. [51] and Crescenzi et al. [52] take advantage of the website structure to cluster webpages, with applications to *Web scrapers* aiming to automatically extract and structure data from HTML pages. Faheem and Senellart [63] directly uses paths in the DOM of each HTML page in order to do focused Web crawling.

4.5.3 Multi-Armed Bandits Algorithms

While UCB [7] and its variants are the state of the art for solving MAB problems, simpler approaches like ϵ -greedy [167] randomly select actions with some probability and otherwise choose the highest scoring one. We also considered Bayesian Bandits, especially TS, a probabilistic method based on posterior distribution estimation. We excluded those approaches to ensure crawler *stability*, i.e., reproducibility across runs on the same site. Also, Bayesian approaches require prior distributions, unavailable to us due to lack of prior knowledge about the websites. We also did not adopt *linear* or *kernel*-based bandit methods, despite their generalization capabilities. These approaches require stable, qualitative feature representations, extracted from webpages, which are hard to define in our context. Indeed, while we hypothesize that links appearing in similar tag paths lead to similar content, this does not guarantee the existence of consistent *content*-based features (not structure) of the pages. We would need to choose *universal* features, i.e., computable for any page of any website we want our crawler to visit. Moreover, most feature-based methods are language-dependent, which conflicts with our objective. Combined with already good performance, this led us to focus on AUER [112], UCB’s sleeping variant.

4.6 Summary

In this chapter, we addressed the problem of scalable Web data acquisition by developing an efficient crawler that aims to maximize the retrieval of targets while minimizing computational resource usage in terms of time and bandwidth, and respecting crawling ethics. Our RL-based solution outperforms several baselines on a heterogeneous set of websites. Directions for future work include exploring more complex reward functions than simply counting reachable targets and integrating deep-Web crawling techniques to improve access to data behind forms or within portals more efficiently, enhancing large-scale acquisition. Finally, while our crawler focuses on the initial acquisition of targets, it does not handle updates or newly published resources: an important limitation for statistical data journalism, where timely information is essential. We leave extending our crawler with *incremental revisits* for future work, combining the knowledge acquired by our RL agent with existing re-crawling strategies.

Efficient and Scalable Search for Statistics

5.1 Introduction

The value of a metric, for a given location and time, such as “deaths in Belgium in 2021”, is a useful ingredient for public debate. Yet, finding it among published statistical datasets is not easy: a statistics file may be very large, sometimes published compressed so that search engines do not index its content, and cumbersome for a user to search by hand across millions of rows.

Making such values accessible requires methods that, given a natural-language question and a collection of statistical tables, return the most relevant ones, a task commonly referred to as **table discovery (TD)**. The related **table question answering (TQA)** problem focuses on extracting answer(s) from one or a few given tables. As discussed in Chapter 3, state-of-the-art TD methods [11, 90, 175] target entity-oriented relational tables. In contrast, the STATCHECK system (Chapter 3) addresses TD *for statistical datasets* [14, 16]. The retrieval method proposed by STATCHECK serves as the foundation for the work presented in this chapter.

STATCHECK is based on the observation that *a statistical dataset (table) is an (unordered) set of statistical facts*, where each fact consists of a set of *dimensions*, together with their values, and a *measure*. For instance, in the example above, the dimensions with their respective values are (“location”=“Belgium”), (“year”=“2021”), while the measure is “number of deaths”. Further dimensions may be present in the data and/or in users’ search, e.g., (“age”=“toddlers”), (“prior pathology”=“asthma”), etc.

Understanding a statistical dataset as a set of facts enables investigating *order-independent* methods for TD and TQA; this has also been noted in [11, 14, 30, 175]. In contrast, several existing TQA methods start by *serializing* table content (typically rows) and leveraging Language Models (LMs, in short), possibly fine-tuned on these serializations [65]. Such methods, by design, reflect (or learn) *the serialization order* of cells in a row, yet this order is meaningless in statistical tables. As a consequence, result quality may be negatively impacted. A second core insight in STATCHECK [14, 16] is that *the fact values, which make up a majority of cells in statistical tables, answer questions but are not part of the questions themselves*, neither for TD nor for TQA. Only the table cells *other than fact values* need to be pre-processed (indexed), so that at query time *the index can lead to the fact values* that answer the question. As we will show, this

keeps STATCHECK indexes of reasonable size even on large statistical corpora, on which indexing is unfeasible (taking excessive time or space) or much slower for competitor systems [11, 90, 175]. Finally, the novel insight of [16] is: statistical data dimensions *time* and *space* are omnipresent in statistical data, and frequent also in user queries. Therefore, the *Space- and Time-Aware Statistic Retrieval* (**STAR**, in short) method solves the TD problem by *extracting space and time dimension values from the statistics*, and from user queries, and *separately processing the time, space, and remaining part of the search query*. This chapter builds on an earlier iteration of the STATCHECK project [16].

Contributions

In this chapter, our contributions are as follows.

- (1) We formalize the **Statistical Table Discovery** (STD) problem as a variant of table discovery tailored to the particularities of multidimensional statistical tables, and characterize its distinction from entity-centric (relational) table retrieval (Section 5.2).
- (2) We present **STAR** (Space- and Time-Aware Statistical Retrieval), a novel STD method that separately indexes and retrieves the space, time, and semantic dimensions of both tables and queries (Sections 5.3 and 5.4). Key contributions include: the use of PEARL embeddings in place of SBERT (improving result quality by up to 14%), a relaxation mechanism for queries with no exact spatio-temporal match, support for natural language questions, and a best-effort TQA mode that narrows results down to the most relevant facts within a matching table.
- (3) We provide a **systematic analysis of STAR’s architectural flexibility**, covering language model modularity, multilingual support, and a consolidated quantitative comparison of key configuration choices that makes the rationale behind STAR’s design explicit (Section 5.5).
- (4) To support further research on the STD problem, we introduce a **novel STD benchmark** comprising 7 605 *Eurostat* statistical tables, 200 carefully curated questions with gold-standard answers, and nearly 5 000 manually annotated (question, table) pairs — the first STD benchmark on high-quality, real, and varied Open Data (Section 5.6.3).
- (5) We conduct a **comprehensive experimental evaluation** comparing STAR to state-of-the-art systems (Section 5.6), showing that:
 - (i) when queries use exact dataset terminology, BM25 provides the best result quality with fast indexing;
 - (ii) when queries are reformulated with synonyms, *STAR with PEARL yields the highest quality* while BM25 degrades sharply;
 - (iii) *STAR scales to large statistical corpora* where SOLO [175] and BIRDIE [90] run out of memory and PNEUMA [11] is (at least) 12× slower.

It is also worth noting that STATCHECK with STAR is much more *frugal* than recent competitors [11, 90, 175]: STAR does not rely on large language models (LLMs), reducing the energy consumption and computational costs incurred by statistical question answering; also, running it does not require a GPU. While LLMs have unparalleled abilities for text generation, summarization, etc., training and using them has significant environmental costs [31], which argues for using them only for problems that lower-footprint tools cannot address. On the STD task, STATCHECK with STAR significantly outperforms baselines *simultaneously* in terms of answer quality, indexing and query speed, scalability, and low resource use.

This chapter is organized as follows. Section 5.2 defines background concepts and presents our problem statement. Sections 5.3 and 5.4 describe our approach for solving the STD problem. Section 5.5 discusses STAR’s architectural flexibility, multilingual capabilities, and key configuration choices. Section 5.6 presents our experimental validation. We discuss related work in Section 5.7 before concluding. The datasets, labeled evaluation questions, experiment results and scripts to compute metrics are available at [72]. The STAR source code is available at [73].

5.2 Statistical Table Representation and Problem Setup

We first introduce the data model underlying statistical tables, defining the key concepts of metric, dimension, and fact in the tradition of data warehouses (Section 5.2.1), then formally state the Statistical Table Discovery problem and explain why standard table discovery methods are ill-suited to address it directly (Section 5.2.2).

5.2.1 Statistical Table Data Model

A statistical dataset, encountered on the Web under the form of a file in Excel, CSV, SDMX [155] format, etc., comprises information which we describe in classic data warehouse [100] and temporal databases [101] terminology below.

A metric is something measured multiple times and reported in a dataset, e.g., “unemployment”, “food bank beneficiaries”, “tennis amateur licenses”, etc. A dimension characterizes a certain property (feature) of a metric. Sample dimensions are the time and location attached to the facts counted by the metric; age, gender, or other properties of a population, etc. A dimension may have multiple granularities, that may be hierarchically organized in a directed acyclic graph (DAG, in short). For instance, location can be reported at granularities: continent, country, region, department, city, etc. Similarly, time can be reported at granularities: year (Y), quarter (Q), month (M), week (W), day (D), e.g., child births are counted per day of the year. A dimension can take different dimension values. The granularity of a value is the lowest level in the dimension hierarchy where the value is given. For instance, “2025”, “March 2025”, and “March 13, 2025” are three temporal values. Their granularities are, respectively, Y, M, D. A fact consists

of: a metric; a set of (dimension = value) pairs; a “datum”¹ which is a numeric value; optionally, one or several comments, e.g., “under consolidation”, or “partial data as of 04/2024”, etc. The datum can be an absolute value, or a relative one, e.g., a share or percentage. In practice, the dimension name may be missing, e.g., when deaths are separately counted among Men and Women (values of the dimension “gender”), the word “gender” (dimension name) may be absent.

A table contains a set of facts together with an optional title and some comments. For example, in Figure 5.1, the title appears at the top underlined; below on the left, we depict some cubes (with the dimensions: country, branch of the economy, and year), each of which is characterized by one value for the (implicit) dimension “age group”. These cubes hold a set of four-dimensional facts. At the right of this conceptual representation, we show a statistical table as it appears in an Excel or CSV file, or HTML table, within a statistical portal.

A dataset is a set of one or more tables, together with an optional *dimension value dictionary*. The latter is a file associating, to some dimension values that are *compact codes*, a more human-understandable description. Codes are used on the one hand to save space, and for language-independence, i.e., the same code has different human-understandable names in different languages. For instance, *Eurostat* uses the codes FR, RO, PT for France, Romania, and Portugal, respectively; RO is decoded as Romania in English, Roumanie in French, etc. A publication consists of one or more datasets, together with an optional description, that may contain a description title and a description body. Each publication has a publication URI. A statistical source is a provider of tables, such as *INSEE*, *Eurostat*, the *US Census Bureau*, etc.

5.2.2 Statistical Table Discovery Problem

Let $\mathcal{D} = \{T_1, T_2, \dots, T_n\}$ be a dataset containing n tables T_i , $1 \leq i \leq n$. Without loss of generality, we assume all tables from $\mathcal{D}$ come from the same provider (source) P ; our approach directly generalizes to multiple providers. We consider **natural language questions** requesting one or several fact values. The question is free text, but we assume it mentions at least a metric and typically one or several dimensions values, that may (or may not) be accompanied by dimension names. In the sample question q_1 : “How many US people own a smartphone?” the metric is “people owning a smartphone” and the dimension is “location”=“US”. In the sample question q_2 : “What part of EU agricultural subsidies has been spent on soybean-growing farms in Poland in 2020?”, the metric is “agricultural subsidies” while the dimensions are “location”=“Poland”, “category”=“soybean-growing farm”, “year”=“2020”. Note that:

- (i) dimension names are often implicit, in particular location and time;
- (ii) some crucial dimensions may be absent, e.g., the time dimension in q_1 , even though it plays an important role in determining the fact (the percentage of US population owning a smartphone grew from 35% in 2011 to 91% in 2024 [146]);

¹While this is the *value of the metric*, we call it data to avoid a confusion with the *dimension values* above.

- (iii) metric names are specified in varied, flexible ways, characteristic of natural language questions; in particular, they may differ quite significantly from the metric names used in $\mathcal{D}$.

We say a **fact** f , present in table T_i , **answers a question** q when the fact’s metric and dimension-value pairs are identical or close to those stated in the question. Several facts may answer a given question, e.g., if the question specifies fewer dimensions than the facts have. For instance, high-school students studying Advanced Maths may be recorded by gender in a table, while the question does not mention gender. Also, some facts may answer a question better than others, e.g., a fact f_1 about “job seekers for less than 3 months” answers a question about “short-term unemployment” better than a fact f_2 about the “long-term unemployed”. We say a table T **answers a question** q when T contains at least one fact answering the question.

Problem 5 (Statistical Table Discovery)

Given a corpus $\mathcal{D} = \{T_1, T_2, \dots, T_n\}$ of statistical tables from source P and a natural language question q , find a ranked list $T_1^q, T_2^q, \dots, T_N^q$ ($N \geq 1$) of tables from $\mathcal{D}$, each answering q to some extent, sorted in decreasing order of relevance (score) to q .

Solving this problem requires a scoring method that quantifies how relevant a fact is for a question; given the flexibility and ambiguity of natural language, the gold standard comes from human evaluation; prior systems made different proposals in this area. We will present our ranking method below.

Why TD does not solve (well) STD. One may wonder why not convert each statistical table T into a relational one, R_T , then apply existing TD methods on the latter. Assume T is characterized by the dimensions $d_1, \dots, d_m$, where each d_i ranges over n_i distinct values for some integers $1 \leq i \leq m$, $1 \leq n_i$. T holds $n_1 \cdot \dots \cdot n_m$ facts, which is also its number of cells (ignoring one or a few header row(s) and/or line(s)). R_T should have one attribute for each dimension, and one row for each fact f , of $m + 1$ cells (one with each d_i value characterizing f , and one with the measure value of f). Thus, R_T has $(m + 1)$ times more cells than T . The actual table size increase is even higher, due to repeated appearance of dimension values in all the rows; even though dimension values are short strings, they still occupy more space than measure values (numbers). This is problematic even for very recent TD systems that *run out of memory* on large tables (see Section 5.6). But even more problematic is the fact that *the relational table will lack many attribute names*: these would be the *dimension names*, that are often absent in statistical datasets. Moreover, TD methods on relational tables *crucially depend on the attribute names* (table schema), see Section 5.7. This is why below we develop specific STD methods.

5.3 Statistical Table Indexing

We now describe our method for *analyzing and indexing the tables* prior to question answering: linguistic content (Section 5.3.1), respectively, time and location (Section 5.3.2).

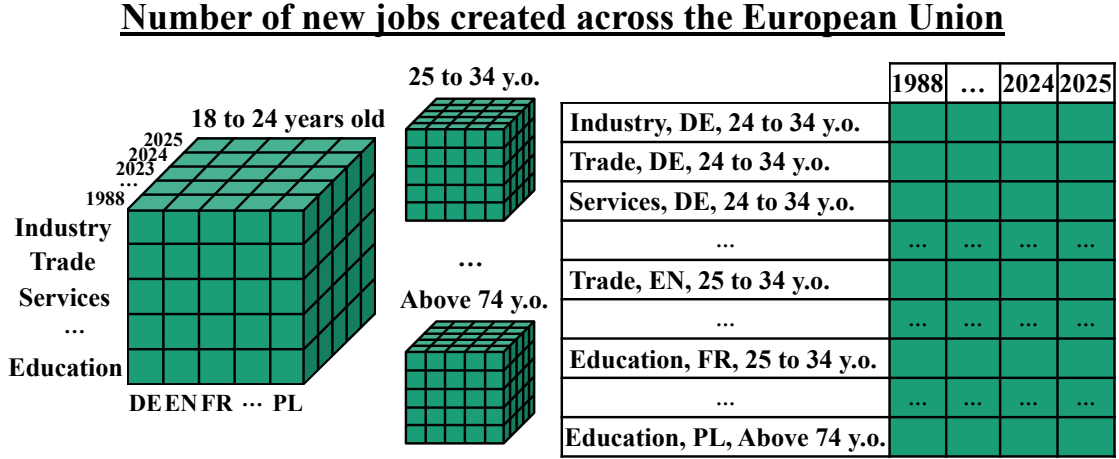


Figure 5.1: Sample statistical dataset: conceptual multidimensional view (left); two-dimension serialization (right).

5.3.1 Linguistic Content Indexing

The content of a table is described (in human-understandable terms) in its title, comments, and any text found in headers. A proper representation and indexing of these components is crucial for both the retrieval and re-ranking phases. Accordingly, most TD methods, e.g., [11, 90, 175], and modern TQA techniques, e.g., [94], compute from each table one or several texts, which are then projected in a high-dimensional space via a Language Model embedding.

Given a table T , several methods can be used to reflect the linguistic content of T (again, recall that data are numbers and not text). We call these **table representation functions**, each of which returns a set of one or more strings:

- $f_t(T)$ returns the table title.
- $f_{ch}(T)$ returns the concatenation of all column header cells, which always occupy the first (topmost) row in a table, and may or may not span over a few more subsequent rows. These have been used in many prior works, e.g., [38, 159], because they intuitively correspond to a relational table’s schema (attribute names). However, in statistical tables (Section 5.2.1), interesting linguistic content is also present in row headers, i.e., leftmost column(s).
- $f_{rh}(T)$ returns the concatenation of all row header cells, i.e., those in the leftmost column(s), symmetrically to f_{ch} .
- $f_{\{h\}}(T)$ returns a set of all distinct strings appearing in a (row or column) header cell of T .
- $f_h(T)$ returns the concatenation of all row and column headers, i.e., the concatenation of strings in $f_{\{h\}}(T)$.

For instance, in the statistical table in Figure 5.1, $f_t(T)$ is “Number of new jobs created across the European Union”, header cells include “Industry, DE, 24 to 34 y.o.” and all the cells below, as well as “1988” up to “2025”. $f_h(T)$ is a single string starting with “1988”, and ending with “Education, PL, Above 74 y.o.” with some separator. $f_{ch}(T)$ is simply the concatenation of column headers “1988” to “2025”, again with separator. Whatever the choice of a function f , we will apply an *embedding*, denoted φ , to transform each $f(T)$ into a multidimensional vector, which we insert into a **multidimensional index** providing efficient **approximate nearest neighbor (ANN)** look-up. At query time, an embedding $\varphi(q)$ computed from the question is used as a lookup key, and we search for the closest vectors, thus, the closest table definitions.

The choice of the string representations, and, to a lesser extent, the choice of the embedding method, are crucial for the quality of our solutions. Regarding the string representations, we make the following remarks.

First, f_t may miss a large part of the linguistic content of a table. For instance, in our example, it does not include the words “Industry” and “EN” which may lead to the sample table being (wrongly) considered not very relevant for a question about “industry in England”.

Second, as mentioned in Section 5.1, functions such as f_{ch} and f_h , which concatenate several header cells from a table T , impose a certain *order* over their content, however, this order is meaningless from the perspective of understanding the data. For instance, “Belgium” being just before “Bulgaria” in a sequence of header cells storing countries does not entail any connection between them. Language models, however, are strongly sensitive to proximity and order between words, given that these are crucial in natural language. Thus, the semantic signal of the concatenated string suffers from some noise.

Third, concatenating dimension values leads to long strings which may make a table appear not similar to the question. Considering again the “industry in England” question, if all economy branches appear in the row headers, f_h will include “Agriculture”, “Tourism & Hospitality”, “Commerce”, etc., as well as “Industry”. *The linguistic signal of the concatenation may appear to differ significantly from each of the concatenated strings, again potentially leading to a false negative.* This has been experimentally demonstrated in STAR [16] and it has been known under the name “Lost in the middle” in Natural Language Processing (NLP) research [125].

Fourth, note an important difference with respect to prior TD or TQA methods for relational (not multi-dimensional) data. In relational tables, *questions can carry over any attribute in any record*. For instance, “what is the address of the city hall of Boston?” needs the “city name” and “city hall address” attributes from the row describing Boston. In such settings, indexing $f(T)$, where f is a table representation function that does not reflect all the table content (all attributes from all the rows), risks false negatives: an index look-up may fail to find an entry matching the question, even when one exists. This is why recent TQA methods rely on a first step (typically a TD system) to *pre-select a small set of tables*, which are *integrally given (at query time) to an LM* that has been trained to extract from each table the most relevant facts for a question. When the answers are in tables of hundreds of thousands, or millions of lines, depending on the

LM, it may be impossible to feed them to the model, or some chunking may be needed, and TQA may be slow. Also, (i) depending on the LM used, a GPU may be needed, or there may be significant computational costs; and (ii) LM answers do not always come with a clear explanation or provenance. In contrast, in the STD problem, *questions only refer to the linguistic content* of tables. This enables us to index this content only, and not the many other table cells (fact values).

Based on these four observations, the representation $f_{\{h\}}$, returning all header cells' content, is more likely than f_h and f_{ch} to avoid false negatives. Further, table titles contain information that may be absent from the headers, e.g., the measure name. For this reason, we decide to also use f_t . Thus, the total set of strings that we need to embed through φ and index for a table T is $f_t(T) \cup f_{\{h\}}(T)$.

5.3.2 Separate and Exact Spatial and Temporal Indexing

Directly embedding $f_t(T) \cup f_{\{h\}}(T)$ still has limitations.

First, in short titles, location names may skew the similarity between titles and questions. For instance, “Jobs in Saint-Pierre-et-Miquelon” (a French overseas territory) may seem similar to “Health in Saint-Pierre-et-Miquelon”, even though the topics are quite different.

Second, when embedding locations, similarity between $\varphi(l_1), \varphi(l_2)$ may not correlate well with users' natural understanding of location hierarchies (Section 5.2.1). For instance, let l_1, l_2 be the French regions “Île-de-France” and “Provence-Alpes-Côte d’Azur”, and l_3 be the French department “Yvelines”, a subdivision of l_1 . Then, $\varphi(l_1)$ may be closer to $\varphi(l_2)$ than to $\varphi(l_3)$ because both l_1 and l_2 are regions thus they appear in similar contexts in the language's training data, e.g., “the region council has decided”, “the region's subsidies to high schools”, etc. However, for a question about some facts from “Île-de-France” (l_1), a table with the right facts from “Yvelines” (l_3) is clearly relevant, because l_3 is in l_1 , whereas the same facts measured in l_2 are not (wrong location). Similar remarks hold concerning time values. For human users, “Q3, 2023” and “November, 2023” are clearly part of “2023”, while “November 2024” is not, even though linguistically it may look more similar. Thus, among locations, or among time values, **the meaningful relationship that should be exploited by question answering is inclusion** (parent-child or ancestor-descendant in the respective DAGs, discussed in Section 5.2.1).

For these reasons, we decide to **extract location and time information** from f_t and $f_{\{h\}}$ results, as follows. We call *spatial value* (or **sval**, in short) any individual spatial unit, e.g., “New York City”, “NY State”, etc. The **spatial hierarchy** of all sval in the dataset $\mathcal{D}$ is a directed acyclic graph (**DAG**). We call *temporal value* (or **tval**) any mention of time found in the data; tvals are also organized in a natural **time hierarchy**.

Locations on Earth are described by high-quality, Open Data reference sources, e.g., *GeoNames*, or dictionaries standardized by statistical institutes, e.g., Cog [1] or NUTS [2]. Building a complete hierarchy of time values over a given interval and at a given granularity is even easier. Thus, without loss of generality, we will denote by $\boxed{H_S}$, respectively, $\boxed{H_T}$ the hierarchies of space and time dimension values that may occur

in $\mathcal{D}$. In particular, the geographic reference H_S (also) provides *alternate names* for a given sval reflecting renaming along the time, shortnames and acronyms, or different languages. H_S may also contain *same-name svals*, e.g., “Hamburg” as the name of a German city or of the corresponding state.

Based on H_S , we assign to each statistical source (Section 5.2.1) a **default location**, e.g., for Eurostat, this is “the 27 EU states”, for the US Census Bureau, it is “USA”, etc. This will allow interpreting user questions that do not specify a location, as we explain below.

We **index each table T by its sval and tval values**, as follows. We call **spatial scope**, respectively, **temporal scope** of a table T , denoted σ_T , respectively, τ_T , its sval, respectively, tval sets.

To compute σ_T and τ_T , first, we extract location and time occurrences from the table title $f_t(T)$, using a small trained language model. We also need to extract them from the many table header cells. However, in contrast with the title that is a well-formed phrase, each cell usually contains a very short string, on which extraction performed poorly due to the lack of context. This happened especially with locations, which may be confused with dimension values, e.g., Apple is the name of a location², but can also appear in a header cell to specify the fruit, e.g., when reporting fruit production. Thus, we apply the following heuristic:

- (i) We tentatively identify svals in header cells by looking for an exact match with the name (or an alternate name) from the geographic reference H_S . If we find a sval label shared by two different spatial reference nodes, such as “Hamburg” previously mentioned, we include in the table’s spatial scope all such sval nodes.
- (ii) We tentatively identify tvals based on pattern matching. More advanced time value extraction tools exist, e.g., HEIDELTIME [162], yet in our experiments HEIDELTIME did not improve accuracy, possibly because time values are well-written in statistical datasets, thus pattern matching is quite successful.
- (iii) If at least $t\%$ of the values in a given header column (or row) are time (resp., location) values, we consider all of them to be of this type.

For instance, in the table in Figure 5.1, the values in the top row are all tentatively labeled years and then all validated as such. In the leftmost column, the country codes DE to PL are tentatively mapped to H_S and then, based on the fact that there is one in every row, they are all identified as svals. Thus, τ_T is $\{1988, \dots, 2025\}$, while $\sigma_T = \{\text{DE}, \text{EN}, \dots, \text{PL}\}$.

For each table T , all the pairs $(s, T.id)$ for $s \in \sigma_T$ are inserted in a **key-value index** I_S . Similarly, all $(t, T.id)$ pairs where $t \in \tau_T$ are inserted in a **key-value index** I_T .

Finally, to avoid the undesired impact of location and time on string embeddings mentioned at the beginning of this section, from each string $s \in f_t(T) \cup f_{\{h\}}(T)$, we remove any sval and any tval identified as above. We call the remaining string the

²https://en.wikipedia.org/wiki/Apple,_Oklahoma

Algorithm 5: Compute Scores for Candidate Tables

Input: Stripped question $\bar{q}$, candidate table IDs C^1 , penalty pen , and threshold θ

Output: Occurrences and scores of tables

- 1 $\varphi(\bar{q}) \leftarrow$ Embedding of $\bar{q}$;
- 2 $C^2 \leftarrow (L_t(\varphi(\bar{q}))_{1:N_1} \cup L_h(\varphi(\bar{q}))_{1:N_1}) \cap C^1$;
- 3 Remove entries in C^2 with similarity $< \theta$;
- 4 **foreach** $(\varphi(\bar{x}_T), T.id) \in C^2$ **do**
- 5 Record occurrence of $T.id$ with x_T ;
- 6 **foreach** table T with at least one occurrence in C^2 **do**
- 7 $score(T) \leftarrow \sum_{x_T} \text{sim}(\varphi(\bar{q}), \varphi(\bar{x}_T)) - pen$;
- 8 **return** Occurrences and $score(T)$;

stripped version of s , denoted $\bar{s}$. We do so carefully based on a syntactic analysis of s , so that $\bar{s}$ remains as natural as possible from a linguistic perspective. For instance, stripping “premature deaths in Poland in 2022 per maternal age” yields “premature deaths per maternal age”: we remove “Poland”, “2022”, and also their leading “in”. A stripped string may be empty, if it only consisted of location or time values; this is the case for the header cells on the top row in Figure 5.1.

For each non-empty stripped string $\bar{s}$ obtained from $f_t(T)$ and $f_{\{h\}}(T)$, the embedding $\varphi(\bar{s})$ is inserted in an **ANN index** $\boxed{L_t}$ (linguistic content from titles), respectively, $\boxed{L_h}$ (linguistic content from headers), paired, in both indexes, with the table ID $T.id$.

5.4 Statistical Table Retrieval

We now describe how to retrieve the tables most relevant to a user question q , using the indexes built in Section 5.3: the core retrieval procedure (Section 5.4.1), a relaxation mechanism for questions whose location or time does not match any table’s scope exactly (Section 5.4.2), and a brief discussion of how STAR can further narrow results down to the most relevant facts (Section 5.4.3).

5.4.1 Retrieval Procedure

Algorithm 6 presents the procedure for retrieving the tables most relevant to a natural language question q . The following provides a detailed explanation of its operation.

We begin by extracting possible **question time and location values** from q just like we did from table titles during the pre-processing stage. Because q is a phrase, a language model yields good results for this task. This leads to zero or more H_S locations, and zero or more time values from H_T . The typical question we expect has one of each; we denote them s_q and t_q respectively, and we describe the next steps considering them

Algorithm 6: Statistical Table Retrieval

Input: The natural language question q
Output: The top- N most relevant tables to q

```

1  $\tau_q, \sigma_q \leftarrow$  Time and location values from  $q$ ;
2  $s_q, t_q \leftarrow$  Unique location and time of  $q$  (conjunctive semantics otherwise);
3  $C_q^1 \leftarrow I_S(s_q) \cap I_T(t_q)$ ;
4 if  $C_q^1 \neq \emptyset$  then
5   | Occurrences, scores  $\leftarrow$  Algorithm 5 with  $C^1 = C_q^1$ ,  $pen = 0$ ;
6 else
7   | Initialize relaxation distance  $d \leftarrow 1$ ;
8   | while fewer than  $N$  results found do
9     |  $(s_q)^d \leftarrow$  neighbors of  $s_q$  at distance  $d$  in  $H_S$ ;
10    |  $(t_q)^d \leftarrow$  neighbors of  $t_q$  at distance  $d$  in  $H_T$ ;
11    | foreach  $t^d \in (t_q)^d$  do
12      |  $C^1 \leftarrow I_S(s_q) \cap I_T(t^d)$ ;
13      | if  $C^1 \neq \emptyset$  then
14        | | Occurrences, scores  $\leftarrow$  Algorithm 5 with  $C^1$ ,  $pen = d \cdot p$ ;
15    | foreach  $s^d \in (s_q)^d$  do
16      |  $C^1 \leftarrow I_S(s^d) \cap I_T(t_q)$ ;
17      | if  $C^1 \neq \emptyset$  then
18        | | Occurrences, scores  $\leftarrow$  Algorithm 5 with  $C^1$ ,  $pen = d \cdot p$ ;
19    | foreach  $s^d \in (s_q)^d$  do
20      | foreach  $t^d \in (t_q)^d$  do
21        | |  $C^1 \leftarrow I_S(s^d) \cap I_T(t^d)$ ;
22        | | if  $C^1 \neq \emptyset$  then
23          | | | Occurrences, scores  $\leftarrow$  Algorithm 5 with  $C^1$ ,  $pen = 2 \cdot d \cdot p$ ;
24    |  $d \leftarrow d + 1$ ;
25 return top- $N$  tables by decreasing score( $T$ );
    
```

present (otherwise, the steps based on the absent s_q and/or t_q are skipped). If there are more than one location (or time) values, we assign them a conjunctive semantics, i.e., we search for tables containing *as many (ideally all)* of the question's s_q and t_q values as possible. With s_q and t_q in hand, we retrieve and score candidate tables as follows.

1. We look up I_S with s_q , respectively, I_T with t_q . Each of these look-ups yields a set of table IDs; let C_q^1 , the set of candidate table IDs for q , be their intersection. If $C_q^1 = \emptyset$, no table has in its scope both s_q and t_q ; we handle such cases via *question relaxation* (see below). For now, we handle the case when $C_q^1 \neq \emptyset$ (steps (2)–(5) are described by Algorithm 5).

2. We compute the embedding $\varphi(\bar{q})$, where $\bar{q}$ is the user question, stripped as described in Section 5.3.
3. We then perform a look-up in the ANN index L_t with $\varphi(\bar{q})$ as search key *and* C_q^1 as *constraint*, i.e., we look for the N_1 multidimensional vectors closest to $\varphi(\bar{q})$ and whose associated table IDs are in C_q^1 . N_1 is an integer larger than N , the number of desired results (recall Section 5.2.2). This finds the C_q^1 tables whose titles are closest to the stripped question. Similarly, we look up in L_h with $\varphi(\bar{q})$, to find the header cell(s) most similar to $\bar{q}$.
4. Call C_q^2 the union of the results of the above two look-ups (in L_t and L_h). Each entry in C_q^2 is of the form $(\varphi(\bar{x}_T), T.id)$ where $x_T \in f_t(T) \cup f_{\{h\}}(T)$. The ID of a given table may appear several times in C_q^2 , with different strings x_T , e.g., if in a table titled “Student population”, two header cells contain “college students” and “high school students”, and the question is about “students”, the table is returned by the L_t lookup together with its title, and twice by the L_h lookup, with each of the relevant header cells.
5. For each table T whose ID appears in some C_q^2 pairs, we **score** T by **summing up** similarities between $\bar{q}$ and $\bar{x}_T$, for each string x_T that leads to an occurrence of T in C_q^2 .

5.4.2 Relaxation Mechanism

We now show how to handle the case $C_q^1 = \emptyset$, i.e., if no table matches the stated question’s location and time. First, this may happen if the look-ups in I_S and/or I_T yield no result. For instance, schoolchildren are not counted at the granularity of a village. Second, some tables may match s_q , while other tables match t_q , but none matches both.

What can be done in such cases? $\mathcal{D}$ may hold facts that answer the question at a *different space or time granularity*. For instance, instead of the student count for a village, a table T' may have the count for the department containing the village. In other cases, the desired facts are *absent* from $\mathcal{D}$ even at different granularities, but *close* information exists. For instance, $\mathcal{D}$ may have no facts about students in (FR, 2025), because the year is not over yet, but it may have table T'' about students in (FR, 2024). Such tables T', T'' can be seen as best-effort results that come close to the user question even if they do not exactly match it. To find tables such as T' and T'' , we *relax* the question as follows.

6. Using the spatial hierarchy H_S , we gather all *neighbors of s_q (the query location) at distance 1* in a set denoted $(s_q)^1$, and similarly the neighbors of t_q at distance 1 in H_T , the time hierarchy, in the set $(t_q)^1$.
7. We repeat the above process (steps (1)–(2)) with (s_q, t^1) for each $t^1 \in (t_q)^1$: this corresponds to keeping s_q unchanged and relaxing t_q . In parallel, we try it also with (s^1, t_q) for each $s^1 \in (s_q)^1$ (relaxing s_q and keeping t_q fixed). One or both of these partial relaxations may lead to relevant tables, which we score as explained at step

- (5) above, subtracting from the score a fixed **penalty** p (a constant empirically determined).
8. If first-level neighbors of s_q, t_q yield no results, we try all combinations (s, t) where $s \in (s_q)^1$ and $t \in (t_q)^1$. This corresponds to simultaneously relaxing s_q and t_q at a distance of 1; results obtained in this way are scored with a penalty $2 \cdot p$.
 9. We repeat the process by relaxing further, e.g., $(s_q)^2$ contains the neighbors of s_q in S at distance 2, etc., increasing the penalty accordingly, until N results have been found, or all neighbors have been explored.

Note that we do not relax $\bar{q}$, the question stripped of the location and time, *explicitly*, but *implicitly*: this is exactly what the approximate nearest neighbor index over $\varphi(\bar{q})$ does (find the closest existing matches). Further, the actual algorithm has two more significant aspects:

- If q specifies no s_q and no t_q , at step (1), we take the *default location of the statistics provider* P (recall Section 5.3.2) as s_q , and the current year as t_q . If $\mathcal{D}$ holds tables from several providers $P_1, \dots, P_m$, we answer q separately over each $\mathcal{D}$ subset produced by one provider, with the respective default location as s_q .
- At step (3), it is important to *reject ANN lookup results that are too far from $\varphi(\bar{q})$* . If we accept “too distant” matches, we might return a table with the exact s_q, t_q but semantically far from $\bar{q}$, which users would consider irrelevant. To avoid this, at step (3), when the similarity between an ANN look-up result and $\varphi(\bar{q})$ is below a **threshold** θ , we discard the result. If, as a consequence, C_q^2 has fewer than N entries, we relax the location and time dimensions (run steps (6) to (9) above). This ensures the algorithm returns tables pertinent for the meaningful, semantic-rich part of the question $\bar{q}$, while remaining as close as possible in the time and space dimensions.

5.4.3 Best-Effort Table Question Answering

While TQA is not the main focus of this chapter, we briefly explain how STAR may also solve it in order to reduce the users’ effort, by showing them the most relevant facts. For each table T in an STD answer computed as above, based on STAR’s knowledge of the table structure, and knowing the strings $\bar{x}\bar{T}$ that brought table T in the result, STAR shows to users, next to a link to T , also *the most specific fact(s) of T relevant for q* . Thus, if T has both a row and a column header pertinent for q , we show the cell at the intersection of the respective row and column; if only a row or column was pertinent, we show (a bounded portion of) that row or column, together with the relevant header. We need to bound this because it may be very large. If only T ’s title matched q , STAR simply returns a link to T .

5.5 Architecture and Configuration

We now present the architectural principles and configuration choices underlying STAR. We first describe its modular and multilingual design (Section 5.5.1), and then detail the indexing and retrieval configuration (Section 5.5.2), linking these choices to their empirical validation.

5.5.1 Model Modularity and Multilingual Capabilities

A key strength of STAR’s architecture is its *modular separation of language understanding from retrieval logic*. Unlike end-to-end neural systems that embed the representation model into learned parameters, STAR cleanly separates the core retrieval logic (spatial/temporal indexing, relaxation mechanism, scoring) from the embedding function used for text representation. The embedding function is used only during indexing and retrieval, and is never integrated into trainable weights, allowing it to be replaced with any fixed-dimensional model without modifying the core algorithm or retraining. This design provides flexibility across languages, domains, and changing corpora.

STAR’s multilingual capacity stems from three architectural choices. First, *spatial and temporal processing is language-agnostic*: location names are matched against standardized hierarchies, and date formats are recognized using general patterns. Second, *STAR can leverage any mono- or multilingual embedding model* to represent text, automatically supporting queries and tables in different (and potentially multiple) languages. Third, *separate indexes can be built for different languages to optimize retrieval*. We evaluated this capability in two scenarios:

- (i) at the national scale, we indexed *INSEE* tables in French using finer-grained spatial hierarchies, with queries in French (experiments presented in [16]); and
- (ii) at a regional and international scale, we indexed *Eurostat* tables covering European countries and occasionally other regions, using coarser spatial hierarchies; most tables are in English, with some fragments in other languages, and queries in English (see Section 5.6.1).

In both cases, multilingual embeddings were used (SBERT, PEARL; further described in Section 5.6.2), highlighting STAR’s ability to combine language-agnostic reasoning with cross-lingual representations.

The advantages of this approach are clear when comparing STAR to alternative methods, which are described in detail in Section 5.6.2. Systems such as SOLO [175] require training from scratch on question-table-query triples in the target language, a process that is *language-specific, time-consuming, and dependent on the generation of synthetic queries*. BIRDIE [90] similarly relies on generating synthetic queries for each language and retraining an encoder-decoder model end-to-end, with *no opportunity for cross-lingual transfer*. LLM-based systems such as PNEUMA [11] can leverage multilingual capabilities, but incur high API costs, show variable quality across languages, and require careful prompt engineering. In particular, creating prompts that are both semantically

accurate and efficient for multilingual data is inherently difficult, as generic LLMs cannot be directly controlled or adapted without fine-tuning or augmenting them via additional training, which brings the drawbacks of end-to-end neural systems. In contrast, STAR’s design allows new languages to be incorporated simply by providing an appropriate embedding model and, if necessary, a regional spatial hierarchy, with no algorithmic changes or retraining required.

These architectural decisions ensure that STAR scales across languages and domains while maintaining high retrieval quality. For STD, where corpora can be multilingual and continuously evolving, this flexibility is essential for practical and scalable deployment. Importantly, it also enables the inclusion of statistics in less-resourced or rare languages, which may require niche monolingual language models, ensuring comprehensive coverage of all countries, populations, and regions.

5.5.2 Indexing and Retrieval Configuration

We have explained, in Sections 5.3 and 5.4, our configuration choices for both the indexed content and the retrieval strategy. These choices are supported by experimental results reported in earlier STATCHECK and STAR work [13, 16]. We outline these earlier experimental findings here. The space of choices comprised four main parameters, which we present below.

Linguistic Representation Model

As described in Section 5.5.1, STAR allows the user to choose its linguistic representation model in a “plug-and-play” manner. We show in Section 5.6.5 that PEARL embeddings lead to better performance on our *Eurostat* benchmark than the more generic SBERT embeddings. The advantages of these neural embedders compared to more traditional approaches have been demonstrated in [16], which evaluated the pioneering WORD2VEC [133] embedder against SBERT, on a dataset of *INSEE* tables. When tested on natural-language queries, SBERT consistently outperforms WORD2VEC independently of the other three parameters, for both original questions (using the same terminology as the data) and reformulated ones. For original questions, we observe gains of (at least) 25.7% in Mean Reciprocal Rank@10 (MRR) and (at least) 16.0% in Precision@10; for reformulated questions, the gains reach (at least) 14.5% in MRR@10 and 36.4% in Precision@10. These results confirm that recent transformer-based models better capture the semantics of the facts. Purely syntactic approaches such as BM25 fail to represent the linguistic information of the facts accurately, especially when the vocabulary used in queries diverges from that of the indexed tables.

Text Representation

As explained in Section 5.3.2, the linguistic model does not embed the raw textual content directly, but a *stripped* version, in which spatio-temporal information is removed from $f_t(T) \cup f_{\{h\}}(T)$. The goal is for embeddings to capture what the facts are about,

rather than where and when they hold, since location (and, to a lesser extent, temporal) information can bias semantic similarity. Ideally, a query should match facts on the same subject even if they differ in time or space. This design choice is empirically validated in [16], where indexing the original text s is compared to indexing its stripped version $\bar{s}$, both using WORD2VEC. On original questions, indexing $\bar{s}$ instead of s improves MRR@10 by 27.2% and Precision@10 by 93.3%; on reformulated questions, the gains reach 83.3% in MRR@10 and 80.7% in Precision@10. These results highlight the strong impact of separating spatio-temporal information from the textual content used for embedding.

Semantic Indexes

As detailed in Section 5.3.2, STAR relies on two complementary families of indexes: *spatio-temporal indexes*, dedicated to location and time information, and *table representation indexes*, dedicated to the semantic content of tables. The *spatio-temporal indexes* were first explored in [16]. Their construction leverages existing geographic reference systems at different granularities, depending on the characteristics of the indexed datasets. While the geographic references may vary across experimental settings, the underlying principles remain the same. Regarding the *semantic indexing* of tables, Section 5.3.1 formalizes several possible table representation functions and motivates the choice of $f_t(T)$ (table titles) and $f_{\{h\}}(T)$ (set of distinct header strings) as the most semantically informative and robust representations of a table T . Accordingly, two indexes are defined: L_t for titles and L_h for header strings. The impact of alternative indexing strategies for table representations is experimentally investigated in [16], where three configurations were compared:

1. Three indexes: $f_t(T)$ over the table titles, $f_{\{h\}}(T)$ over all distinct header strings, and one for table-level comments when available. Note that the comment index was not kept in STAR’s final version, since comments are often absent from datasets.
2. Three indexes: $f_t(T)$ over the table titles, $f_{ch}(T)$ over the concatenated column headers, and $f_{rh}(T)$ over the concatenated row headers.
3. Two indexes: $f_t(T)$ and $f_{\{h\}}(T)$. This configuration corresponds to the final design adopted (Section 5.3.1).

Configuration 1 matches the setting of the prior [13] and is used as a baseline for assessing STAR’s improvements. It was evaluated using WORD2VEC embeddings, augmented with spatio-temporal information extraction. In contrast, configurations 2 and 3 are evaluated with SBERT embeddings. Quantitatively, configurations 2 and 3 consistently outperform the baseline configuration 1. On original questions, switching from configuration 1 to 2 yields a 32.5% increase in MRR@10 and 14.3% in Precision@10, while configuration 3 yields gains of 37.7% and 43.5%, respectively. On reformulated questions, configuration 2 improves over 1 by 14.9% in MRR@10 and 36.5% in Precision@10, whereas configuration 3 reaches 19.4% and 77.3%. These results, observed consistently across both metrics and both query sets, validate the indexing strategy adopted in Section 5.3.1.

Moreover, the systematic improvements of configuration 3 over 2 highlight the benefit of using the finer-grained header representation $f_{\{h\}}(T)$ rather than separating concatenated column and row headers ($f_{ch}(T)$ and $f_{rh}(T)$). This empirically confirms the design choice discussed in the second and third remarks of Section 5.3.1.

Score Aggregator

Since STAR indexes semantic content at the header granularity (in addition to table titles), a question q using L_t and L_h typically retrieves several textual elements $x_T \in f_t(T) \cup f_{\{h\}}(T)$ associated with the same table T . As described in Section 5.4, the final score of a table is obtained by *aggregating* the similarities between the stripped query $\bar{q}$ and each stripped element $\bar{x}_T$. In [16], two aggregation strategies are compared: *sum* and *maximum*. These two strategies are evaluated while keeping all other parameters fixed (SBERT embedder, stripped textual representations, and same indexes). Although the impact of the aggregation parameter is less pronounced than other ones, the sum consistently outperforms the maximum. On original questions, using summation increases MRR@10 by 10.7% and Precision@10 by 14.8%; on reformulated questions, the gains amount to 3.5% in MRR@10 and 12.2% in Precision@10. Overall, these results support the use of sum as the default score aggregator. They empirically confirm the intuition that a table is more likely to be relevant when more than one of its header cells and/or its title match the query, rather than a single (even strong) signal.

5.6 Experimental Evaluation

We now present our experimental validation of STAR on the STD task. We describe the statistical tables we use in Section 5.6.1, and our baselines in Section 5.6.2. We present the evaluation questions and the methodology used to generate them in Section 5.6.3, then our evaluation metrics in Section 5.6.4. We discuss experiment results in Section 5.6.5, before concluding in Section 5.6.6. Figure 5.2 presents the end-to-end pipeline of the whole experimental process; we discuss it through our presentation.

5.6.1 Dataset of Statistical Tables

We use Open Data statistical tables provided by *Eurostat*, the official statistical office of the European Union, for their varied and rich content. We detail the *Eurostat* table structure to make our discussion self-contained; STAR is not tied to this structure, e.g., [13, 16] use HTML and Excel tables from *INSEE*, which are organized differently.

Each table has $d \geq 1$ dimensions (recall the sample table in Figure 5.1). The *time* dimension is always present. The *location* dimension is also frequent, unless the table is only about a specific region (e.g., all facts are at the EU scale): in such a case, it is in the title. The remaining $d - 1$ dimensions vary depending on the table topic, e.g., demographic attributes (such as age or sex), production types or economic indicators.

To give our TD baselines (see next) the best chance at retrieval, we **transform each statistical table into a relational one** (see Section 5.2.2). We chose *Eurostat* tables

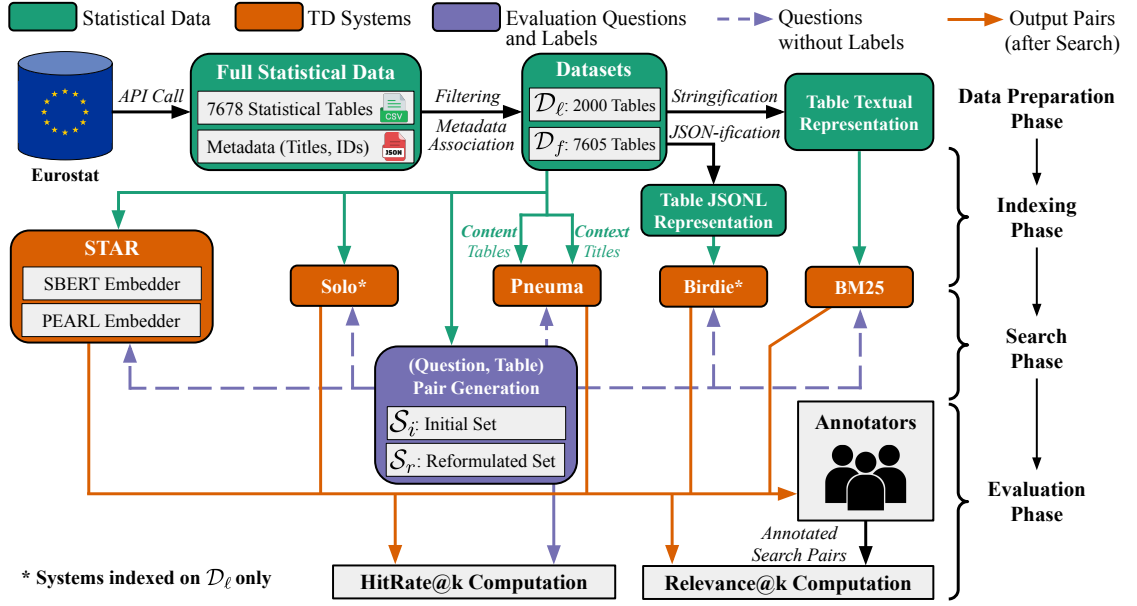


Figure 5.2: End-to-end pipeline of the experimental process.

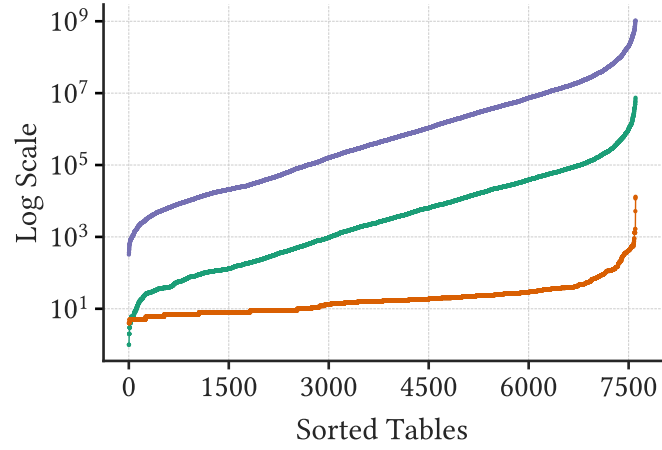
Figure 5.3: Rows, columns, and file sizes for $\mathcal{D}_f$.

Table 5.1: Common Eurostat flags providing additional information about statistics.

Flag	Description
-	Not applicable
:	Not available
b,	Break in time series
d	Definition differs
e	Estimated
f	Forecast
i	Imputed by Eurostat or other sources
m	Missing value; data cannot exist
n	Not significant
p	Provisional
u	Low reliability
0n	Less than half of the unit used, different from real zero

also because they are *ideally suited* for this transformation: they come with *all dimension names explicit in the first row*. This gives **optimal chances to each TD baseline**, that has access to a complete, high-quality relational schema, which they require: in other datasets, such as INSEE [13, 16], this is never the case. In addition to numerical values, Eurostat encodes special cases using *flags*, which convey metadata such as missingness, estimation, provisional status, or methodological notes. Table 5.1 summarizes the most common flags.

As of January 2025, Eurostat’s API gave access to 7 678 tables which we retrieved, together with their *metadata* (title and unique identifier). The raw tables encode dimension values using Eurostat-internal *codes*. We reconstructed human-readable tables by mapping the codes into their NL labels using the Eurostat English code dictionary. We then applied two filtering steps:

- (i) **Duplicate removal:** Tables with different identifiers but identical content are discarded.
- (ii) **Size filtering:** Tables exceeding 1 GB in size are excluded for practical processing reasons (see Section 5.6.2). Only 24 such tables were removed.

We thus obtained a cleaned collection of **7 605** distinct statistical tables, denoted $\mathcal{D}_f$. Since some competitors (as we show below in Section 5.6.5) suffered from *Out-of-Memory* (OOM) failure on $\mathcal{D}_f$, we extracted from $\mathcal{D}_f$ its subset $\mathcal{D}_\ell$, consisting of the 2 000 smallest tables (in terms of file size). Table 5.2 presents the main characteristics of $\mathcal{D}_\ell$ and $\mathcal{D}_f$: the number of tables (**#Tables**), their size in MB (**Size (MB)**), and the mean number of rows and columns per table, with their standard deviation (respectively, **#rows** and **#columns**). Figure 5.3 shows the distribution of the number of rows, columns, and file sizes (in bytes) in $\mathcal{D}_f$, with the tables sorted in increasing order of their size. We can see that about 3 000 tables are larger than 1 MB.

Table 5.2: Main characteristics of the datasets.

Dataset	#Tables	Size (MB)	#rows	#columns
$\mathcal{D}_\ell$	2 000	30.3	111 ($\pm$ 101)	19 ($\pm$ 23)
$\mathcal{D}_f$	7 605	98 521.7	69 077 ($\pm$ 318 706)	41 ($\pm$ 270)

5.6.2 Systems

On the STD task, we evaluate STAR against several baseline retrieval models from the literature. Since we are the first to study STD as a particular flavor of table discovery (TD), we consider three categories of retrieval systems: the classical **BM25** which is *lexical-only*, and a variant we call **BM25+Syn** (see below) augmented with synonyms; two very recent TD systems, **Solo** and **Birdie**, based on *language semantics*, both demonstrating performance superior to prior systems (STAR also mostly belongs to this category); and **Pneuma**, another very recent *hybrid* system, leveraging both lexical and semantic table aspects. The various steps involved in the processing are shown in Figure 5.2. Unless otherwise specified, we used all settings from the original works.

BM25 is a purely lexical retrieval model based on the *Term Frequency–Inverse Document Frequency* (TF–IDF) framework [106]. As it was originally designed for textual documents rather than tabular data, we convert each table into a textual representation by concatenating:

1. The title of the table;
2. The first row of the table, which contains the $d - 1$ non-temporal dimension names and the associated time points;
3. The first $d - 1$ columns of the table (excluding the first row already included above), representing all combinations of values for the $d - 1$ non-temporal dimensions.

We do not serialize the fact values, since as explained in Section 5.3, this is not needed for STD. The text obtained as above from each table is indexed in an ELASTICSEARCH [3] engine, which implements the BM25 algorithm. Given the question q , ELASTICSEARCH returns the IDs of the N most pertinent tables.

BM25+Syn is an extension of **BM25** where each query is extended with synonyms of the verbs and nouns it contains (1 synonym per verb or noun). Observe that enriching *the index* by synonyms would bloat it significantly thus adding synonyms to the query is preferable. To give this baseline its best chance, we performed hyper-parameter tuning over three aspects:

1. *Which words to expand via synonyms?* We tested three options: all words; only nouns and verbs; and nouns, verbs, plus adjectives.
2. *Term weighting.* Since not all words have synonyms (and not the same number), we either *appended synonyms* directly to the query, or *reweighted* all terms so that

the original word and its synonyms shared equal weight (e.g., with 3 synonyms, each of these terms and the original word has weight $\frac{1}{4}$).

3. *Number of synonyms per word.* We tested values from 1 to 10. If a word had less than the target number of synonyms in the thesaurus, we kept all the synonyms.

We tested all combinations of the parameters above; when synonyms were restricted to specific word classes, we applied Part-Of-Speech (POS) tagging with `nltk`’s `averaged_perceptron_tagger_eng` model. We chose the best parameter combination based on the Hit-Rate@10 obtained on our datasets and questions. The best results were obtained with the configuration: *synonyms only for verbs and nouns, appended to the query, with one synonym per word*. We denote this new baseline **BM25+Syn**. Indeed, adding more than one synonym degraded the results. This is expected: the more synonyms added, the more irrelevant noise outweighs potential gains. Interestingly, performance already drops with two synonyms, showing that any benefit from a good synonym is quickly overwhelmed by noise (results are further discussed in Section 5.6.5).

Solo [175] is a semantic data discovery system that employs a self-supervised method to learn meaningful representations of tabular data. Given a set of CSV tables paired with their titles, SOLO embeds and indexes them in a FAISS ANN index [105]. Once the system is trained and the index is built, users can submit natural language questions. SOLO retrieves the top- N most relevant tables from FAISS, then it re-ranks them using a pre-trained teacher retrieval model [98]. We use the authors’ implementation from [174]. To accommodate titles of arbitrary length, we applied a minor modification to the code, as the original version extracts titles from filenames, which are constrained by OS limits.

Pneuma [11] is a *Retrieval-Augmented Generation* (RAG) data discovery system leveraging LLMs for both table representation and retrieval. It indexes tables based on a combination of their *content* (tabular data) and *context* (title). During indexing, an LLM generates concise *schema summaries* and *row-level summaries* for a sample of rows from each table. PNEUMA uses a *hybrid retrieval strategy*, combining *lexical matching* (via BM25) and *semantic similarity*. These two signals are weighted and combined to get an initial ranking of relevant tables. This ranking is refined by the LLM acting as a *ranking judge*, to give a top- k ranked tables. We use the authors’ implementation from [12].

Birdie [90] is an end-to-end and semantic table discovery system that integrates indexing and search into a single encoder–decoder LM. Unlike systems that separately encode tables into vectors, build an index, and then perform retrieval, BIRDIE assigns a unique identifier to tables, and uses an LLM to generate synthetic queries for each table. It then trains the encoder–decoder to map (query, table) pairs directly to their table IDs through supervised learning, using structured JSON representations of tables as input. The parameters of the trained model directly embed the data, such that at search time, it encodes NL queries to generate likely table IDs, from which the top N results are selected. We used the authors’ implementation [89]. Since the fine-tuned LLAMA3 [129] used for query generation is not provided, we ported the system to *OpenAI*’s GPT-3.5

Turbo [142] instead. Converting the *Eurostat* CSV tables into the required JSONL format increased their size by approx. $2.75\times$, resulting in a total size of 267 GB.

STAR with SBERT is the system used in [16], to which we add: the relaxation (steps (6) to (9)) and the rejection of low-relevance ANN search results, described in Section 5.4. Experiments in [16] on a smaller, French-language corpus have shown that the representation functions f_t and $f_{\{h\}}$, together, lead to highest-quality results (as discussed in Section 5.3) thus we also use them here. This system uses the pre-trained SBERT model [151] to represent (embed) linguistic table content (stripped titles and each stripped header cell, as discussed in Section 5.3). Retrieval is then performed as described in Section 5.4. We use *SpaCy* to perform *Named-Entity Recognition* (NER) for extracting location and time information from both table titles and user questions, *QDrant* for the ANN indexes L_t , L_h and *Redis* for the key-value indexes I_S , I_T . STAR’s parameters had the values we determined empirically to work well: $t = 80$ (Section 5.3.2), $\theta = 0.6$ (limit on the *scipy cosine* distance), $p = 0.1$ (Section 5.4).

STAR with PEARL is the same as above, but replacing SBERT with PEARL [35], a more recent embedding model trained with contrastive learning techniques. Unlike SBERT, which is designed for sentence-level semantics, PEARL is optimized for encoding short phrases with little or no context. This makes it particularly well-suited for statistical tables, whose titles and especially header texts are short.

5.6.3 Generation of Evaluation Questions

Our evaluation aims to answer the following questions:

- Q1.** How does each system perform on questions that are syntactically very similar to the tables they refer to?
- Q2.** How does each system perform on paraphrased questions, with the same meaning but different wording?
- Q3.** How does each system scale when facing a large number or volume of tables?

Of course, **Q1.** is of academic interest only, since it allows us to differentiate syntactic from semantic solutions. In practice, users may formulate things differently, thus **Q2.** is much more important. **Q3.** is of practical interest for real-life statistical tables such as those we use. While *Eurostat* tables are quite large, other providers also publish large tables, e.g., in tables published by the *US Census Bureau*, there are on average more than 600 000 cells per table; in the *US National Center for Education Statistics*, more than 85 000, etc. Thus, scalability is important.

To answer **Q1.** and **Q2.**, we respectively build two sets: an initial question set $\mathcal{S}_i$ (on $\mathcal{D}_\ell$ tables) and a paraphrased one $\mathcal{S}_r$. Since $\mathcal{D}_\ell \subset \mathcal{D}_f$, questions generated on $\mathcal{D}_\ell$ remain relevant for $\mathcal{D}_f$, allowing us to study performance as the corpus grows.

$\mathcal{S}_i$ is built semi-automatically, as shown in Figure 5.4 (a zoom on the purple box on Figure 5.2). We randomly sample 100 tables from $\mathcal{D}_\ell$, and prompt GPT-4o [143] to

generate five English question templates per table using only the table’s title, dimension names, and time range. Templates must include placeholders ([]) for dimension values, and we provide the model with an example of five illustrative templates. For each template, we instantiate five concrete questions by sampling a row, and replacing the placeholders with its values. A second LLM call then selects, for each table, the single clearest question (and may rewrite it for clarity, without changing the meaning), yielding one final question per sampled table, i.e., 100 questions.

However, automatic generation alone proves insufficient. Many questions are grammatically incorrect or semantically incoherent, e.g., *How many health graduates in Nurses (EU recognized qualification) were there in Iceland during 2017?* (a correct version could be about “nursing graduates”), or *What was the production volume of Products obtained (1 000 t) milk powder in Germany for 2021?* (a quantity wrongly appears in the question). Also, most questions simply replicate terms from the table titles. The core issues we encountered can be summarized as:

1. **LLMs often fail to produce sound, interesting, and complete questions, especially when the data is complex or multi-dimensional.** As noted in the PNEUMA paper [11], directly prompting LLMs to ask questions over tables often leads to poor results. Instead, they first generate SQL queries over tables, and then translate them into NL using an LLM. However, this still produces many low-quality questions: on their CHICAGO OPEN [46] dataset, we find examples such as: *Which address corresponds to the condition that September is 7626?*, *What is the average zip code for the state of IL?*, or *Which gender category is associated with the total of 97 exits?* (from [10]). Many of their generated questions are of this kind: while grammatically correct, they are semantically incoherent or irrelevant, limiting their usefulness for training or evaluation.
2. **LLM outputs are highly sensitive to prompt wording, which can introduce biases.** This is especially problematic when the generated questions are used to train retrieval models. For example, the prompt used in BIRDIE [90] for query generation specifies that the question should include table context. This leads the LLM to insert the full table title into many questions, leading, on our dataset, to outputs like: *What is the average propensity to consume by household type in 2020 based on the table 'Aggregate propensity to consume by household type - experimental statistics'?* or *What is the trend of passenger cars by unloaded weight from 1996 to 2023 in the table 'Passenger cars by unloaded weight'?*. Such questions offer little value for training, as the table needed to answer them is already given, creating a mismatch between training and real-world usage (users typically do not mention table names in their queries).

To mitigate these problems, we manually curate and **sanitize** each question in $\mathcal{S}_i$, ensuring that the questions are interesting, correct, and likely to be asked by a human being.

$\mathcal{S}_r$ is built by manually paraphrasing each question (**reformulation** in Figure 5.4) in $\mathcal{S}_i$. The goal is to extensively reword each question, so that it largely differs lexically,

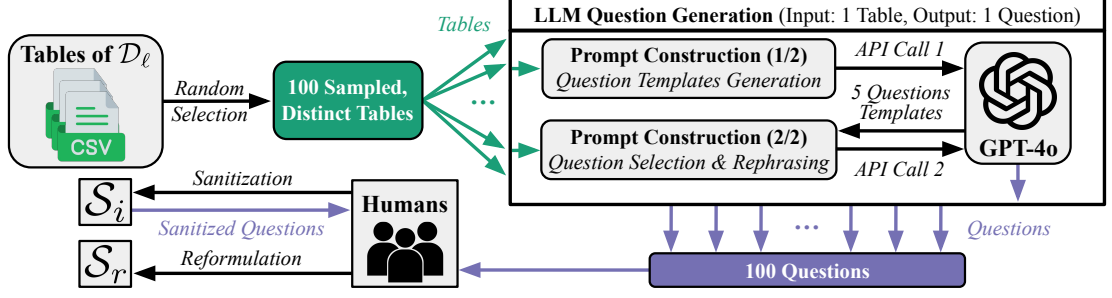


Figure 5.4: (Question, Table) pair generation procedure for $\mathcal{S}_i$ and $\mathcal{S}_r$ (zooming in on the purple box of Figure 5.2).

but still has the exact same semantic meaning. This tests model robustness to lexical variation and shows how over-reliance on cues like table titles can limit generalization and performance.

Together, $\mathcal{S}_i$ and $\mathcal{S}_r$ form two small but high-quality evaluation sets that, to our knowledge, are the first to systematically assess the models’ ability to perform over complex and multi-dimensional tables, beyond surface-level matching.

5.6.4 Evaluation Metrics

We use two evaluation metrics that capture different aspects of retrieval quality.

HitRate@k measures the proportion of questions for which a system successfully retrieves the table from which the question was originally generated, within its top- k ranked results, where $1 \leq k \leq 10$. This metric assesses the model’s ability to accurately and efficiently locate the relevant table. Varying k gives insight into how quickly the correct table is retrieved: the smaller the value of k at which the hit occurs, the better the model’s performance for that metric. HitRate@k (sometimes called Precision@k) is a standard and widely used metric in table retrieval evaluation [11, 90, 175].

Relevance@k measures a system’s ability to return *semantically relevant* tables among its top- k results, where $1 \leq k \leq 5$. This provides a finer-grained evaluation than HitRate@k, as it captures the relevance of the retrieved tables beyond the “originally intended” table. To compute this score, **human annotators** scored each (question, table) pair returned by any system listed in Section 5.6.2 with one of three labels:

- (i) *highly relevant* (score of 2), when the table contains one or more facts that answer the question;
- (ii) *relevant* (score of 1), when the table does not contain the answer but is about the same or a closely related topic;
- (iii) *not relevant* (score of 0), when the table is unrelated or only distantly related to the question.

Table 5.3: Indexing times (in seconds).

Model	$\mathcal{D}_\ell$	$\mathcal{D}_f$
BM25 & BM25+SYN.	13	7 701
SOLO	81 847	OOM
PNEUMA	45 300	532 020
BIRDIE	24 135	OOM
STAR-SBERT	371	45 103
STAR-PEARL	208	44 279

Table 5.4: Search times (in seconds), for all 100 questions.

Model	$(\mathcal{D}_\ell, \mathcal{S}_i)$	$(\mathcal{D}_\ell, \mathcal{S}_r)$	$(\mathcal{D}_f, \mathcal{S}_i)$	$(\mathcal{D}_f, \mathcal{S}_r)$
BM25	0.95	0.91	41.83	45.23
BM25+SYN.	3.39	3.34	47.16	51.47
SOLO	1 267.05	1 143.06	OOM	OOM
PNEUMA	3 793.23	3 680.90	1 770.44	1 814.24
BIRDIE	20.83	20.32	OOM	OOM
STAR-SBERT	36.77	37.66	186.62	190.95
STAR-PEARL	29.68	34.83	106.19	116.68

For a $k \in \{1, \dots, 5\}$, we compute the relevance score of the top- k tables retrieved for each question, aggregated over an evaluation set $\mathcal{S}$ and for a system $\mathcal{M}$:

$$\text{Relevance}_k(\mathcal{D}, \mathcal{S}, \mathcal{M}) = \frac{1}{|\mathcal{S}|} \sum_{q \in \mathcal{S}} \sum_{i=1}^k \text{Score}(\mathcal{M}(q, \mathcal{D})^i, q)$$

where $\mathcal{M}(q, \mathcal{D})^i$ is the i -th table returned by system $\mathcal{M}$ on dataset $\mathcal{D}$ for question q , and $\text{Score}(\cdot, q)$ is the human relevance score for the corresponding (table, question) pair.

5.6.5 Results and Analysis

All experiments were conducted on a server running AlmaLinux 8.10 with 187 GB of RAM and dual Intel Xeon Gold 5218 CPUs, totaling 32 cores (64 threads). It is equipped with an NVIDIA Tesla V100 PCIe 32 GB GPU. Note that STAR does not require a GPU, but the competitors SOLO, PNEUMA and BIRDIE do, thus we run all systems on the same machine.

Computational Efficiency

Table 5.3 presents the **data pre-processing and/or indexing times** for all systems, covering all steps until the system is ready to answer questions, including training where applicable (SOLO and BIRDIE). BM25 is the fastest on both datasets, as a purely lexical

retriever. STAR is the second fastest and leads among hybrid and semantic-only retrievers: it is 116–393 $\times$ faster than others on $\mathcal{D}_\ell$, and over 12 $\times$ faster than PNEUMA on $\mathcal{D}_f$. SOLO and BIRDIE fail to scale to large datasets due to out-of-memory (OOM) errors: SOLO, for $\mathcal{D}_\ell$ alone, used more than 43 GB of RAM (for only 30.3 MB of raw data), and BIRDIE ran OOM during the embedding phase for files greater than 200 MB. PNEUMA, like STAR, avoids RAM issues but requires significantly more time due to its use of an LLM for schema and row summarization.

Table 5.4 presents **question answering time** for each set of 100 questions and all systems, excluding SOLO and BIRDIE on $\mathcal{D}_f$ due to the noted OOM errors. BM25 remains the fastest, for the same reasons as above. On $\mathcal{D}_\ell$, BIRDIE is the second fastest, with search times in the same order of magnitude as STAR. Both outperform PNEUMA and SOLO by 33–180 $\times$ on $\mathcal{D}_\ell$. As BIRDIE fails on $\mathcal{D}_f$, STAR becomes the second fastest on it, 15.5 $\times$ quicker than PNEUMA. Using PEARL instead of SBERT in STAR reduces both indexing and QA time.

Retrieval Performance

We evaluate all models (excluding those with OOM errors during indexing) on both datasets and question sets. Note that SOLO, PNEUMA, and BIRDIE do not always return N (10) results. In SOLO, the number of results is impacted by `retr_top_n`, the upper bound on the number of entries retrieved from the index. BIRDIE, which trains a model to generate (predict) table IDs, sometimes outputs an ID that does not correspond to any table. In our experiments, SOLO returned on average 9.91 tables per question, PNEUMA 9.58, and BIRDIE 9.27. STAR returns 10 tables for each question.

HitRates@k. Figure 5.5 presents the HitRate@k scores ($k \in \{1, \dots, 10\}$) on $\mathcal{D}_\ell$ and $\mathcal{D}_f$, excluding SOLO and BIRDIE on $\mathcal{D}_f$. We first observe that BM25 and BM25+SYN. perform best on $(\mathcal{D}_\ell, \mathcal{S}_i)$, the most lexical-friendly setting, with near-perfect scores for $k=10$. However, they perform poorly in all other evaluations, when relevant tables are surrounded by numerous others or when questions are lexically different from table content. This highlights the need for semantic-based retrieval to ensure generalization. Excluding $(\mathcal{D}_\ell, \mathcal{S}_i)$, STAR-PEARL consistently achieves the best performance across all evaluation sets, slightly outperforming its SBERT variant (ranging from similar results to up to +14%).

To answer **Q1.**, all systems perform well when the questions are lexically close to table content: BM25 and its variant lead, followed by STAR and BIRDIE. SOLO and PNEUMA, while being 10% behind for most k values, still deliver reasonable performance. **Q2.** reveals the limitations of lexical models: BM25 suffers a large performance drop (-45% for $k=10$ from $(\mathcal{D}_\ell, \mathcal{S}_i)$ to $(\mathcal{D}_\ell, \mathcal{S}_r)$), as expected. BM25+SYN. is slightly better than BM25 and competes with BIRDIE at $k=10$, showing relevance of synonym expansion in some cases. BIRDIE’s performance degrades significantly (-30%), reflecting its training bias toward LLM-generated questions that often include full table titles (recall Section 5.6.2). Yet, $\mathcal{S}_r$ questions do not repeat table titles. On $(\mathcal{D}_\ell, \mathcal{S}_r)$, STAR-PEARL is best, outperforming PNEUMA by 11% and SOLO by 15% at $k=10$. STAR achieves strong performance from

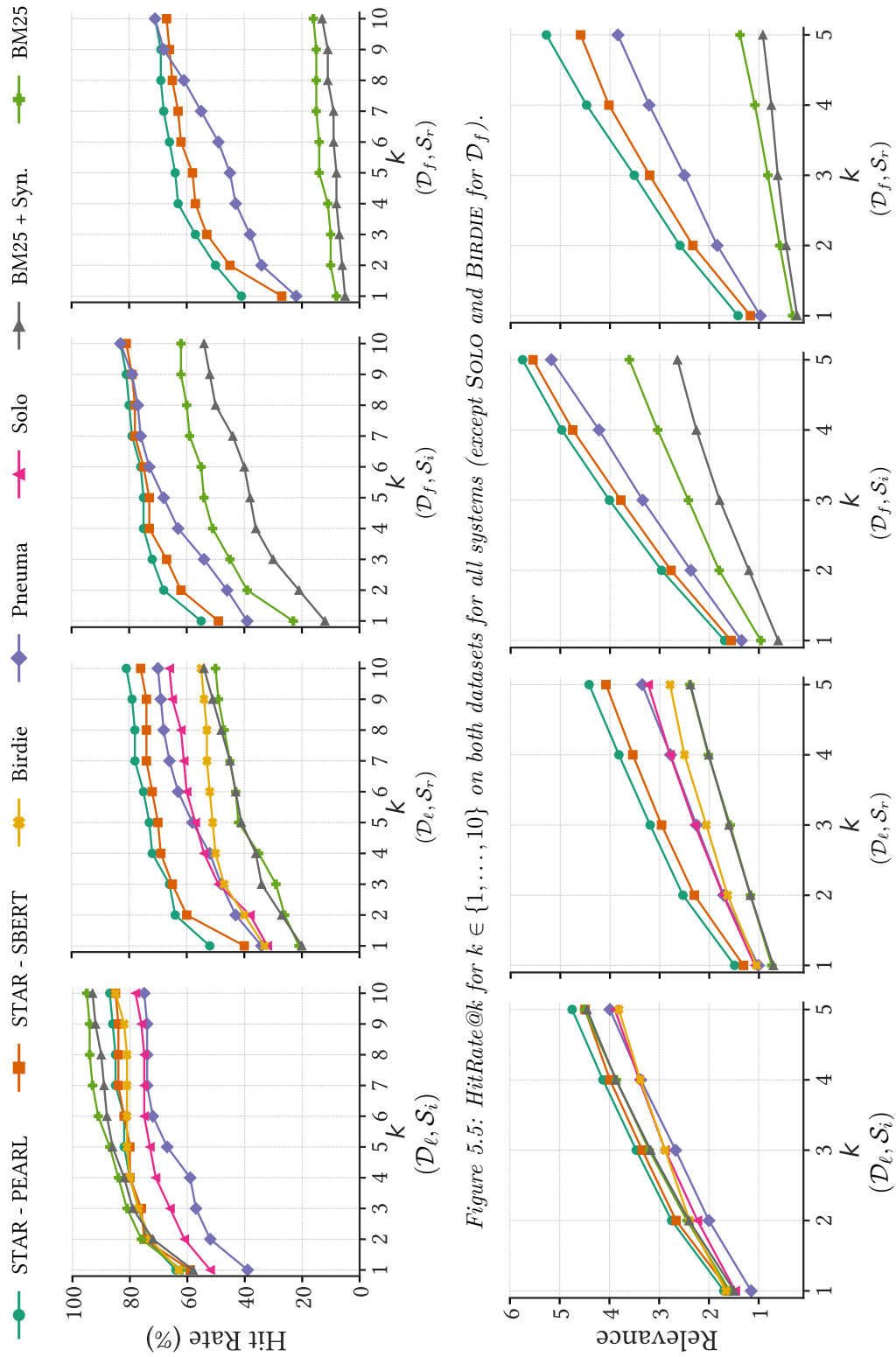


Figure 5.6: Relevance@ k for $k \in \{1, \dots, 5\}$ on both datasets for all systems (except SOLO and BIRDIE for $\mathcal{D}_f$).

the top of the ranking: both PEARL and SBERT variants exceed 60% by $k=2$, whereas others reach it only from $k=6$.

Regarding **Q3**, STAR (with both PEARL and SBERT) and PNEUMA adapt well to larger datasets, PNEUMA even improving from $(\mathcal{D}_\ell, \mathcal{S}_i)$ to $(\mathcal{D}_f, \mathcal{S}_i)$. While they reach comparable performance at $k=10$, STAR obtains strong results quicker: for instance, on $(\mathcal{D}_f, \mathcal{S}_i)$, STAR-PEARL exceeds 70% at $k=3$, while PNEUMA requires $k=6$. In contrast, BM25 fails to scale effectively, dropping by 33% at $k=10$ from $(\mathcal{D}_\ell, \mathcal{S}_i)$ to $(\mathcal{D}_f, \mathcal{S}_i)$. Except on $(\mathcal{D}_\ell, \mathcal{S}_i)$, and on HitRate only, BM25+SYN. performs significantly worse than BM25:

- For questions in $\mathcal{S}_i$, at least one relevant table contains the exact terminology of the query (the golden one). Synonyms therefore add only noise, with no improvement in HitRate and Relevance.
- For questions in $\mathcal{S}_r$, synonyms could in theory help. However, this only works if the expanded terms match the terminology in the relevant tables as opposed to introducing more noise; success here is essentially a matter of luck (depending on whether the reformulated term corresponds exactly to one of the synonyms). Identifying truly useful synonyms would require semantic understanding, which lexical methods lack by definition.

Thus, while adding synonyms to questions could seem a good idea to enhance BM25, it turns out not to help in practice, due to the “noise” added by the synonyms that did not exactly match the tables.

Relevance@k. Figure 5.6 reports Relevance@k ($k \in \{1, \dots, 5\}$) on $\mathcal{D}_\ell$ and $\mathcal{D}_f$, excluding SOLO and BIRDIE on $\mathcal{D}_f$. To compute Relevance@k scores, we manually annotated the top-5 retrieved tables produced by each model, for both datasets and question sets. In total, **4 963 (question, table) pairs were labeled by 9 human annotators**. To assess annotation consistency, 220 pairs (approx. 5%) were randomly selected for double annotation: 24 received differing labels, which shows that annotators agreed **89.1%** of the time, which is considered very high. In cases of disagreement, in the gold standard, we kept the more favorable score.

The trends observed for HitRate@k largely hold for Relevance@k, with a few exceptions. On $(\mathcal{D}_\ell, \mathcal{S}_i)$, BM25 (+ Syn.) no longer lead: STAR-PEARL achieves the highest relevance scores, although BM25 (+ Syn.) remain close. This further shows BM25’s inability to handle lexical variations. On the other evaluation settings $((\mathcal{D}_\ell, \mathcal{S}_r), (\mathcal{D}_f, \mathcal{S}_i), \text{ and } (\mathcal{D}_f, \mathcal{S}_r))$, STAR with both PEARL and SBERT outperforms all other models, with PNEUMA usually following at a distance of less than half a relevance point. Interestingly, while STAR models underwent a performance drop from $\mathcal{D}_\ell$ to $\mathcal{D}_f$ on HitRate@k, they show no comparable drop on Relevance@k: even when they fail to retrieve the exact table the question was generated from, they still retrieve semantically relevant alternatives.

Replacing SBERT with PEARL improves Relevance@k across all four configurations. However, in some cases, we note that STAR-PEARL underperforms. The worst-case occurs for the question “*What was the percentage of GDP allocated to private sector credit flow in non-financial corporations for Slovakia in 2021?*” and its reformulation “*In*

2021, what portion of the GDP was allocated to credit flow for private organizations that are not involved in finance in Slovakia?”, where STAR-SBERT exceeds STAR-PEARL by 4 points on Relevance@5. Similarly, for the question “What fraction of the population in Latvia lacked sanitary amenities in 2011?” on $(\mathcal{D}_\ell, \mathcal{S}_r)$, the difference is 3 points. These examples might suggest that STAR-PEARL underperforms on long or complex questions, since PEARL is trained on short, context-free phrases. Yet, such a conclusion is not supported. Indeed, we encountered complex questions where STAR-PEARL is the only method to perform well, e.g. the first question analyzed in Section 5.6.5. Further supporting this, Figure 5.7 plots, for each model and dataset, the mean Relevance@5 as a function of the number of words in the query, with the numbers above points indicating the number of questions with that score. The trends for STAR-PEARL align with those of other models, rejecting the hypothesis that it struggles with longer or more complex queries.

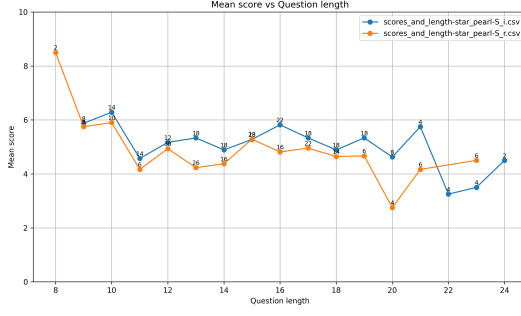
On both quality metrics considered, STAR outperforms all competitors, except BM25 (+ Syn.) which perform better only when questions are phrased almost identically to the data. Even when its performance is close to that of competitors (notably PNEUMA), STAR remains faster in both indexing and search, making it the more practical option overall.

Qualitative Analysis

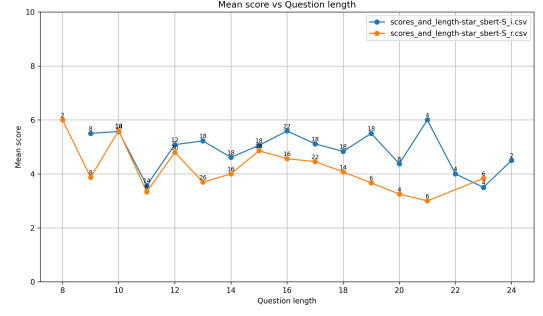
We demonstrate STAR’s relevance by analyzing outputs for two queries over $\mathcal{D}_\ell$ and $\mathcal{D}_f$.

On $\mathcal{D}_\ell$, we consider: “In 2020, what was the median saving rate of employed persons in Bulgaria as a percentage of disposable income?” ($\mathcal{S}_i$) and “In 2020, what was the median rate of savings among people in employment in Bulgaria, as a percentage of available revenue?” ($\mathcal{S}_r$). The question is complex, using economic and demographic information. Lexical baselines perform well on $(\mathcal{D}_\ell, \mathcal{S}_i)$, e.g., BM25 achieves Relevance@5 of 6 via exact overlaps: titles contain “Median saving rate”, and tables “Percentage of disposable income” or “Employed persons”. On $\mathcal{S}_r$, Relevance@5 drops to 0, as only generic words like “median” and “percentage” remain (other new terms do not match relevant tables), retrieving off-topic tables (violence statistics, minimum wage, higher-education researchers). TD systems capture partial semantics but fail to cover the full meaning of the query (Relevance@5 of 0 or 1 on $\mathcal{S}_i$ and $\mathcal{S}_r$). SOLO returns tables with overlapping notions (saving rate, employment, population), showing partial understanding. PNEUMA’s titles are barely related (except “rate” and “people”), though content includes income, percentage, and people. Outputs are nearly identical between $\mathcal{S}_i$ and $\mathcal{S}_r$, showing good semantic proximity detection but poor weighting. BIRDIE outputs tables with titles about saving rate, employment, or compensation (with content partially matching the questions), but no tables connecting the notions, thus results irrelevant for the query. In contrast, STAR-PEARL performs well: for $\mathcal{S}_i$ it overlaps with BM25 on 4 tables, replacing the irrelevant one. For $\mathcal{S}_r$, predictions remain identical (showing strong semantic proximity), yielding Relevance@5 of 7 for both. STAR-SBERT is identical on $\mathcal{S}_i$, but drops on $\mathcal{S}_r$ to a Relevance@5 of 2, keeping only 2 tables: new ones are about minimum wage or unemployment rates, lacking core notions.

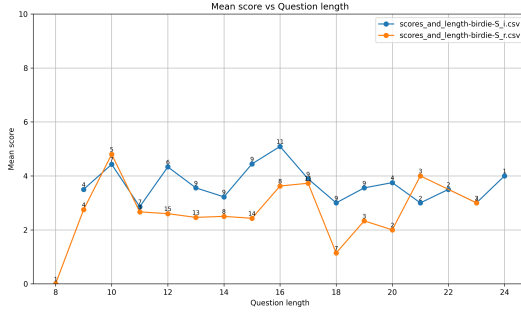
5.6. EXPERIMENTAL EVALUATION



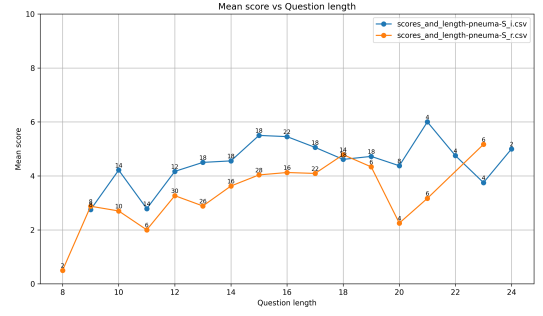
(a) STAR-PEARL



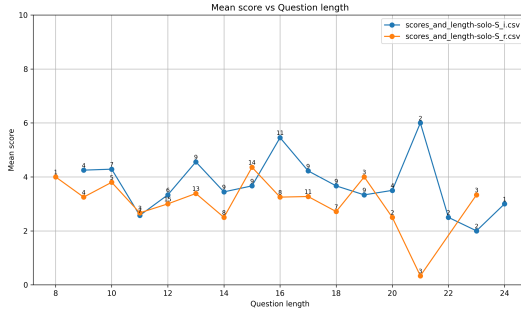
(b) STAR-SBERT



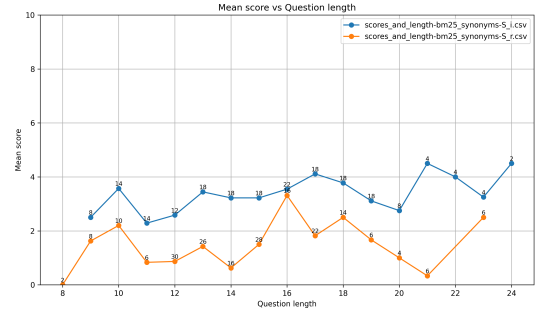
(c) Birdie



(d) Pneuma



(e) Solo



(f) BM25 + Syn.



(g) BM25

Figure 5.7: Mean Relevance@5 score per question length (number of words), for all systems, on both $\mathcal{D}_\ell$ and $\mathcal{D}_f$. Numbers above each point denote the number of questions corresponding to that question length.

On $\mathcal{D}_f$, we study the (much simpler) query “How many deaths were recorded in Belgium in 2021?” ($\mathcal{S}_i$) and “How many people died in Belgium in 2021?” ($\mathcal{S}_r$). The quality of results returned by lexical baselines collapses: BM25 retrieves irrelevant tables (innovation surveys, ICT impact, financing difficulties), leading to a Relevance@5 of 0 on $\mathcal{S}_i$ and $\mathcal{S}_r$. In details: 1. On $\mathcal{S}_i$, the *titles* of the top-5 tables returned by BM25 contain *no word from the query*. The *contents* of the top-5 returned tables have thousands of occurrences of the words “were”, “recorded”, and “Belgium”; the tables themselves are irrelevant. These frequent words probably lead BM25 to miss relevant tables even those containing “death” exactly like the simpler query above. Note that removing duplicates in table headers is not an option because word frequencies are needed by methods such as BM25. 2. On $\mathcal{S}_r$, the *titles* of BM25’s results do not match the questions at all, whereas their *contents* have many occurrences of “people” and “many”.

On the same question, PNEUMA also performs poorly (Relevance@5 of 1 on $\mathcal{S}_i$ and $\mathcal{S}_r$); it returns tables about Belgian railway traffic. In contrast, STAR separately looks up the meaningful metric (here, death), thanks to the separate indexing of location and time. One PNEUMA result partially fits (transport-accident deaths), but lacks the geographic restriction.

In contrast, STAR-PEARL excels (Relevance@5 of 9 on $\mathcal{S}_i$ and $\mathcal{S}_r$), retrieving only highly relevant mortality tables (overall deaths, deaths by sex or cause), that match location and time constraints. STAR-SBERT follows closely (Relevance@5 of 8 on $\mathcal{S}_i$, 7 on $\mathcal{S}_r$); on $\mathcal{S}_i$ its result set is almost identical to that of STAR-PEARL, while on $\mathcal{S}_r$ performance slightly drops, retrieving road accident fatalities or prison capacity data.

5.6.6 Experiment Conclusions

Our experiments demonstrate that STAR outperforms both semantic and hybrid models. It is outperformed by BM25+SYN. on $(\mathcal{D}_\ell, \mathcal{S}_i)$ and HitRate only, when questions match the terminology used in the tables; however, this method collapses on a larger dataset and reformulated questions. In contrast, STAR returns the most semantically pertinent results, with significantly lower indexing and search times than those of semantic competitors, no OOM errors, and robust scalability to large datasets. STAR does not require a GPU, nor does it rely on costly and potentially slow calls to remote LLMs, making it a robust and resource-efficient solution.

5.7 Related Work

We survey prior work across four areas. Section 5.7.1 reviews table discovery methods for relational tables (from early feature-based approaches to recent neural and LLM-based systems) and identifies the structural reasons why they fall short on statistical tables. Section 5.7.2 covers table question answering, which extracts answers from a single (or a few) given table(s). Section 5.7.3 discusses end-to-end question answering pipelines that combine retrieval and answering steps. Finally, Section 5.7.4 positions this chapter within the specific lineage of prior StatCheck iterations on the STD task.

5.7.1 Table Discovery

The closest is **table discovery (TD)**, which seeks to find the relational table(s) most relevant to answer user questions. It is a subtopic of dataset discovery [145], where datasets are tables. Zhang and Balog [185] build a set of representations, syntactic (or statistical) and semantic (based on word embeddings) for queries and tables, and combines with other features into a *learning-to-rank* framework. Semantic representations improve retrieval quality, compared to only using statistical methods (such as BM25), which fails to detect *synonyms* or semantically close words. In [166], the authors transform the query and each table (title and all headers) into a single string, then use BM25 to retrieve the most relevant tables. To rank them, they introduce a set of non-neural and neural features, representing the query and the table based on *Recurrent Neural Networks* (RNN); weight features are learned via a forest of decision trees. In [159], several *modalities* of each table (e.g., description, schema, each row, column) are independently embedded, then a joint representation thereof is learned using *Gated Multimodal Units* (GMU). LLMs use for TD problem is pioneered in [38], which uses BERT. Because only *limited-length strings* can be embedded, their *content selectors* extract, from each table, bounded-length fragments to represent it. On a very large table, this still fails to represent *most* of the content, leading to potentially many missed results (recall the “industry in England” example using f_t in Section 5.3.1). Also, this method needs to *embed each table jointly with the query at query time*, which can be prohibitively slow for datasets of many and/or large tables. The retrieval approach [95] builds a table representation as in [94], a similar one for the query, and retrieves the tables closest to the query.

More recently, such works have been outperformed by the recent systems SOLO [175], PNEUMA [11], and BIRDIE [90] which we presented in Section 5.6.2, and that STAR outperforms on statistical tables. **STAR’s advantage** is due to:

- (i) its built-in understanding of statistical tables, which allows it to index only the tables’ linguistic content;
- (ii) its strategy of representing a table by its title together with each non-empty stripped header cell;
- (iii) its separate indexing and treatment of location and time values, universal dimensions of any measured human or natural activity (thus, of any statistical dataset).

5.7.2 Table Question Answering

NLP question answering over a table (TQA) has been implemented by translating the question into a SQL query, answered by a database holding the tables (see the survey [110]). A text-to-SQL quality benchmark was introduced in [123] and updated since; as of June 2025, it shows an F1 score of 73.84% for the best-performing model, while humans score 92.96%. Others apply neural methods directly on the table, e.g., TAPAS [94]. More recent works, e.g., BINDER [182], DATER [40], or [177, 186] represent tables as strings, given as inputs to an LLM prompted to return the results from the

table. For large tables, this is not feasible. Thus, authors in [124, 164] extract, from large tables, subtables that fit in an LLM’s context, but they still face performance challenges for large tables, and search precision may suffer. **TABLERAG** [36] starts with a *schema retrieval* step matching the question against the database schema (table and attribute names), complete with synthetic information about the values in each column, e.g., the minimum, maximum, and a few sample values. Then, *cell retrieval* narrows down on the cells (in various columns) that are needed in order to compute the question answers. Finally, **TABLERAG** leverages the **REACT** [181] framework, allowing an LLM to interact in a chain-of-thought fashion with tools such as Python, to compute the query answers. **REACTTABLE** [187] also builds upon **REACT**. It answers complex question over a given table, combining LLMs, Python, and SQL tools. Both **TABLERAG** and **REACTTABLE** surpass **BINDER**, **DATER** and other prior works; **TABLERAG** can handle very large tables, using a fixed budget of tokens of interaction with the LLM.

5.7.3 End-to-end Question Answering

End-to-end QA systems aim to answer a question, given a large set of tables. This is typically implemented by a first TD (or *retrieval*) step, which restricts the scope of the search, followed by (*re*)-*ranking*, with a higher-quality but more expensive model. As an example, Herzig et al. [95] use a reader model to extract the answer for a given question using previously retrieved candidate tables. The model scores each candidate, and extracts a suitable answer span from the table. Tables and questions are jointly encoded using **TAPAS** [94]. **SCRUTINIZER** [107, 108] helps human users fact-check *statistical claims*, e.g., “Covid killed more people in France than in Italy”, over relational tables, by interactively composing SQL queries. More generally, many end-to-end QA solutions can be obtained by combining a TD method and one for QA.

All the works mentioned above target relational tables, not multidimensional ones. As described in Section 5.2.2 and experimentally shown, even after converting statistical tables into relational ones, existing TD systems are either unable to handle large tables, or outperformed by **STAR** in terms of efficiency and/or result quality.

5.7.4 Statistical Table Discovery

For the **STD** task, we are only aware of prior versions of **STATCHECK**. In [30], individual facts were indexed by the values of their dimensions, and stored in an RDF database; this did not scale to datasets considered in this chapter. [13, 14] improve over Cao et al. [30] by indexing at the granularity of tables and a more efficient storage of statistics; the demonstration [16] improves result quality over [14] by adopting **SBERT**, and indexing time and locations separately. The present chapter improves over [16] by: (i) pruning low-relevance results and relaxing queries; (ii) replacing **SBERT** with **PEARL** embeddings. Also, we present a *novel and extensive experimental comparison* with recent, high-performing TD systems, which demonstrates **STAR**’s superiority in both efficiency and result quality, and provide a large, novel benchmark.

5.8 Summary

In this chapter, we have formalized the *Statistical Table Discovery* (STD) problem and introduced STAR, a *Space and Time-Aware Statistic Retrieval* method tailored to statistical data. STAR combines structured indexing over spatial and temporal dimensions with semantic embeddings of tabular text, leveraging the PEARL model for improved phrase matching. Relevance is scored along space, time, and query semantics.

Experiments on a large *Eurostat* dataset show that STAR consistently outperforms prior *Table Discovery* systems as well as simpler methods based on BM25, especially when questions do not use the exact words present in the data. We show this on a novel benchmark of curated and reformulated queries, with corresponding relevance scores of tables. Competing systems either fail (OOM) on large datasets or degrade in quality on lexically diverse queries. Even when a competitor is close in accuracy, it is at least $12\times$ slower. Unlike LLM-based systems, STAR requires no GPU and has a lower computational cost.

In future work, we would like to extend STAR to more complex or ambiguous questions, and to refine retrieval through user feedback.

Scalable Spreadsheet Table Understanding

6.1 Introduction

Much of the statistical data published online comes as spreadsheets (XLSX, XLS, ODS, CSV, TSV), continuously updated and reused in downstream pipelines (e.g., Chapter 5). As discussed in Chapter 3, spreadsheets are designed for human interpretation rather than machine consumption: their structure is only partially explicit, conveyed through layout, formatting, merged cells, and spatial proximity, with conventions that vary across publishers, domains, languages, and software. A single sheet rarely contains one cleanly delimited table; it often mixes several data regions with hierarchical headers, titles, footnotes, and metadata, which makes recovering machine-interpretable tables substantially harder than from HTML tables or database exports.

Spreadsheet understanding is therefore a foundational step for structured-data acquisition, as errors in recovering cell roles or table boundaries propagate to every downstream task. Two tasks are central to this problem: *Cell-Type Classification* (CTC) assigns a semantic role (e.g., HEADER, DATA, TITLE) to each cell, and *Table Extraction* (TE) recovers the table boundaries within a sheet. Although both CTC [37, 57, 85, 86, 113, 165, 176] and TE [48, 58, 59, 114, 117, 171] have received continuous attention over the last decade, existing approaches still face important limitations.

1. Recent progress relies on large neural architectures *trained on proprietary corpora* and which are *expensive*. Transformer-based systems such as TUTA [176] and LLM-based approaches such as SPREADSHEETLLM [59] achieve strong results through large-scale pre-training, but at substantial cost (Section 6.5.1 and 6.5.2). Such GPU-intensive systems can be problematic for pipelines processing thousands to millions of spreadsheets, where throughput and operational cost remain primary constraints.
2. Prior work *only partially captures the heterogeneous signals* governing spreadsheet structure. Some methods emphasize semantic embeddings while under-exploiting formatting and geometric regularities; others focus on visual or region-based segmentation while insufficiently modeling long-range dependencies. Yet spreadsheet semantics emerges precisely from the interaction of multiple weak signals, includ-

ing lexical content, formatting consistency, row- and column-level regularities, neighborhood coherence, and global spatial organization.

3. *Reproducibility and evaluation remain problematic.* Several influential systems do not release implementations [57, 58, 114, 117, 165], or rely on proprietary datasets [57–59], preventing direct comparison and independent validation. These issues also make it hard to isolate what drives performance. They are compounded by the lack of public datasets jointly supporting CTC and TE: existing benchmarks are either single-task, structurally limited, or derived from outdated corpora. DECO [116], the only public benchmark covering both tasks, is based on the Enron archive [92] and reflects spreadsheet practices from the early 2000s; it excludes large spreadsheets, lacks multilingual coverage, and omits visually hidden content, making it poorly suited for evaluating modern structured-prediction methods.

Contributions

In this chapter, our contributions are as follows.

- (1) We introduce STATSHEETS, a **publicly available dataset of 737 manually annotated spreadsheet sheets** collected from 14 public organizations across multiple countries, languages, and spreadsheet formats. Unlike prior datasets, STATSHEETS jointly supports both CTC and TE under realistic conditions, including large sheets, multilingual content, heterogeneous layouts, and multiple file formats (Section 6.4).
- (2) We propose an **efficient end-to-end spreadsheet understanding pipeline** combining structured cell-type prediction with deterministic table extraction (Section 6.3). A non-linear classifier predicts per-cell roles from a rich feature space combining lexical, formatting, positional, and row-/column-aware structural descriptors; predictions are then refined through a pairwise CRF enforcing spatial coherence (Section 6.3.1). Table ranges are subsequently recovered by a deterministic 5-stage extraction algorithm operating on the predicted cell-type grid (Section 6.3.2).
- (3) We conduct a **thorough experimental evaluation** of both tasks against non-DL and DL state-of-the-art systems (Section 6.5), showing that:
 - (i) on CTC, our CRF-LIGHTGBM pipeline achieves a Mean File-Macro F_1 of 0.937, within 0.6 percentage points of the Transformer-based TUTA, while requiring substantially lower computational resources and no GPU (Section 6.5.1);
 - (ii) on TE, our deterministic extraction algorithm consistently outperforms generic region-based methods and remains competitive with recent LLM-based systems at orders-of-magnitude lower cost (Section 6.5.2).

These results show that **carefully designed structured models, combining non-linear feature modeling with structured spatial inference, remain highly**

competitive for spreadsheet understanding, particularly in realistic large-scale data-management settings **where scalability, robustness, and interpretability are critical**.

The STATSHEETS dataset, our implementation of SPREADSHEETLLM, and the code to reproduce all experiments are available at [83].

This chapter is organized as follows. Section 6.2 surveys related work on CTC, TE, and end-to-end spreadsheet understanding. Section 6.3 presents our pipeline. Section 6.4 describes the STATSHEETS dataset. Section 6.5 reports the experimental evaluation. Section 6.6 concludes.

6.2 Related Work

We survey prior work on the two core tasks addressed in this chapter: cell-type classification (Section 6.2.1) and table extraction (Section 6.2.2), then we review unified end-to-end approaches (Section 6.2.3) and available datasets for spreadsheet understanding (Section 6.2.4).

6.2.1 Cell-Type Classification in Spreadsheets

Methods for cell-type classification in spreadsheets have evolved from feature-engineered approaches to large pre-trained neural architectures. Early work from Chen and Cafarella [37] use an undirected graphical model combining formatting and content cues to recover annotation-to-data mappings for relational integration, with semi-automatic error repair, emphasizing spatial dependencies but requiring human interaction and targeting relational extraction rather than general-purpose CTC. Koci et al. [113] introduce the first dedicated CTC framework using a Random Forest (RF) over formatting, content, and positional features to classify cells independently. Gonsior et al. [86] later add active learning to reduce annotation costs while preserving this independent-cell setting. Our RF-KOCI baseline follows [113] with a similar feature design. Christodoulakis et al. [44] introduce PYTHEAS, a fuzzy-rule-based system classifying CSV rows into roles such as header, data, and context from column coherency and semantic keywords. Its formulation operates at the row level, assigning a single label to each spreadsheet row. As a consequence, it cannot represent multiple tables appearing side by side (left/right of one another), making precise 2-dimensional TE impossible. It also precludes cell-level CTC, since all cells in a row share the same label, despite spreadsheets commonly mixing roles within rows (e.g., header then data cells). Recent works use DL approaches with semantic and contextual representations. Ghasemi-Gol et al. [85] combine pre-trained cell embeddings, recurrent architectures and formatting features. Dong et al. [57] propose a multi-task framework jointly addressing TE, structural component recognition, and CTC using BERT embeddings, while Sun et al. [165] combine neural embeddings with Probabilistic Soft Logic constraints encoding structural regularities. Neither released code nor datasets. TUTA [176] extends Transformers with structure-aware self-attention capturing spreadsheet layout and semantics, pre-trained on large spreadsheet and Web

table corpora. It substantially outperforms earlier neural methods; we therefore use it as our main DL baseline (Section 6.5.1). FORTAP [39] extends TUTA with formula-aware pre-training objectives, but relies on expert-authored formulas, which limits its generality.

6.2.2 Table Extraction in Spreadsheets

We identify two categories of extraction methods in spreadsheets: *region-based* methods, which extract generic grid structures, and *table-specific* methods, which model table semantics and structure to extract table boundaries.

Region-Based Approaches

Coletta et al. [48] group adjacent non-empty cells into connected components represented by bounding boxes. As a purely structural, learning-free approach targeting generic dense regions rather than tables, it is sensitive to sparsity and cannot distinguish tables from, e.g., metadata blocks. Vitagliano et al. [171] introduce MONDRIAN, which represents spreadsheets as binary images segmented into homogeneous rectangular regions through density-based clustering and compares regions across files via region-layout graphs to identify recurring templates. Although learning-free and unsupervised, both methods operate on generic visual or structural regions rather than tables. Their evaluation in *region-anchored mode* overestimates true TE quality; as shown in Section 6.5.2, performance drops sharply at higher IoU thresholds, highlighting the limits of generic region extraction for precise TE.

Table-Specific Approaches

Other works explicitly focus on predicting table boundaries. Koci et al. [114] propose the graph-based *Remove and Conquer* (RAC) framework, which infers cell layout roles before constructing graphs over spatially related regions to detect table boundaries through curated rules. As one of the earliest spreadsheet pipelines coupling CTC and TE, it is also the prior approach closest to ours. [117] extends RAC with a genetic optimization procedure searching for table boundaries by maximizing a fitness function over predicted cell-type grids. Although effective with accurate CTC predictions, it is sensitive to classification errors and noise [171]. Neither [114] nor [117] releases code, datasets, or trained models, preventing direct comparison. [58] introduces TABLESENSE, a *Convolutional-Neural-Network*-based (CNN-based) end-to-end TE framework formulating extraction as an object-detection problem over featurized cell grids. Trained on the proprietary WEBSHEET10K dataset, it is reported as a strong specialized TE system, though no code, models, or datasets were released; later, SPREADSHEETLLM, from the same Microsoft research group, reports substantially outperforming it. SPREADSHEETLLM [59] encodes spreadsheets into compact token sequences through the multi-stage SHEETCOMPRESSOR pipeline and fine-tunes LLMs for TE and related tasks (details in Section 6.5.2). We include it as a direct competitor using our own implementation, derived from the supplementary materials of [59].

6.2.3 End-to-end Spreadsheet Understanding

Few works address spreadsheet understanding as unified CTC+TE pipelines. [114], later extended by [117], couples the tasks through rule-based extraction but releases neither code nor trained models. Koci et al. [115] demonstrate XLINDY, an interactive Excel add-in for layout inference and table recognition, without publishing an evaluation or sharing the code. Dong et al. [57] jointly address TE, structural component recognition, and CTC via a multi-task *Fully Convolutional Neural Network* (FCNN) with strictly coarse-to-fine dependencies: TE feeds component recognition, which then feeds CTC, without reverse interaction (CTC does not refine TE). In contrast, our pipeline performs CTC based solely on the sheet and derives table boundaries deterministically and exclusively from the predicted cell-type grid. Moreover, Dong et al. [57] rely on a proprietary dataset and release no implementation. Overall, prior end-to-end systems rely on rule-based extraction, proprietary implementations, or coarse-to-fine designs where TE is not influenced by downstream CTC. In contrast, we derive table boundaries solely from CTC predictions while jointly evaluating both tasks under cross-validation.

6.2.4 Datasets for Spreadsheet Understanding

The only publicly available dataset covering both CTC and TE is DECO [116], built from the Enron email archive [92]. DECO is unsuitable in our setting for several reasons. First, it excludes files larger than 5 MB, while large tables are common in public data portals, e.g., *Eurostat* tables regularly exceed hundreds of MB (recall the dataset of Chapter 5). Second, it filters non-CP-1252 content, excluding non-Western languages such as Arabic and Japanese; this introduces semantic and structural biases (e.g., right-to-left layout in Arabic spreadsheets). Third, since it is based on Enron emails (2000–2001), DECO reflects outdated formatting practices that are not representative of current spreadsheets. Finally, annotations only cover visible content: hidden rows/columns (via zero height/width or explicit hide/unhide) are unlabeled, leaving large parts of sheets unannotated and making the dataset incompatible with spatial and sequential methods that require complete grid annotations. Several smaller CTC-only benchmarks also exist. DEEX [60] provides 444 sheets with a six-type label scheme, while SAUS [37] and CIUS [85] follow the same scheme on narrow domains. However, all three provide only CTC labels, lack TE annotations, and have limited linguistic, formatting, and structural diversity; also, DEEX and SAUS are no longer accessible. The proprietary WEBSHEET10K dataset used in TABLESENSE [58], TUTA [176], and SPREADSHEETLLM [59] is also unavailable publicly. A subset of related annotations (VENRON2, VEUSES, VFUSE) released by Dong et al. [58] contains only TE table ranges (no CTC labels) and is restricted to English XLS/XLSX files derived from Enron and EUSES [97]. To facilitate extensive evaluation at scale, we therefore introduce STATSHEETS (Section 6.4), a publicly available dataset addressing these limitations: 737 sheets in six languages across five file formats, with complete cell-level CTC and table-range TE annotations collected from contemporary statistical spreadsheets published by international organizations.

6.3 End-To-End Spreadsheet Table Understanding

We decompose spreadsheet understanding into a two-stage pipeline, illustrated in Figure 6.1. Cell-Type Classification (Section 6.3.1) assigns each cell a label using a non-linear classifier combined with a pairwise CRF enforcing spatial coherence. Table Extraction (Section 6.3.2) takes the resulting label grid and recovers axis-aligned table extents through a deterministic, learning-free algorithm.

6.3.1 Cell-Type Classification

Let $\mathcal{C} = \{\text{EMPTY}, \text{HEADER}, \text{DATA}, \text{TITLE}, \text{OTHER}\}$ be the set of five cell-type classes. We formulate CTC as a structured prediction problem on a 2-dimensional grid, where each cell is a node, and edges connect horizontal and vertical neighbors. Each label $\hat{y}_i \in \mathcal{C}$ should be assigned considering both local features and neighborhood consistency. This motivates a two-stage pipeline in which a non-linear classifier produces per-cell class probabilities that serve as unary potentials for a pairwise CRF which enforces spatial coherence.

Problem 6 (*Cell-Type Classification*)

A spreadsheet region is represented as a 2-dimensional grid $G = (V, E)$, where each node $i \in V$ corresponds to a cell and $E \subseteq V \times V$ is the set of pairs of horizontally or vertically adjacent cells. Given G and some features describing its elements, the CTC problem consists of predicting a class for each cell, or, equivalently, predicting for all cells the label vector $\hat{\mathbf{y}} \in \mathcal{C}^{|V|}$.

Features

Each cell is represented by a 65-dimensional unary feature vector, organized into three groups. The design follows [113], and extends it with additional (richer) formatting attributes and row-/column-level statistical features. Table 6.1 provides a summary.

Group A: Content and positional features (21 features, IDs 1–21).

These features capture intrinsic cell properties and simple local context. *Type indicators* encode whether a cell is empty, numeric (integer/float), string, date, or formula. *Text statistics* describe the string representation: length, counts of digits/letters/spaces/other characters, and binary start-with-letter/start-with-digit flags. *Positional features* include normalized row/column indices within the non-empty region and a boundary flag for sheet borders. *Context features* capture local structure via the number of non-empty 2-dimensional neighbors, an isolation flag (surrounded by empty cells), and two flags distinguishing ODS from XLS/XLSX formats.

Group B: Formatting features (29 features, IDs 22–50).

This group encodes visual and layout attributes from rich spreadsheet formats (XLSX, XLS, ODS). Features include font weight, italic, underline, font size, hashed font name, text/background colors (RGB, normalized to $[0, 1]^3$), a merged-cell flag, and four border

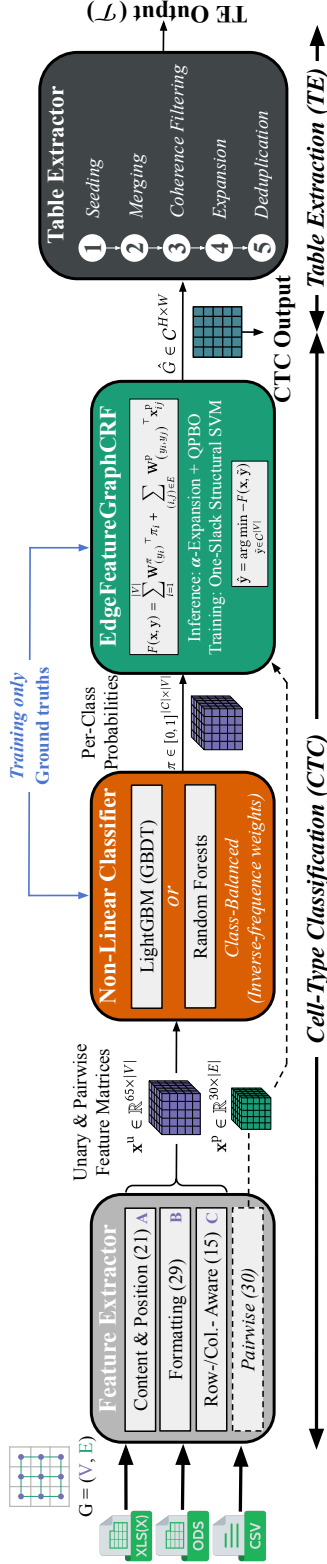


Figure 6.1: End-to-end Cell-Type Classification (CTC) and Table Extraction (TE) pipeline.

Table 6.1: All 65 unary features used in the CTC task, organized into the three groups A, B, and C.

ID	Name	Description	ID	Name	Description
1	IsEmpty	No cell content	8	#CHARS	Character count
2	ISNUM	Value is numeric	9	#DIGITS	Digit count
3	ISSTR	Value is a string	10	#ALPHA	Alphabetic character count
4	ISDATE	Value is a date	11	#SPACES	Space count
5	ISFORMULA	Cell contains a formula	12	#OTHER	Other character count
6	ISINT	Integer value	13	STARTSALPHA	First char. is alphabetic
7	ISFLOAT	Non-integer numeric value	14	STARTSNUM	First char. is numeric
22	BOLD	Font weight is bold	27-29	FontColor	Font color (R, G, B $\in [0, 1]$)
23	ITALIC	Font style is italic	30-32	BgColor	Background col. (R, G, B $\in [0, 1]$)
24	UNDERLINE	Text is underlined	33	ISMERGED	Cell belongs to a merged region
25	FONTSIZE	Font size (pt)	34	BORDERLEFT	Has left border
26	FONTHASH	Stable hash of font name	35	BORDERRIGHT	Has right border
51	ROWNONEMPTYRATIO	Fraction non-empty in row	56	ISALONEINROW	Only non-empty cell in row
52	ROWFRACNUM	Fraction numerical in row	57	COLNONEMPTYRATIO	Fraction non-empty in column
53	ROWFRACSTR	Fraction string in row	58	COLFRACNUM	Fraction numerical in column
54	ROWHOMOG	Type homogeneity in row	59	COLFRACSTR	Fraction string in column
55	ROWISUNIFORM	All non-empty row cells same type	60	COLHOMOG	Type homogeneity in column
			61	ISALONEINCOL	Only non-empty cell in column
			62	COLTYPEOUTLIER	Type differs from col. majority
			63	MERGECOLRATIO	Fraction merged cells in column
			64	MERGEROWRATIO	Fraction merged cells in row
			65	MERGECOLSPAN3+	Merge spans ≥ 3 columns

flags (left, right, top, bottom). Horizontal and vertical cell alignments are one-hot encoded (7 horizontal modes: general/start, right, center, justify, left, fill, distributed; 6 vertical modes: top, center, bottom, distributed/justify, baseline, other), harmonized across file formats. Extraction is implemented separately for XLSX (using *OpenPyXL*), XLS (*xlrld*), and ODS (*ODFPy*), but mapped to a shared 29-dimensional representation. Robust fallbacks handle missing or inconsistent style metadata, common in real-world spreadsheets. For plain-text formats (CSV, TSV), all formatting features are set to default values.

Group C: Row- and column-aware features (15 features, IDs 51–65).

These features capture global row/column-level structure beyond the local 4-neighborhood. For each row, we compute the fraction of non-empty, numeric, and string cells, a type-homogeneity score (majority-type fraction), a type-uniformity flag (whether all non-empty cells share the same type), and an isolation flag (only non-empty cell in the row). Analogous statistics are computed for columns. In addition, a column-type outlier flag indicates whether the cell’s type differs from the column majority, and 3 merge-span statistics capture the fraction of merged cells in the row/column, and a flag indicating whether any column span exceeds three cells.

Pairwise features (30 features).

For each edge (i, j) in the 2D grid, we compute a feature vector describing the relationships between adjacent cells. These features fall in four groups. *Orientation features* encode the relative horizontal and vertical positioning between cells. *Type features* encode agreement and transitions, including whether cells share the same type (empty, numbers, strings, dates, or formula) and type transitions (empty–non-empty, number–string, number–date). *Content difference features* capture absolute differences in text statistics (character and digit counts, jointly starting with a letter, etc.). *Formatting features* measure visual agreement: same font size and name, consistent bold/italic, same font and background colors, shared-border presence, horizontal/vertical alignment differences, and whether both cells are part of merged regions.

Classification Pipeline

The classification pipeline addresses two complementary limitations of cell-wise prediction. First, most Group **A** and **B** features are binary or categorical (BOLD, ISNUM, STARTSALPHA...), and class membership typically depends on their joint interactions rather than individual signals; for example, HEADER cells often combine formatting and row-level homogeneity cues that are not linearly separable. This motivates a *non-linear classifier*, which captures feature conjunctions better than linear models, as confirmed empirically in Section 6.5.1. Second, cell labels exhibit strong spatial dependencies in the grid: a cell may be HEADER partly because it precedes a DATA region, and TITLE cells are often defined by their structural position relative to headers and data blocks. To model these dependencies, we use a *pairwise CRF* to propagate information across adjacent cells and enforce neighborhood consistency.

Step 1: LightGBM classifier.

We train a Gradient-Boosted Decision Tree (GBDT) classifier using LightGBM [111] on the 65 unary features of all training cells. Compared to Random Forests (RF) [27], GBDT iteratively fits trees to correct residual errors of the current ensemble, yielding stronger learners on heterogeneous feature sets combining binary flags, counts, ratios, and color values. Class imbalance (DATA and EMPTY cells dominate most sheets) is handled by reweighting classes inversely to their frequency in the loss. The model outputs a per-class probability vector $\pi_i = \text{LightGBM}(\mathbf{x}_i^u) \in [0, 1]^{|C|}$ for each cell i , where $\mathbf{x}_i^u$ is the 65-dimensional feature vector.

Step 2: EdgeFeatureGraphCRF.

Building on the grid G defined above, we define a first *CRF scoring* function as:

$$F^{\text{LINEAR}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|V|} \mathbf{W}_{(y_i)}^u \top \mathbf{x}_i^u + \sum_{(i,j) \in E} \mathbf{W}_{(y_i, y_j)}^p \top \mathbf{x}_{ij}^p \quad (6.1)$$

where $\mathbf{x}_i^u \in \mathbb{R}^{65}$ and $\mathbf{x}_{ij}^p \in \mathbb{R}^{30}$ are, respectively, the unary and pairwise feature vectors; $\mathbf{W}^u \in \mathbb{R}^{65 \times |C|}$ contains one weight vector per class, and $\mathbf{W}^p \in \mathbb{R}^{30 \times |C|^2}$ one weight vector per ordered label pair. We use the EDGEFEATUREGRAPHCRF of [138], which learns a distinct pairwise weight vector $\mathbf{W}_{(k,l)}^p$ per label pair (k, l) , allowing label-specific pairwise interactions rather than shared across transitions. For instance, a HEADER $\rightarrow$ DATA transition may be encouraged when neighboring cells differ in alignment or font weight, and discouraged otherwise. Eq. (6.1) corresponds to a purely linear CRF using the 65 unary features. To incorporate the non-linear LightGBM classifier, we replace the feature vector $\mathbf{x}_i^u$ with the class-probability vector π_i , yielding:

$$F^{\text{LIGHTGBM}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|V|} \mathbf{W}_{(y_i)}^\pi \top \pi_i + \sum_{(i,j) \in E} \mathbf{W}_{(y_i, y_j)}^p \top \mathbf{x}_{ij}^p \quad (6.2)$$

where $\mathbf{W}^\pi \in \mathbb{R}^{|C| \times |C|}$. In Section 6.5.1, we compare both cases experimentally: Equation (6.1) as CRF-LINEAR, and Equation (6.2) as CRF-LIGHTGBM. CRF-RF follows Equation (6.2) except that the class-probability vector is produced by a Random Forest model.

Maximum A Posteriori (MAP) inference. At inference, the labeling for a spreadsheet $\mathbf{x}$ is obtained by minimizing the *energy* function ($-F$):

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{\tilde{y}} \in \mathcal{C}^{|V|}} -F(\mathbf{x}, \mathbf{\tilde{y}})$$

Exact MAP inference over a general pairwise CRF is NP-hard [158], so we use the α -expansion algorithm [26] via the *PyQPBO* library. It is a move-making algorithm that iteratively proposes a candidate label $\alpha \in \mathcal{C}$ and, for each α , jointly optimizes over all cells the binary decision of keeping the current label or switching to α . This reduces the multi-class problem to a sequence of binary graph-cut subproblems until no move decreases the energy (convergence). Each binary subproblem is solved using Quadratic Pseudo-Boolean Optimization (QPBO) [152], which reformulates binary energy minimization

$(-F(\mathbf{x}, \tilde{\mathbf{y}}))$ as a minimum graph cut: pairwise terms penalizing label disagreement between neighbors (attractive terms) are solved exactly in polynomial time [118], while terms that encourage label disagreement (repulsive terms) are handled approximately [152]. This yields exactly the MAP result when all pairwise interactions are attractive, and a strong approximation otherwise, while remaining tractable for grids of several thousand cells seen in practice [118].

CRF training. CRF parameters $\mathbf{W}$ ($\mathbf{W}^u/\mathbf{W}^\pi$ and $\mathbf{W}^p$) are learned by minimizing the structured hinge loss via the structural SVM framework [169]. Given n training sheets with ground-truth labelings $\{\mathbf{y}_t^*\}_{t=1}^n$, the idea is to find $\mathbf{W}$ such that the score $F(\mathbf{x}_t, \mathbf{y}_t^*)$ exceeds the score of any competing labeling $\bar{\mathbf{y}}_t$ by a margin proportional to how wrong $\bar{\mathbf{y}}_t$ is (measured by the Hamming loss $\Delta(\mathbf{y}_t^*, \bar{\mathbf{y}}_t)$, i.e., the fraction of mislabeled cells). A slack variable $\xi \geq 0$ absorbs margin violations when the constraint cannot be satisfied exactly. In the n -slack variant, one slack per training sheet is introduced, leading to a cutting-plane problem with $O(n/\varepsilon^2)$ iterations to reach ε -accuracy. The one-slack formulation [104] instead introduces a *single* slack variable summed over all training examples:

$$\begin{aligned} \min_{\mathbf{W}, \xi \geq 0} \quad & \frac{1}{2} \|\mathbf{W}\|^2 + \lambda \xi \quad \text{s.t.} \quad \forall (\bar{\mathbf{y}}_t)_{t=1}^n \in \prod_{t=1}^n \mathcal{C}^{|V_t|}, \\ & \frac{1}{n} \sum_t \mathbf{W}^\top [\Psi(\mathbf{x}_t, \mathbf{y}_t^*) - \Psi(\mathbf{x}_t, \bar{\mathbf{y}}_t)] \geq \frac{1}{n} \sum_t \Delta(\mathbf{y}_t^*, \bar{\mathbf{y}}_t) - \xi \end{aligned}$$

where $\Psi(\mathbf{x}_t, \mathbf{y})$ is the joint feature map (concatenation of unary and pairwise sufficient statistics) and $\lambda > 0$ controls the regularization strength. This constraint enforces the margin *on average* across all sheets rather than per sheet, reducing the cutting-plane complexity to $O(1/\varepsilon)$ iterations regardless of n . The trade-off is that per-sheet margin violations can be hidden by other sheets, potentially allowing a small number of poorly labeled sheets to go uncorrected. However, the n -slack variant does not scale to large datasets, and the *average* constraint remains sufficiently representative in our setting where n exceeds 500 sheets per fold (see Section 6.4 and Section 6.5 for dataset information and experimental setup).

Handling class imbalance in the linear CRF baseline. For the linear CRF ablation (*CRF-Linear*, Section 6.5.1) using raw unary features $\mathbf{x}_i^u$ instead of π_i , class imbalance must be handled explicitly. We extend pystruct’s EDGEFEATUREGRAPHCRF with a class-balanced structured hinge loss, weighting each cell by the inverse square root of its class frequency within the training fold. We additionally evaluate an optional *hard-EMPTY clamping* strategy: for cells with `ISEMPTY=1`, a large negative bias is applied to all non-EMPTY unary scores, effectively preventing the CRF from predicting other classes. Analogously, we prevent prediction of EMPTY when `ISEMPTY=0`. These mechanisms are unnecessary in the final CRF-LIGHTGBM, as class imbalance is already handled by LightGBM, and empty cells naturally receive near-zero values to non-EMPTY classes.

6.3.2 Table Extraction

For a spreadsheet region of H rows and W columns, we have $H \times W = |V|$ where V is the cell (vertex) set from Problem 6. Having obtained the cell type prediction $\hat{y}$ as discussed in Section 6.3.1, let $\hat{G} \in \mathcal{C}^{H \times W}$ be the predicted cell-type label grid. Formally:

Problem 7 (Table Extraction)

Given $\hat{G}$, the TE problem consists of predicting a set of non-overlapping, axis-aligned rectangles $\mathcal{T} = \{T_k\}$, where each T_k is the minimal bounding box enclosing a maximal coherent group of HEADER and DATA cells forming a single table in $\hat{G}$.

We solve this problem with a deterministic algorithm that requires no learning. It follows a five-stage pipeline mirroring the heuristics a human annotator would apply when delineating tables on a grid of class labels. We first identify maximal connected regions of HEADER and DATA cells (*seeding*), then iteratively *merge* pairs of regions that are separated only by within-table artifacts (sub-header rows, inter-table titles, or thin empty bands). Next, we *filter out* regions that are too sparse or too incoherent to plausibly be a table; we *expand* each surviving region to recover top, left, and right header bands that may be locally disconnected; finally, we *deduplicate* residual overlapping detections. The method uses a small set of interpretable hyperparameters with default values provided when introduced. They are also summarized in Table 6.2.

Notation and definitions. Rows are indexed by $r \in \{0, \dots, H-1\}$ and columns by $c \in \{0, \dots, W-1\}$. We introduce:

Definition 8 (Ranges and Densities)

A range is a non-empty axis-aligned rectangle $R = [r_0, r_1] \times [c_0, c_1]$ with area $|R| = (r_1 - r_0 + 1)(c_1 - c_0 + 1)$. The Intersection-over-Union and the (asymmetric) containment ratio of two ranges are, respectively:

$$\text{IoU}(R, R') = \frac{|R \cap R'|}{|R \cup R'|} \quad \text{and} \quad \text{cont}(R, R') = \frac{|R \cap R'|}{|R|}.$$

For a class subset $S \subseteq \mathcal{C}$, the S -density of R is:

$$\rho_S(R) = \frac{1}{|R|} \sum_{(r,c) \in R} \mathbb{1}[\hat{G}_{r,c} \in S].$$

For a row r and a column span $[c_0, c_1]$, the absolute and relative header densities are:

$$\eta(r, c_0, c_1) = \rho_{\{\text{HEADER}\}}(r \times [c_0, c_1]) \quad \text{and} \quad \eta^*(r, c_0, c_1) = \frac{\rho_{\{\text{HEADER}\}}(r \times [c_0, c_1])}{\rho_{\mathcal{C} \setminus \{\text{EMPTY}\}}(r \times [c_0, c_1])}$$

with $\eta^* = 0$ by convention when the denominator is 0. The definitions of $\eta(c, r_0, r_1)$ and $\eta^*(c, r_0, r_1)$ are symmetric for columns.

	0	1	2	3	4		0	1	2	3	4	
$\hat{G}$	T	T	T	T	T	→ M =	0	0	0	0	0	TITLE — not in mask
1	H	H	H	H	H		1	1	1	1	1	} Seed a ; $B(a) = [1, 2] \times [0, 4]$
2	H	D	D	D	D		2	1	1	1	1	
3	E	E	E	E	E		3	0	0	0	0	1 separator (sep.) — isolated EMPTY
4	E	H	H	H	H		4	1	1	1	1	Seed b ; $B(b) = [4, 5] \times [0, 4]$
5	H	D	D	D	D		5	1	1	1	1	$\tau = \eta_4 = 0.8 \geq \tau_H \wedge 1 \text{ sep.} \Rightarrow \text{merge } (a, b)$
6	E	E	E	E	E		6	0	0	0	0	} 2 separators (2 sep.) — EMPTY then TITLE
7	T	T	T	T	T		7	0	0	0	0	
8	H	H	H	H	H		8	1	1	1	1	Seed c ; $B(c) = [8, 9] \times [0, 4]$
9	D	D	D	D	D		9	1	1	1	1	$\tau = \eta_8 = 1 \geq \tau_H \wedge 2 \text{ sep.} > 1 \Rightarrow \text{no merge}$

Figure 6.2: Seeding and merging on a two-table grid $\hat{G}$ (■ TITLE, ■ HEADER, ■ DATA, white EMPTY). $M_{r,c} = \mathbb{1}[\hat{G}_{r,c} \in \{H, D\}]$ yields seeds a, b, c . η_r abbreviates $\eta(r, 0, 4)$.

Above, the *absolute* score η measures how much of a column range is covered by HEADER cells. Empty cells reduce η , which is useful when deciding whether a row separates two table candidates: a partial header should not span both tables. The *relative* score η^* , instead, measures the proportion of *non-empty* cells in a row that are headers, ignoring empty padding. This is useful when deciding whether a sparse row acts as a header: rows that span wider than the data block below them often contain blank cells above groups of columns, so η would be diluted by those empty cells even for a genuine header row, whereas η^* remains high.

Definition 9 (Seed Mask and Seeds)

The seed mask of $\hat{G}$ is the binary matrix $M \in \{0, 1\}^{H \times W}$ defined by $M_{r,c} = \mathbb{1}[\hat{G}_{r,c} \in \{\text{HEADER}, \text{DATA}\}]$ for all (r, c) . We define the graph (V_M, E_M) as $V_M = \{(r, c) : M_{r,c} = 1\}$, $E_M = \{\{(r, c), (r', c')\} : |r - r'| + |c - c'| = 1\}$ (horizontal/vertical adjacency). A seed S is a connected component of (V_M, E_M) . We derive its row and column extents $r_{\min}(S)$, $r_{\max}(S)$, $c_{\min}(S)$, $c_{\max}(S)$, and bounding box:

$$B(S) = [r_{\min}(S), r_{\max}(S)] \times [c_{\min}(S), c_{\max}(S)].$$

We write

$$n_D(S) = \sum_{(r,c) \in S} \mathbb{1}[\hat{G}_{r,c} = \text{DATA}]$$

the number of DATA cells in S , and $n_H(S)$ analogously for HEADER. $\text{SEEDS}(\hat{G})$ denotes the set of all seeds.

Algorithm 7: TABLERANGEEXTRACTOR($\hat{G}$)

Input: Cell-type grid $\hat{G} \in \mathcal{C}^{H \times W}$

Output: Set of detected table ranges $\mathcal{T}$

```

1  $\mathcal{S} \leftarrow \text{SEEDS}(\hat{G});$ 
2  $\mathcal{S} \leftarrow \{S \in \mathcal{S} : n_D(S) \geq n_D^{\min} \vee (n_H(S) \geq n_H^{\min} \wedge n_D(S) = 0)\};$ 
3 while  $\exists (S_i, S_j) \in \mathcal{S}^2, i \neq j, \text{ s.t. } \text{SHOULDMERGE}(S_i, S_j, \hat{G})$  do
4   |   Pick any such pair  $(S_i, S_j);$ 
5   |    $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{S_i, S_j\}) \cup \{S_i \cup S_j\};$ 
6  $\mathcal{S} \leftarrow \{S \in \mathcal{S} : \text{COHERENT}(S, \hat{G})\};$ 
7  $\mathcal{T} \leftarrow \{\text{EXPAND}(S, \hat{G}) : S \in \mathcal{S}\};$ 
8 return DEDUPLICATE( $\mathcal{T}$ );
```

The full extraction procedure TABLERANGEEXTRACTOR($\hat{G}$) is given in Algorithm 7: starting from a set of seeds, it repeatedly merges compatible candidates, filters incoherent ones, expands the remaining candidates into table ranges, and removes duplicates.

Stage 1: Seeding. We define a table as the bounding box covering its HEADER and DATA cells (excluding TITLE and OTHER cells). We first compute the connected components of the graph (V_M, E_M) . Each component forms a seed candidate (line 1 of Algorithm 7) that may correspond to a more or less complete fragment of a table, but also include noise or errors. We filter candidate seeds as follows: a seed S is retained only if either $n_D(S) \geq n_D^{\min} = 4$ or if S contains no data cells ($n_D(S) = 0$) and has at least $n_H^{\min} = 8$ HEADER cells. The latter preserves genuine header bands separated from their associated data block by over-segmentation; such seeds can later be recovered during the merging stage. Figure 6.2 illustrates the seeding of a sheet containing two tables, yielding three seeds.

Stage 2: Merging.

Connected components in the raw seed mask are **systematically over-segmented** for two reasons. **1.** Tables may contain *sub-header rows* that split the data region (e.g., section dividers between row groups). These rows are correctly classified as HEADER, but their column-wise structure may differ from that of the top header, leaving them disconnected from rows above and below, splitting a single table into multiple seeds. A simple example is shown in Figure 6.2: seeds a and b are initially separated by an empty row (row 3), even though row 4 acts as a sub-header spanning both regions. Since this sub-header sufficiently overlaps the column structure of both seeds, a merge is triggered. **2.** Adjacent tables on the same sheet are often separated by a thin band of EMPTY rows together with a TITLE row for the next table. This case is illustrated in Figure 6.2: seed c is separated from seed b by an empty row and a title row, and is therefore not merged. On the seed mask, this appears as empty space, which does not reliably distinguish *inter-table separation* from *intra-table structure*. We address both cases (**1.** and **2.**) using

the predicate `SHOULDMERGE` (Algorithm 8), which, given a pair of seeds, determines if they should be fused into one. We call this iteratively until a fixed point is reached (lines 3–6 of Algorithm 7). Each round strictly reduces $|\mathcal{S}|$, thus ensuring convergence. Within `SHOULDMERGE`:

- (i) The vertical case (lines 1–14 of Algorithm 8) applies to seeds with overlapping column spans and a non-empty vertical gap. Let $[c_l, c_r]$ denote the shared column range. We scan the gap rows and count, in the `sep` variable, the rows that contain only `EMPTY` or `TITLE` cells over $[c_l, c_r]$. We consider an intermediate row with density $\eta(r, c_l, c_r) \geq \tau_H$ (default value 0.4) an interior header, indicating a within-table split; in this case, we immediately merge the seeds. At the bottom of the gap, we evaluate $\tau = \eta(r_b, c_l, c_r)$ on the first row r_b of the lower seed. If this row is a header band ($\tau \geq \tau_H$), we merge them only if the gap is minimal ($\text{sep} \leq 1$), corresponding to a sub-header preceding a continuation block. Otherwise, when the lower seed begins with data, we merge whenever the number of separator rows satisfies $\text{sep} < g_v$ (default value 4). This reflects that intra-table interruptions are typically shorter than inter-table whitespace.
- (ii) The horizontal case (lines 15–22 of Algorithm 8) is symmetric and simpler. Let $[r_l, r_r]$ denote the overlapping row span. We count the number `ev` of columns in the gap that are entirely `EMPTY` over $[r_l, r_r]$, and merge the seeds whenever $\text{ev} < g_h$ (default value 2). This suffices because horizontal gaps lack the sub-header and title artifacts that complicate the vertical case.

Stage 3: Coherence filtering.

The merging stage may still produce two types of **invalid candidates**. First, repeated merges can occasionally create large, sparse seeds by connecting distant regions through narrow bridges of `HEADER` or `DATA` cells. Such seeds have large bounding boxes but contain little table-related content. Second, a data block whose header was misclassified may survive the seeding stage despite not corresponding to a valid table. We remove these cases using the predicate `COHERENT` ($S, \hat{G}$). A seed S is rejected if both its data density and header density inside $B(S)$ fall below their respective thresholds, i.e., $\rho_{\{\text{DATA}\}}(B(S)) < \rho_D^{\min}$ (default value 0.05) and $\rho_{\{\text{HEADER}\}}(B(S)) < \rho_H^{\min}$ (default value 0.5). In addition, data-only seeds ($n_H(S) = 0$) are rejected when they contain fewer than $n_D^{\dagger}$ (default value 20) `DATA` cells and no nearby `HEADER` cell within Chebyshev radius r_N (default value 3) of $B(S)$. Header-only seeds ($n_D(S) = 0$) are retained iff $n_H(S) \geq n_H^{\min}$, consistently with the seeding stage. All remaining seeds are accepted.

Stage 4: Expansion.

By construction, a surviving seed S contains the data block of a table together with the header cells that are 4-connected to it in the seed mask. However, **some header cells may remain disconnected** because of empty padding rows, merged-cell rendering artifacts, or isolated cells misclassified as `TITLE` or `OTHER`. Although disconnected in the mask, these cells still belong to the table. We recover them using a directional expansion procedure (Algorithm 9) that grows the seed bounding box upward, leftward,

Algorithm 8: SHOULDMERGE: merge predicate.

Input: Seeds S_i, S_j ; grid $\hat{G}$
Output: Boolean indicating whether to merge S_i and S_j

```

1   $c_l \leftarrow \max(c_{\min}(S_i), c_{\min}(S_j));$ 
2   $c_r \leftarrow \min(c_{\max}(S_i), c_{\max}(S_j));$ 
3  if  $c_l \leq c_r$  then
4       $r_t \leftarrow \min(r_{\max}(S_i), r_{\max}(S_j));$ 
5       $r_b \leftarrow \max(r_{\min}(S_i), r_{\min}(S_j));$ 
6      if  $r_b > r_t + 1$  then
7           $\text{sep} \leftarrow 0;$ 
8          for  $r$  from  $r_t + 1$  to  $r_b - 1$  do
9              if  $\eta(r, c_l, c_r) \geq \tau_H$  then
10                 return true;
11                 if  $\hat{G}_{r,c} \in \{\text{EMPTY}, \text{TITLE}\}$  for every  $c \in [c_l, c_r]$  then
12                      $\text{sep} \leftarrow \text{sep} + 1;$ 
13              $\tau \leftarrow \eta(r_b, c_l, c_r);$ 
14             return  $(\tau \geq \tau_H \wedge \text{sep} \leq 1) \vee (\tau < \tau_H \wedge \text{sep} < g_v);$ 
15  $r_l \leftarrow \max(r_{\min}(S_i), r_{\min}(S_j));$ 
16  $r_r \leftarrow \min(r_{\max}(S_i), r_{\max}(S_j));$ 
17 if  $r_l \leq r_r$  then
18      $c_t \leftarrow \min(c_{\max}(S_i), c_{\max}(S_j));$ 
19      $c_b \leftarrow \max(c_{\min}(S_i), c_{\min}(S_j));$ 
20     if  $c_b > c_t + 1$  then
21          $\text{ev} \leftarrow \left| \left\{ c \in [c_t + 1, c_b - 1] : \forall r \in [r_l, r_r], \hat{G}_{r,c} = \text{EMPTY} \right\} \right|;$ 
22         return  $(\text{ev} < g_h);$ 
23 return false;
    
```

and rightward. We do not expand downward since, in statistical spreadsheets, headers conventionally appear above or beside the data, never below. Expansion continues as long as the next row or column has relative header density $\eta^* \geq \tau_H$ (or τ_H^c for columns), two thresholds set to 0.4. We use relative density η^* rather than absolute density η because header rows frequently contain empty padding above column groups; such rows should still be considered valid headers, even when they do not span the full table width. Finally, the column scans in lines 9 and 14 are evaluated over the seed's original row range $[r_{\min}(S), r_{\max}(S)]$, rather than over the expanded range. This ensures that side-header columns remain anchored to the original data region rather than to newly recovered top headers.

Stage 5: Deduplication.

In rare cases, **two distinct seeds yield overlapping table ranges**, e.g., when a single header band spans two nearby column-aligned data blocks. We resolve these with a Non-Maximum-Suppression (NMS) step: ranges are processed in decreasing order

Algorithm 9: EXPAND: expansion of a seed into a table range.

Input: Seed S , grid $\hat{G}$
Output: Range R representing the detected table

```

1  $(\bar{r}_0, \bar{r}_1, \bar{c}_0, \bar{c}_1) \leftarrow (r_{\min}(S), r_{\max}(S), c_{\min}(S), c_{\max}(S));$ 
2  $r_0 \leftarrow \bar{r}_0;$ 
3  $r \leftarrow \bar{r}_0 - 1;$ 
4 while  $r \geq 0$  and  $\eta^*(r, \bar{c}_0, \bar{c}_1) \geq \tau_H$  do
5    $r_0 \leftarrow r;$ 
6    $r \leftarrow r - 1;$ 
7  $c_0 \leftarrow \bar{c}_0;$ 
8  $c \leftarrow \bar{c}_0 - 1;$ 
9 while  $c \geq 0$  and  $\eta^*(c, \bar{r}_0, \bar{r}_1) \geq \tau_H^c$  do
10   $c_0 \leftarrow c;$ 
11   $c \leftarrow c - 1;$ 
12  $c_1 \leftarrow \bar{c}_1;$ 
13  $c \leftarrow \bar{c}_1 + 1;$ 
14 while  $c < W$  and  $\eta^*(c, \bar{r}_0, \bar{r}_1) \geq \tau_H^c$  do
15   $c_1 \leftarrow c;$ 
16   $c \leftarrow c + 1;$ 
17 return  $[r_0, \bar{r}_1] \times [c_0, c_1];$ 
```

of area, and a range R is accepted unless an already-accepted range R' satisfies either $\text{IoU}(R, R') \geq \tau_{\text{IoU}} = 0.5$ (strong overlap) or $\text{cont}(R, R') \geq \tau_{\text{cont}} = 0.8$ (near-containment). The remaining ranges form the final extraction set $\mathcal{T}$.

6.4 The StatSheets Dataset

Given the limitations of existing datasets for CTC and TE (Section 6.2), we construct and release STATSHEETS [83], a new dataset of spreadsheets annotated for both tasks, originating from statistical datasets published by public organizations. We describe how it was assembled (Section 6.4.1), then detail the annotation protocol and present summary statistics (Section 6.4.2).

6.4.1 Dataset Construction

The dataset is highly heterogeneous, with respect to: (i) the data sources, (ii) the language used in the sheets, (iii) the covered topics, (iv) the sheet sizes, and (v) the file formats. This enables evaluating the robustness of cell-type classifiers and table extractors in highly variable settings, which is critical for downstream applications relying on clean and structured tables, such as fact-checking or data journalism (Chapter 5).

Table 6.3 shows the distribution of sheets across the 14 selected sources, chosen for their geographical diversity, which also increases language diversity. Topic diversity results

Table 6.2: Hyperparameters of the table range extractor.

HP	Default	Stage	Description
$n_D^{\min}$	4	1	Min. DATA cells in a seed
$n_H^{\min}$	8	1,3	Min. HEADER cells in a header-only seed
τ_H	0.4	2,4	Header-density threshold along rows
τ_H^c	0.4	4	Header-density threshold along columns
g_v	4	2	Min. separator rows to keep two seeds apart
g_h	2	2	Min. EMPTY columns to keep two seeds apart
$\rho_D^{\min}$	0.05	3	Min. DATA density of a seed bounding box
$\rho_H^{\min}$	0.5	3	Min. HEADER density of a seed bounding box
$n_D^\dagger$	20	3	Acceptance threshold for header-less seeds
r_N	3	3	Header-neighborhood radius (Chebyshev)
τ_{IoU}	0.5	5	IoU threshold for NMS
τ_{cont}	0.8	5	Containment threshold for NMS

from the broad scope of the selected sources. The collection proceeds as follows. We first retrieve a large pool of spreadsheets from each source using the focused Web crawler described in Chapter 4. Each spreadsheet is then processed to identify candidate sheets likely to contain statistical tables using a simple but effective heuristic: a sheet is retained if at least 50% of its non-empty cells are either purely numerical or predominantly numerical (i.e., a majority of characters are digits, allowing symbols or precision annotations). To limit source imbalance, we retain at most 100 candidate sheets per source and only one sheet per spreadsheet file. The resulting candidates are randomly shuffled and manually annotated, keeping only sheets containing at least one statistical table. Annotation continues until reaching 750 sheets, corresponding to ~ 150 test sheets under the 5-fold cross-validation setup of Section 6.5. A second sanitization pass then corrects annotation errors and removes ambiguous cases, yielding a final dataset of **737 annotated sheets**, written in six languages: English, French, Arabic, Japanese, Spanish, and Icelandic. The file formats are: XLSX (316 sheets, 42.9%), XLS (296, 40.2%), ODS (60, 8.1%), CSV (52, 7.1%), and TSV (13, 1.8%). Rich spreadsheet formats (XLS, XLSX, ODS) provide formatting information; plain-text formats (CSV, TSV) do not.

6.4.2 Annotation Protocol and Dataset Statistics

Each retained sheet is annotated for both the CTC and TE tasks. For **CTC**, we annotate each cell within the minimal bounding rectangle containing all the non-empty cells of the sheet. We use the five labels: EMPTY (0), HEADER (1), DATA (2), TITLE (3), and OTHER (4). HEADER includes column/row headers and sub-headers, which may be adjacent to higher-level headers or appear between DATA cells. Each DATA cell is associated with at least one (and typically several) HEADER cell. TITLE cells describe table content and are usually, though not necessarily, located above it. OTHER covers remaining non-empty cells (e.g., footnotes, annotations, data sources, measurement details), acting as a reject class. Overall, **7 570 354 cells** are annotated for CTC, with

Table 6.3: Distribution of sheets across the 14 sources. The **Ch. 4** column gives the code of the corresponding organization in Table 4.1, when applicable.

Source	Ch. 4	Country	#Sheets
<i>Australian Bureau of Statistics</i>	(<i>ab</i>)	Australia	64
<i>French Ministry of Justice</i>	(<i>ju</i>)	France	64
<i>OECD</i>	(<i>oe</i>)	International	63
<i>US National Center for Education Statistics</i>	(<i>nc</i>)	USA	61
<i>US Bureau of Economic Analysis</i>	(<i>be</i>)	USA	58
<i>Japan’s Ministry of Internal Affairs</i>	(<i>jp</i>)	Japan	58
<i>INSEE</i>	(<i>is</i>)	France	57
<i>General Authority for Statistics</i>	—	Saudi Arabia	57
<i>US Census Bureau</i>	(<i>ce</i>)	USA	54
<i>World Bank</i>	(<i>wo</i>)	International	54
<i>French Ministry of Interior</i>	(<i>in</i>)	France	50
<i>International Labour Organization</i>	(<i>il</i>)	International	48
<i>World Health Organization</i>	(<i>wh</i>)	International	48
<i>French Council for Statistical Information</i>	(<i>cn</i>)	France	1

an average of 10 272 per sheet. Sheet sizes vary considerably, from 27 to 1 542 303 cells, illustrating strong structural variability in real-world spreadsheets. The largest sheet is particularly useful for scalability evaluation (Section 6.5.2). Class distribution is heavily imbalanced: DATA cells account for 62.2% of the annotations, followed by HEADER (20.4%) and EMPTY (17.1%), while TITLE (<0.1%) and OTHER (<0.2%) are rare.

For the **TE** task, we annotate *table ranges*. Since each DATA cell is associated with at least one HEADER, a table is defined as the rectangular bounding box containing all HEADER and DATA cells belonging to it. Consequently, table ranges contain only HEADER and DATA cells, excluding TITLE and OTHER. Moreover, each HEADER or DATA cell can belong to only one table, meaning that overlapping tables are not allowed. In practice, if two regions share HEADER or DATA cells, they are considered part of the same table, potentially with hierarchical headers. These constraints are used as automatic sanitization checks to detect annotation inconsistencies before a manual sanitization. After annotation, sequences of more than two consecutive fully empty rows or columns are compressed to exactly two, to save space and without losing information.

Our 737 sheets lead to **818 annotated tables**. Most sheets (690) contain a single table; 29 contain two, and 18 contain three or more.

6.5 Experimental Evaluation

In this section, we evaluate our end-to-end spreadsheet table understanding pipeline against state-of-the-art competitors from the literature. Sections 6.5.1 and 6.5.2 respectively present the CTC and TE experiments, before concluding in Section 6.5.3. All

Table 6.4: Main characteristics of the CTC systems.

Model	NL	Seq.	Fmt.	Spat.	Sem.	GPU
CRF-LINEAR	✗	✓	✓	✓	✗	✗
RF-KOCI	✓	✗	✓	✗	✗	✗
RF	✓	✗	✓	✓	✗	✗
LIGHTGBM	✓	✗	✓	✓	✗	✗
CRF-RF	✓	✓	✓	✓	✗	✗
CRF-LIGHTGBM	✓	✓	✓	✓	✗	✗
TUTA	✓	✓	✓	✓	✓	✓

systems are evaluated using 5-fold cross-validation. The sheets are shuffled with a fixed random seed and split into five folds of approximately 147 sheets each, yielding an 80/20 train/test split per fold. Depending on the system, the training fold is used for training, fine-tuning, hyperparameter selection, or not used at all. The test fold is shared across tasks for consistency. When a system fails on some sheets, we report results on the subset successfully processed by all systems.

6.5.1 Cell-Type Classification

We evaluate seven systems on the CTC task, ranging from simple feature-based classifiers to a pre-trained transformer. We first describe the compared systems, then present the evaluation metrics and computational efficiency analysis, before discussing the results.

Systems

We compare several CTC systems, classified in Table 6.4 according to their use of spatial awareness (**Spat.**), sequential modeling (**Seq.**), and non-linearity (**NL**). We also indicate whether the systems exploit content-formatting features (**Fmt.**), semantic information embedded in cell text (**Sem.**), and whether they require GPU (**GPU**) for deep learning models.

CRF-Linear is the CRF model described in *Step 2* of Section 6.3.1. It uses $\mathbf{x}^u$ and $\mathbf{x}^p$ for the unary and pairwise components, and thus follows the score function of Eq. (6.1). The model includes the class-balanced loss and EMPTY-case clamping described in Section 6.3.1. This method has sequential and spatial modeling, but is linear. Hyperparameters are selected through a global 5-fold validation procedure to obtain a single configuration for all folds. The configuration uses $\lambda = 0.1$, clamping, a square-root inverse-frequency class-balanced hinge loss, and a batch size of 128. Early stopping is based on the M-FM-F1 score (presented below), evaluated every 100 iterations.

RF-Koci is a baseline based on *Step 1* of Section 6.3.1, where LightGBM is replaced by an RF classifier. It does not use the full unary feature set, but only the non-spatial features, excluding all Group C features as well as feature #19 from Group A. We retain feature #18, the only neighborhood-aware feature from [113]. This baseline is therefore a

non-linear approach comparable to [113], without sequential modeling, and with minimal spatial awareness. Hyperparameters are selected using the same shared 5-fold validation procedure as for CRF-LINEAR. The retained hyperparameters include bootstrap sampling, a maximum depth of 20, `log2` feature selection, a minimum of 7 samples per leaf, a minimum split size of 17, and 481 trees.

RF is similar to RF-KOCI but uses the complete unary feature set ($\mathbf{x}^u$). It is also non-linear. Hyperparameters are selected using the same shared 5-fold validation procedure. The retained configuration uses no bootstrap sampling, a maximum depth of 20, `sqrt` feature selection, a minimum of 6 samples per leaf, a minimum split size of 18, and 574 trees.

LightGBM is a baseline following *Step 1* of Section 6.3.1, based on optimized GBDT. Like RF, it is non-linear, models spatial awareness, but without sequential modeling. Hyperparameters are selected using the same shared 5-fold validation procedure. The retained configuration uses a column subsampling rate of 0.6, a learning rate of 0.06, a maximum depth of 8, 51 minimum child samples, 723 estimators, 111 leaves, ℓ_1 and ℓ_2 regularization of 0.01 and 0.5, respectively, and a subsampling rate of 0.74.

CRF-RF applies a CRF on top of the RF baseline (Section 6.3.1), with LightGBM replaced by RF (non-linear, sequential, with spatial awareness). We train the RF model and the CRF sequentially (the CRF using RF probabilities π as unary inputs, following Eq. (6.2)) on each training fold, keeping the RF hyperparameters fixed and selecting CRF hyperparameters via the same 5-fold validation procedure. The retained CRF configuration uses $\lambda = 10$, no class-balanced loss, no clamping, and a batch size of 128.

CRF-LightGBM combines a CRF model on top of LIGHTGBM, similarly to CRF-RF. The CRF scoring function thus follows Eq. (6.2). It is therefore a non-linear (**NL**) and sequential (**Seq.**) model with spatial awareness (**Spat.**). The same training procedure is used, keeping the LIGHTGBM hyperparameters fixed and reusing the same CRF configuration as CRF-RF.

TUTA [176] (Section 6.2) is pre-trained on large collections of unlabeled Web and spreadsheet tables using three progressive objectives at the token, cell, and table levels. The authors fine-tune TUTA on CTC (which we focus on) and table type classification (TTC), which assumes known table boundaries, i.e., the output of our TE task. TUTA is the only DL baseline we consider; it outperforms other DL-based CTC methods [176]. We use the official implementation [132], adapted to our label set and extended with a preprocessing pipeline to convert raw spreadsheets into the expected model inputs. We fine-tune the last two layers of the 12-layer backbone and the classification head, reinitializing from the pre-trained checkpoint at each fold. The original setup fine-tunes for 4 epochs on approximately 3 500 tables (one million labeled cells), while our dataset contains 818 labeled tables with over seven million labeled cells; we therefore use 20 epochs to match a comparable number of seen tables. We train with a learning rate of 3×10^{-5} and batch size of 16, compared to 8×10^{-6} and batch size of 4 in the original work, following the linear scaling rule.

Metrics

The five cell types (recall $\mathcal{C}$ from Section 6.3.1) are heavily imbalanced across sheets: DATA cells usually dominate, while TITLE and OTHER cells are sparse and unevenly distributed. Standard cell-level accuracy is therefore uninformative, and cell-level macro F1 is distorted by sheet size as larger sheets weigh more in the aggregate score.

Instead, we measure **per-class file-macro F1** (FM-F1), which treats each sheet as a single observation. Let $\mathcal{F}$ be the test sheets and $\mathcal{C}$ the classes. For $f \in \mathcal{F}$ and $c \in \mathcal{C}$, let $F_1(f, c)$ denote the F1 score of class c on sheet f . Denoting $\mathcal{F}_c \subseteq \mathcal{F}$ the set of sheets in which class c is present (in ground truth or predictions), the per-class file-macro F1 is defined as:

$$\text{FM-F1}_c = \frac{1}{|\mathcal{F}_c|} \sum_{f \in \mathcal{F}_c} F_1(f, c).$$

Absent classes from a sheet are excluded from FM-F1_c , avoiding both trivially perfect scores and unfair penalization of rare classes. We summarize system performance with the **mean file-macro F1** (M-FM-F1), the macro-average of FM-F1_c across all classes:

$$\text{M-FM-F1} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{FM-F1}_c.$$

All scores are presented for each of the five folds.

Computational Efficiency

All systems except TUTA run on **Machine 1**, a Dell PowerEdge R730 with two Intel Xeon E5-2640 v4 processors (40 logical cores at 2.40 GHz), 128 GB of DDR4 ECC RAM, and **no GPU**. TUTA needs a GPU and we ran it on **Machine 2**, a Dell PowerEdge R770 with two Intel Xeon 6505P processors (48 logical cores), 256 GB of DDR5 RAM, and one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB VRAM).

To *estimate* equivalent cloud costs, we map both machines to the closest [Amazon EC2 On-Demand](#) instances available in `us-east-1`: `m5.8xlarge` (32 vCPUs, 128 GiB RAM, \$1.536/hr) for Machine 1 (denoted **C**) and `g7e.8xlarge` (32 vCPUs, 256 GiB RAM, 1× NVIDIA RTX PRO 6000 Blackwell, \$5.268/hr) for Machine 2 (denoted **G**). Prices correspond to Linux on-demand rates with per-second billing (60-second minimum), as of May 2026. Cost estimates assume a single instance running for the full experiment duration. Tables 6.5 and 6.6 report runtimes and estimated costs over the full 5-fold cross-validation. File loading and feature extraction (*Load*) are reported separately from model computation (*Train/Infer*), although both contribute to the total billed time. The results show a clear computational gap between the proposed systems and TUTA. Training all six non-DL systems combined costs <\$21, while TUTA alone exceeds \$135 due to extended GPU training. Inference costs are smaller overall, but CPU-only TUTA inference remains much slower and costlier than GPU inference. All non-DL systems complete inference within 15 minutes at negligible cost.

Table 6.5: Training time and estimated cloud cost (5-fold total). $C = m5.8xlarge$ (\$1.536/hr); $G = g7e.8xlarge$ (\$5.268/hr).

Model	Load (s)	Train (s)	Total (h)	Instance	Cost (\$)
RF-KOCl	256.7	31.6	0.08	C	0.12
RF	252.2	120.3	0.10	C	0.16
LIGHTGBM	254.7	177.1	0.12	C	0.18
CRF-LINEAR	252.3	14 658.2	4.14	C	6.36
CRF-RF	250.5	15 848.4	4.47	C	6.87
CRF-LIGHTGBM	254.3	14 952.5	4.22	C	6.49
TUTA	178.5	92 376.0	25.71	G	135.44

Table 6.6: Inference time and estimated cloud cost (5-fold total). $C = m5.8xlarge$ (\$1.536/hr); $G = g7e.8xlarge$ (\$5.268/hr). [†] CPU-only run of TUTA for reference.

Model	Load (s)	Infer (s)	Total (h)	Instance	Cost (\$)
RF-KOCl	540.0	40.0	0.16	C	0.25
RF	555.2	49.0	0.17	C	0.26
LIGHTGBM	541.8	95.9	0.18	C	0.27
CRF-LINEAR	540.1	268.5	0.22	C	0.35
CRF-RF	550.6	300.8	0.24	C	0.36
CRF-LIGHTGBM	542.8	288.7	0.23	C	0.35
TUTA [†]	363.8	30 994.6	8.71	C	13.38
TUTA	203.4	679.1	0.25	G	1.29

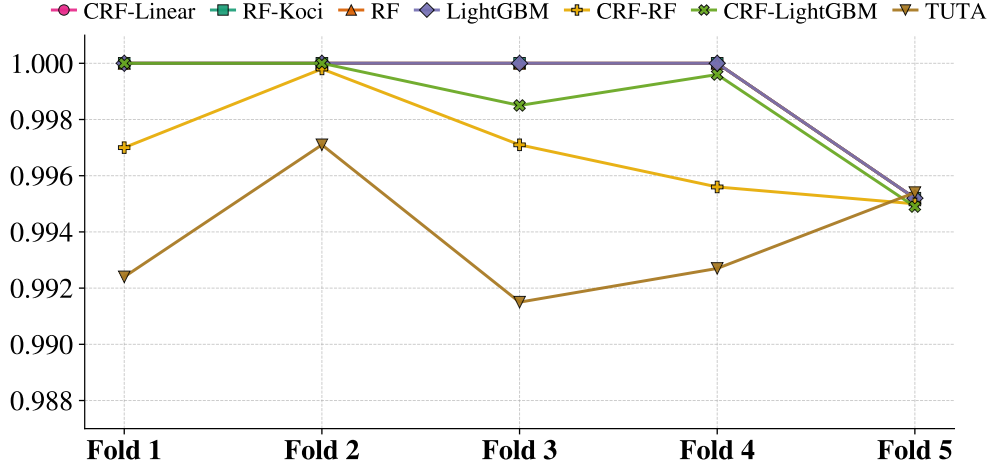


Figure 6.3: Fold-by-fold FM-F1_c trajectories for the EMPTY class.

Results

Figure 6.8 reports the M-FM-F1 scores of all seven systems averaged over the 5 folds. Markers indicate per-fold values, and error bars, the min-max range. Figure 6.9 shows the corresponding per-class FM-F1_c scores using the same color scheme. Figures 6.3 to 6.7 detail the fold-by-fold FM-F1_c trajectories for each of the five cell-type classes individually, allowing direct inspection of cross-fold stability and variance per system and per class.

Effect of spatial awareness, non-linearity, and sequential modeling. We assess the contribution of each component comparing successive bars in Figure 6.8. CRF-LINEAR (sequential, spatial, linear) achieves an M-FM-F1 of 0.894, close to RF-KOCI (0.899), which is non-linear but with minimal spatial awareness and no sequential modeling. Adding full spatial features (RF-KOCI → RF) improves performance by +2.1 *percentage points* (pp) (0.899 → 0.920). Adding sequential modeling through the CRF layer (RF → CRF-RF) yields another +1.1 pp, while replacing RF with LightGBM (CRF-RF → CRF-LIGHTGBM) adds +0.6 pp. Overall, *all three components contribute positively and combine effectively*, empirically validating the usefulness of our modeling. TUTA reaches an M-FM-F1 of 0.943, only +0.6 pp above CRF-LIGHTGBM (0.937).

Per-class analysis. The overall gain from RF-KOCI to CRF-LIGHTGBM is +3.8 pp in M-FM-F1, but improvements are uneven across classes. As shown in Figure 6.9, EMPTY is almost perfectly classified by all systems, while DATA is already saturated (FM-F1 ≈ 0.976–0.981), leaving limited room for improvement on both classes. Gains are larger for harder classes: HEADER improves by +2.8 pp, TITLE by +8.5 pp (0.816 → 0.901), and OTHER by +7.4 pp (0.773 → 0.847). These classes are harder to distinguish using formatting alone and benefit from added structural context provided by spatial and sequential modeling. TUTA performs notably better on OTHER (+4.5 pp), the most semantically dependent class, but slightly worse on TITLE (−1.9 pp), which relies more on structural and formatting cues. The systems perform similarly on EMPTY,

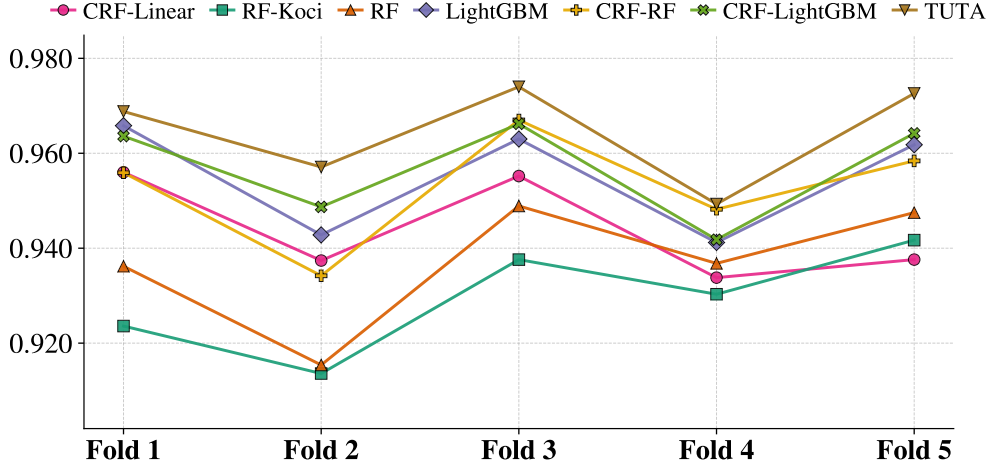


Figure 6.4: Fold-by-fold FM-F1_c trajectories for the HEADER class.

DATA, and HEADER. These results suggest that *semantic modeling mainly benefits the most content-dependent class*, while *structural and non-linear modeling already capture most of the remaining signal*. However, TUTA’s benefits come at a substantially higher computational cost.

6.5.2 Table Extraction

We evaluate TE on the same five folds, measuring how accurately each system recovers the bounding boxes of annotated tables. We first describe the compared systems, then present the evaluation metrics and computational efficiency analysis, before discussing the results.

Systems

We compare six systems for the TE task, including three variants of our deterministic approach from Section 6.3.2 applied on the outputs of different CTC systems, two region-based spreadsheet partitioning methods, and one LLM-based table extractor. Among these systems, only the LLM-based approach requires task-specific TE training. Our methods indirectly rely on learning through the CTC component, but the TE stage itself is fully deterministic. Together with the two region-based baselines, these systems are therefore evaluated directly on each fold without TE-specific training.

TE(RF-Koci) applies the deterministic TABLERANGEEXTRACTOR (Algorithm 7 from Section 6.3.2) to the grids predicted by RF-KOCI. Default hyperparameters from Table 6.2 are used.

TE(CRF-LightGBM) applies TABLERANGEEXTRACTOR to the grids predicted by CRF-LIGHTGBM. Default hyperparameters from Table 6.2 are used.

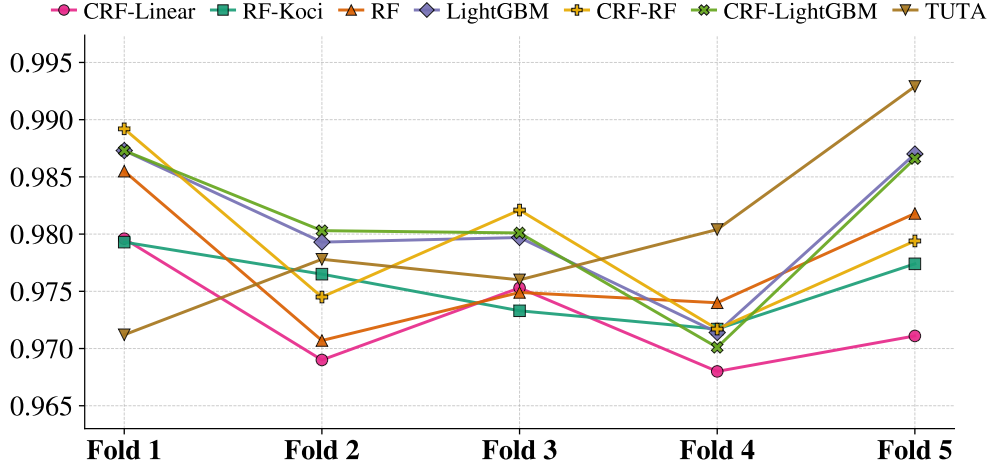


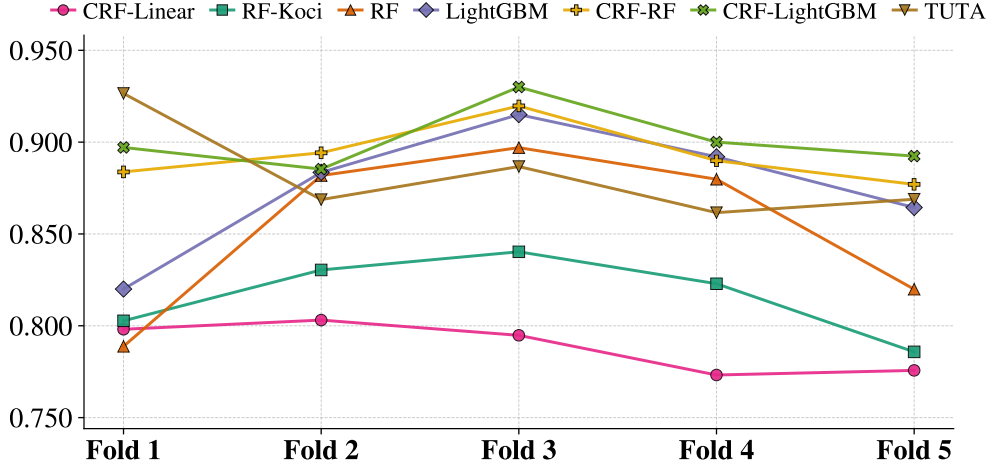
Figure 6.5: Fold-by-fold FM-F1_c trajectories for the DATA class.

TE(Oracle) applies `TABLERANGEEXTRACTOR` to ground-truth cell-type grids, providing an upper bound on TE performance under perfect CTC predictions and isolating the TE stage from CTC errors. Default hyperparameters from Table 6.2 are used.

Connected-Component (CC) is the region-based baseline from [171], following the connected-component approach of [48]. It groups adjacent non-empty cells into connected components and uses their bounding boxes as extracted regions. It requires no training and serves as a lower bound for region-based methods. We use the implementation from the Mondrian repository [172]. As discussed in Section 6.2, CC extracts contiguous *regions* rather than semantic tables, potentially splitting tables or merging them with metadata; it is evaluated in *region-anchored* mode, yielding an optimistic precision estimate.

Mondrian [171] is the unsupervised, training-free approach for spreadsheet region extraction and layout template inference presented in Section 6.2. Like CC, **MONDRIAN** extracts generic regions rather than semantic tables and is therefore evaluated in *region-anchored* mode. We use the authors’ region extraction implementation [172], specifically the *dynamic* variant, which selects the clustering radius independently for each file, as this setting achieved the best results in the original work [172]; also, we noted substantial variability in suitable radii across spreadsheets. **MONDRIAN** only operates on CSV/TSV files and deliberately ignores formatting information.

SpreadsheetLLM (Mistral-7B) [59] is an LLM-based framework for spreadsheet understanding, evaluated on TE and question answering. Its core component, **SHEET-COMPRESSOR**, converts spreadsheet grids into compact token sequences by (i) identifying heterogeneous boundary rows and columns; (ii) encoding the cells located on or near these boundaries (within a fixed neighborhood of k rows/columns) into an inverted-index representation mapping values to compressed cell addresses excluding empty cells; (iii) further aggregating adjacent cells by inferred data type. Following the original paper, we disable (iii) for TE, as the authors report better performance without it. Lacking a complete

Figure 6.6: Fold-by-fold FM-F1_c trajectories for the TITLE class.Table 6.7: Inference time and estimated cloud cost for TE (5-fold total). Train.: training time for SPREADSHEETLLM; loading and CTC inference time for TE variants. [†]Assumes perfect CTC at no cost. [‡]Fold 5 timed out after 7 days.

Model	Train. (s)	Infer. (s)	Total (h)	Inst.	Cost (\$)
SPREADSHEETLLM (train)	33 096.3	—	9.19	G	48.43
SPREADSHEETLLM (infer)	—	1 954.8	0.54	G	2.86
TE(RF-KOCI)	580.0	301.5	0.24	C	0.38
TE(CRF-LIGHTGBM)	831.5	326.3	0.32	C	0.49
TE(ORACLE) [†]	—	355.9	0.10	C	0.15
CC	—	2 723.7	0.76	C	1.16
MONDRIAN [‡]	—	> 653 896	> 181.6	C	> 279.00

implementation, we built one in Python (available in [83]), using the partial C# source code provided in the supplementary material as a starting point for the *SheetCompressor*. The released code omits several filtering functions and requires substantial adaptation to operate over the variety of spreadsheets we have in STATSHEETS. We reimplement all documented components while staying as close as possible to the described behavior. The original work reports better results fine-tuning proprietary LLMs. We choose to fine-tune `mistralai/Mistral-7B-Instruct-v0.2`, which achieves the best performance among the open models reported by the authors, motivated by reproducibility and cost considerations. We follow the hyperparameters from [59, Appendix G]: LoRA adaptation (rank 32, $\alpha = 64$, dropout 0.01) on all linear projection layers, AdamW with learning rate 5×10^{-5} and cosine scheduling, batch size 5 with 8 gradient accumulation steps (effective batch size 40), and a sequence cutoff of 5 800 tokens. Unlike the original setup, which trains for 15 epochs on WEBSHEET10K (~ 10 k sheets), we use early stopping (patience 3) as validation loss stabilizes after one to two epochs on our smaller training folds (~ 550

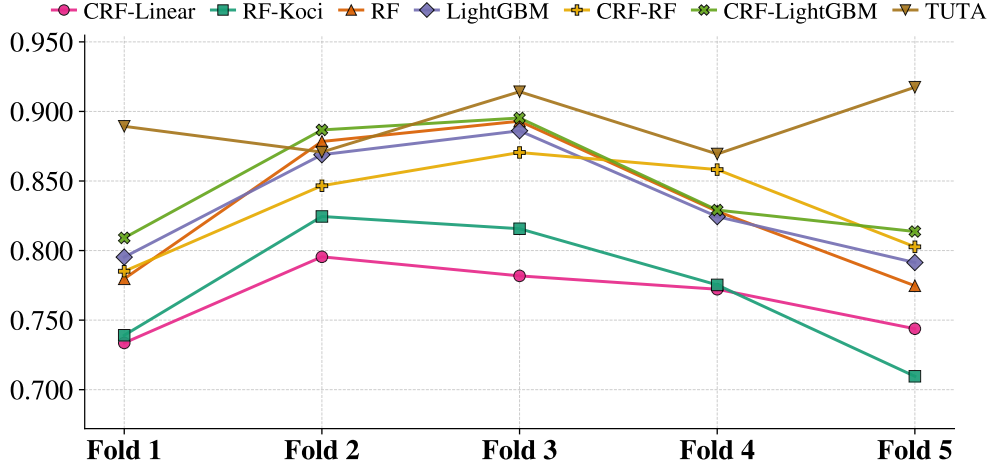


Figure 6.7: Fold-by-fold FM-F1_c trajectories for the *OTHER* class.

sheets). Fine-tuning on our own folds is also necessary to ensure compatibility with our annotation protocol. The model predicts table ranges in Excel notation (e.g., A1:B5). Even after fine-tuning, predictions occasionally deviate from the expected format, producing quoted ranges, malformed coordinates, or isolated cell references. We therefore apply a lightweight post-processing step based on regular expressions to recover valid ranges while rejecting ambiguous outputs. This highlights a practical limitation of generative approaches for structured prediction, where additional recovery logic remains necessary despite task-specific fine-tuning.

Metrics

A table is represented as a rectangular bounding box defined by its row and column extents in the cell grid. For a predicted box $\hat{t}$ and a ground-truth box t , we measure overlap with the standard Intersection over Union (IoU):

$$\text{IoU}(\hat{t}, t) = \frac{|\hat{t} \cap t|}{|\hat{t} \cup t|}$$

where areas are counted in cells.

Precision curve. We report precision as a function of the IoU threshold $\tau \in [0, 1]$. At threshold τ , a matched pair $(\hat{t}, t)$ is counted as a true positive if $\text{IoU}(\hat{t}, t) \geq \tau$, and as a false positive otherwise. Precision at τ is thus:

$$\text{Precision}(\tau) = \frac{|\{(\hat{t}, t) : \text{IoU}(\hat{t}, t) \geq \tau\}|}{N_{\text{pred}}}.$$

We report both **micro** precision (pooling all predictions across sheets) and **macro** precision (averaging per-sheet precision curves), sweeping τ continuously over $[0, 1]$.

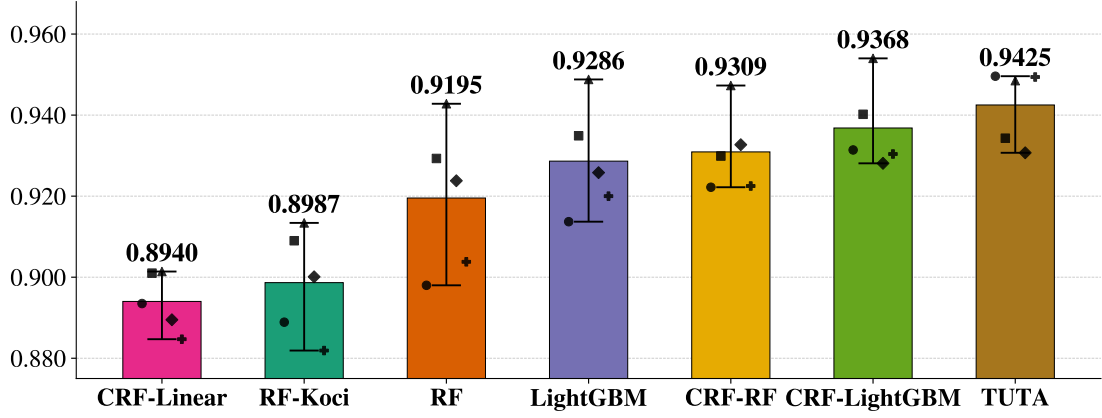


Figure 6.8: Mean File-Macro F_1 per model. Error bars span per-fold min/max; individual fold scores are shown as markers.

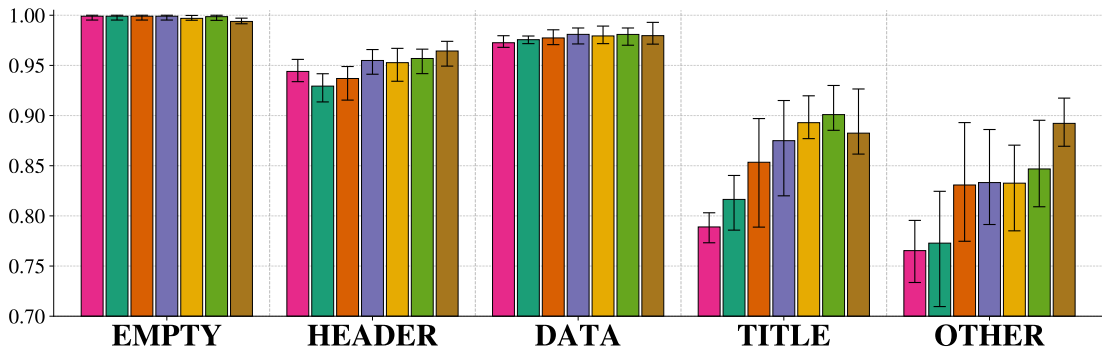


Figure 6.9: Per-class file-macro F_1 breakdown; error bars span per-fold min/max.

Matching. Predicted and ground-truth tables are matched greedily in decreasing order of IoU: each table on either side is matched at most once (tables cannot overlap), and pairs with $\text{IoU} = 0$ are not matched. Unmatched predicted tables count as false positives; unmatched ground-truth tables count as false negatives. Systems that predict tables directly use N_{pred} as the total number of predictions (*table-anchored* mode). For region-based systems (CC and MONDRIAN), evaluated in *region-anchored* mode, N_{pred} is capped at $\min(N_{\text{pred}}, N_{\text{gt}})$ to avoid penalizing over-segmentation that is intrinsic to the region extraction paradigm. This favors region-based methods by discarding excess predictions, yielding an optimistic precision estimate. In both modes, since matching is one-to-one, every predicted table is either matched or unmatched, yielding $\text{Recall}(\tau) = \text{Precision}(\tau) \times N_{\text{pred}}/N_{\text{gt}}$ for the micro curve: recall is a constant rescaling of precision, thus we do not report it.

Computational Efficiency

Experimental infrastructure and cost estimation follow Section 6.5.1, using the same instance mappings: **C** (m5.8xlarge, \$1.536/hr) and **G** (g7e.8xlarge, \$5.268/hr). SPREADSHEETLLM requires a GPU and runs on Machine 2 (**G**); others run on Machine 1 (**C**). Table 6.7 reports runtimes and estimated costs over the full 5-fold cross-validation. Due to the heterogeneous nature of the compared systems, columns have different interpretations. For SPREADSHEETLLM, *Train.* and *Infer.* correspond to fine-tuning and inference. For TE variants, *Train.* reports *loading and CTC inference time* of the underlying CTC model (in Table 6.6), while *Infer.* measures the deterministic TE stage alone. TE(ORACLE) has zero training cost as perfect CTC predictions are assumed (unrealistic upper-bound). For CC and MONDRIAN, only inference time is reported, including file loading. For MONDRIAN, we interrupted fold 5 after seven days (604 800 s) spent processing a spreadsheet with over one million cells. The reported values combine those for folds 1 to 4 (49 095.9 s) with the full timeout duration; they are lower bounds.

The computational cost of TE is dominated by two outliers. MONDRIAN is by far the most expensive system due to the 7-days timeout in fold 5, yielding a lower-bound cost of \$279. Excluding this, total cost is \$53.47, of which \$51.29 comes from SPREADSHEETLLM alone due to GPU fine-tuning. By contrast, our TE variants are inexpensive: TE(RF-KOCl) and TE(CRF-LIGHTGBM) cost \$0.38 and \$0.49 respectively, including the required CTC inference stage, while CC costs \$1.16. Accounting for both CTC training and inference, the full TE(CRF-LIGHTGBM) pipeline amounts to \$6.98 in total. Overall, the deterministic TE pipeline adds little overhead on top of the underlying CTC system and remains substantially cheaper than both GPU-based and unsupervised alternatives.

Results

Figures 6.10 and 6.11 report macro and micro precision as a function of τ for all systems. All methods consistently achieve higher macro than micro precision, indicating over-prediction of tables in larger/more complex spreadsheets, where micro metrics are more sensitive to excess predictions. Figures 6.12 to 6.16 and 6.17 to 6.21 provide fold-by-fold

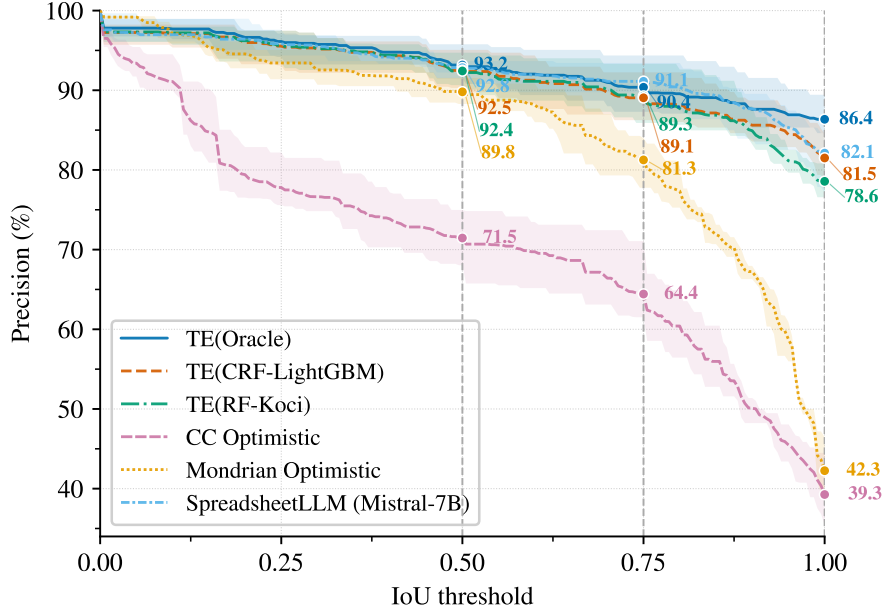


Figure 6.10: Macro-averaged (per-sheet) TE precision. Solid lines show fold means; shaded areas indicate min-max.

breakdowns of macro and micro precision, respectively. Each per-fold figure shows the raw precision curves for every method evaluated on that fold, enabling direct inspection of per-fold behavior and cross-fold consistency beyond the aggregated view. Methods with missing results for a given fold are omitted from the corresponding figure; this only concerns MONDRIAN on fold 5.

Effect of CTC quality on table extraction. The TE variants directly reflect the quality of their underlying CTC systems. Replacing RF-KOCI with CRF-LIGHTGBM improves precision at $\text{IoU} = 1$ by +2.9 pp on macro ($78.6 \rightarrow 81.5\%$) and +3.3 pp on micro ($73.5 \rightarrow 76.8\%$). As shown in Figures 6.10 and 6.11, these gains mainly appear at high IoU thresholds, indicating more accurate boundary recovery rather than better coarse localization. This is consistent with the CTC results of Section 6.5.1: DATA and EMPTY cells are already well classified by all systems, while remaining errors mostly affect boundary-related classes such as TITLE, HEADER, and OTHER. Using perfect CTC predictions, TE(ORACLE) reaches 86.4% macro and 82.3% micro precision at $\text{IoU} = 1$. Relative to TE(CRF-LIGHTGBM), this leaves a margin of +4.9 pp and +5.5 pp respectively, suggesting that CTC quality gains would also translate into better table extraction.

Region-based methods. Despite the optimistic evaluation protocol for CC and MONDRIAN, both region-based approaches remain substantially below TE variants at high IoU thresholds. At $\text{IoU} = 1$, MONDRIAN reaches 42.3% macro and 40.0% micro precision, trailing TE(CRF-LIGHTGBM) by -39.2 pp and -36.8 pp. However, at lower IoU thresholds, it performs comparably to several other systems, indicating that its predicted

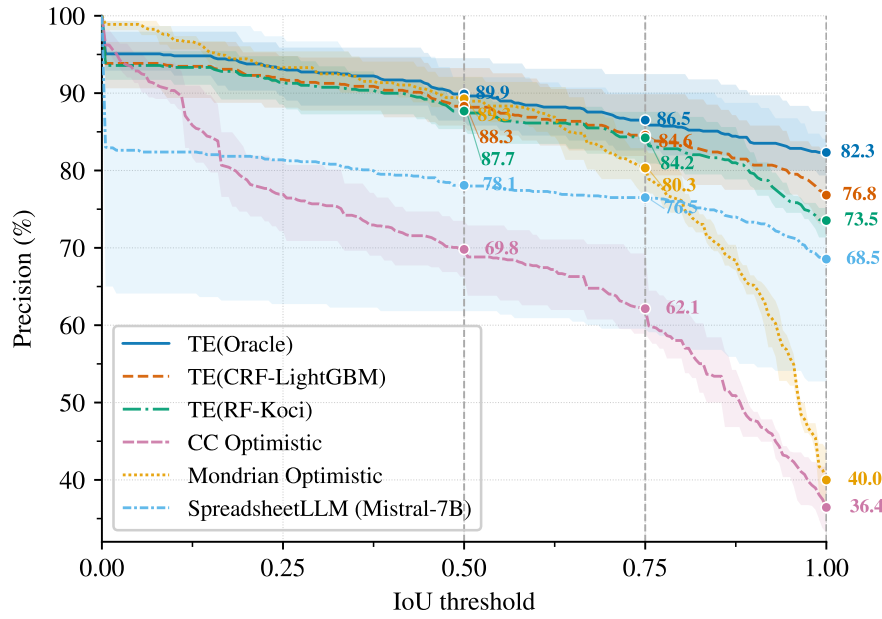


Figure 6.11: Micro-averaged TE precision. Solid lines show fold means; shaded areas indicate min-max.

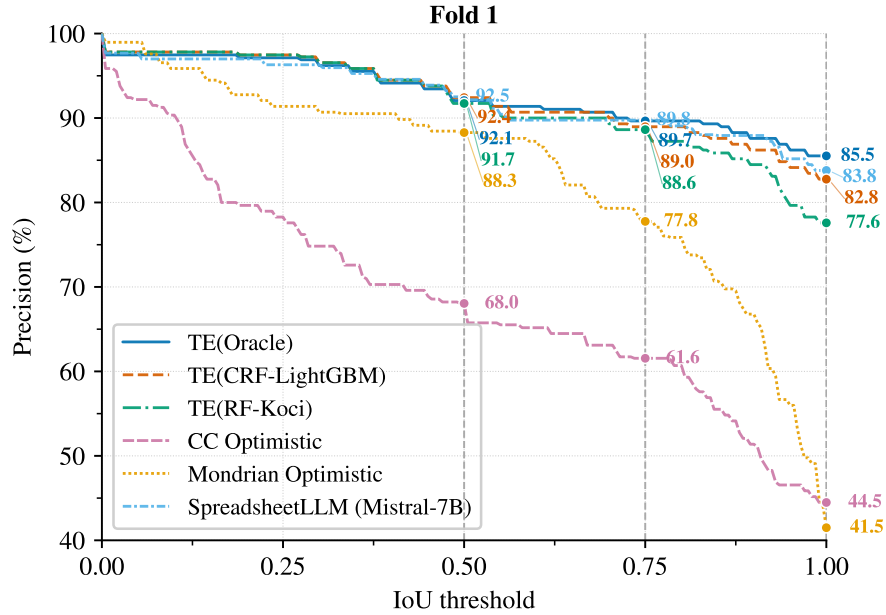


Figure 6.12: Macro-averaged precision as a function of IoU threshold τ for fold 1.

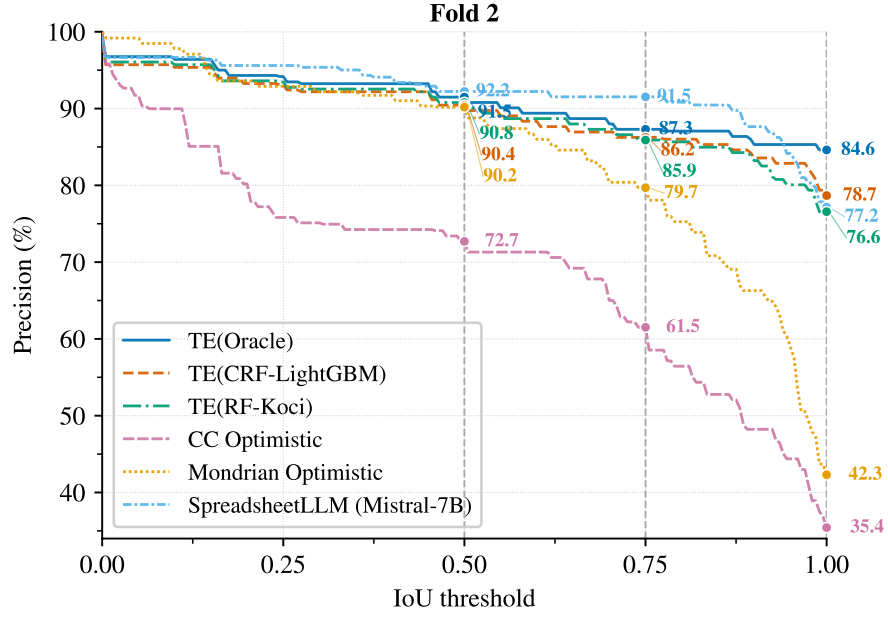


Figure 6.13: Macro-averaged precision as a function of IoU threshold τ for fold 2.

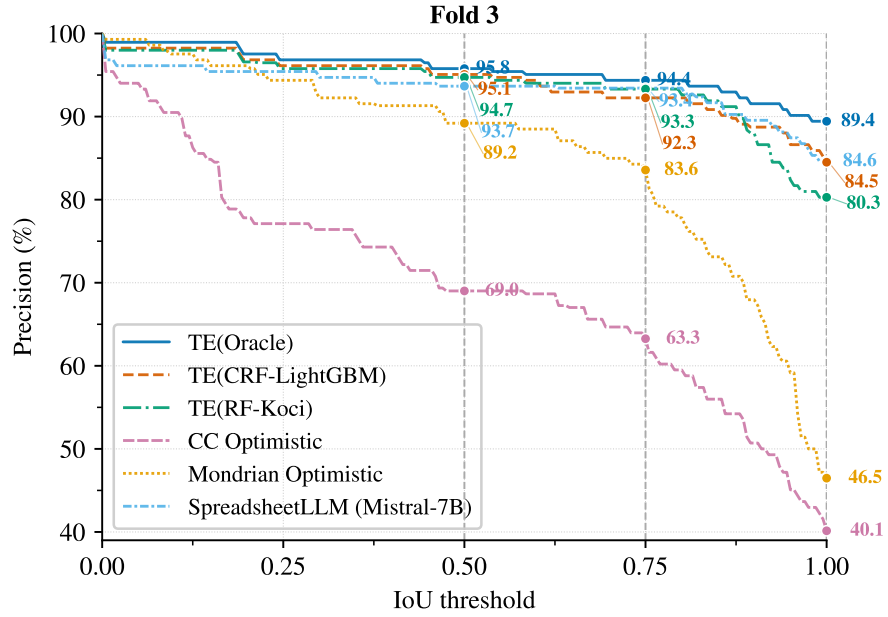


Figure 6.14: Macro-averaged precision as a function of IoU threshold τ for fold 3.

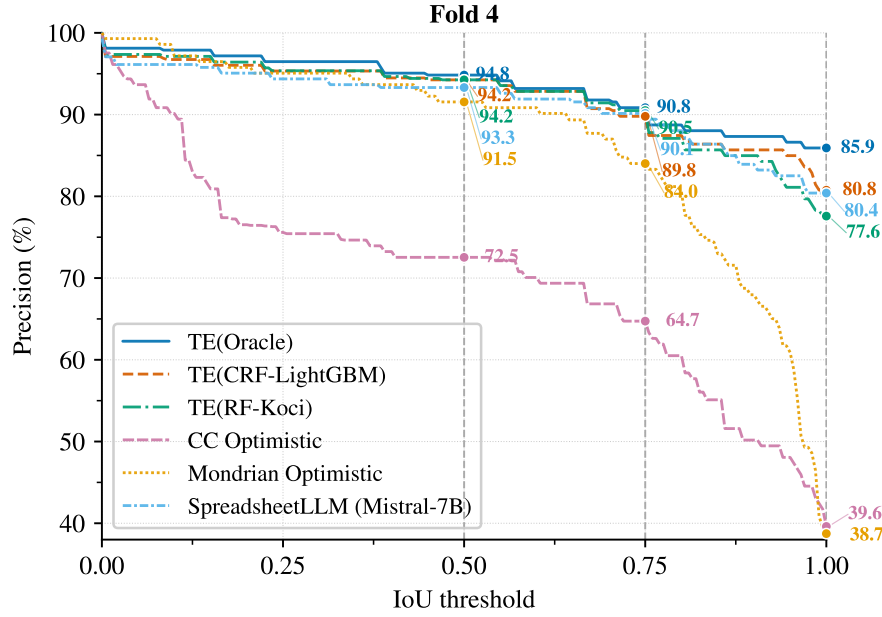


Figure 6.15: Macro-averaged precision as a function of IoU threshold τ for fold 4.

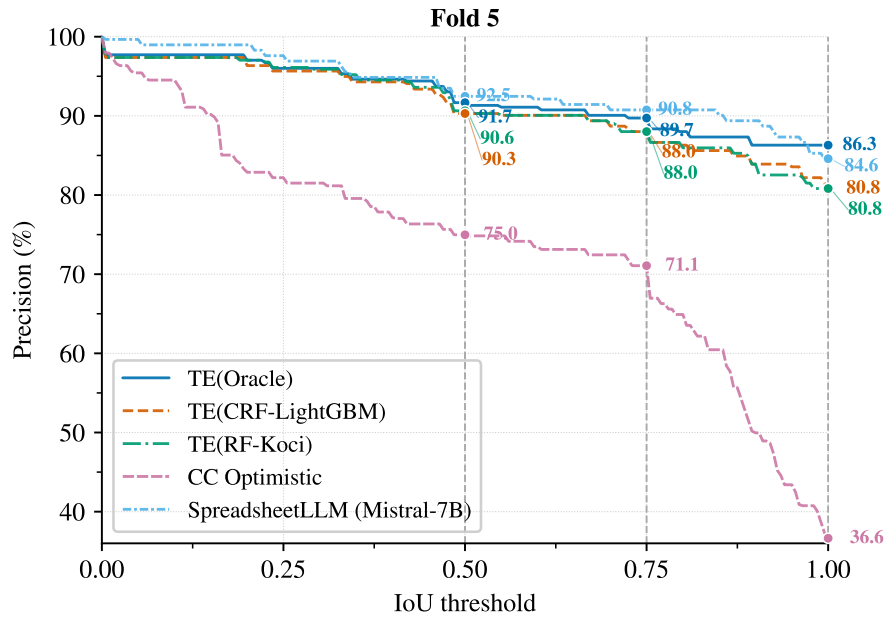


Figure 6.16: Macro-averaged precision as a function of IoU threshold τ for fold 5. MONDRIAN timed out on this fold and is excluded.

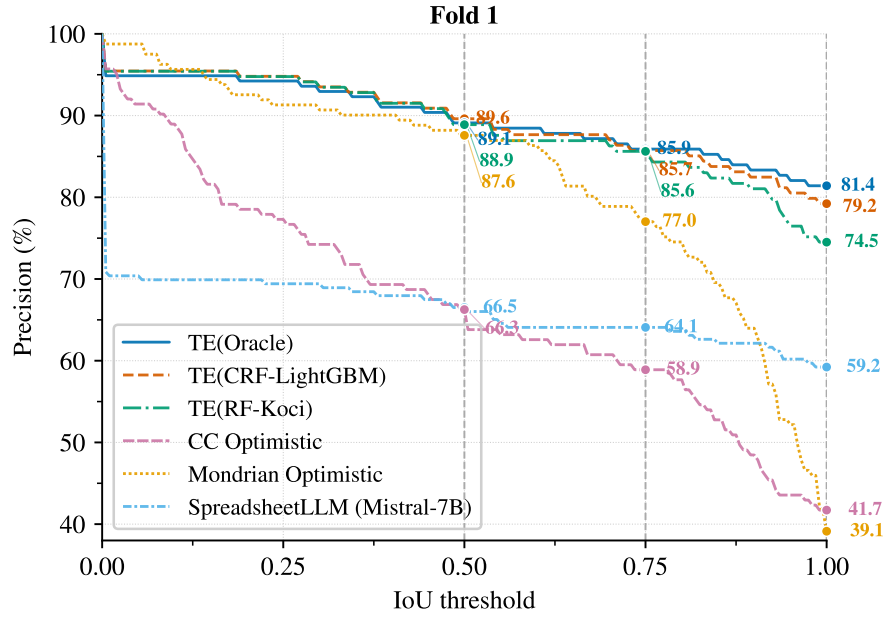


Figure 6.17: Micro-averaged precision as a function of IoU threshold τ for fold 1.

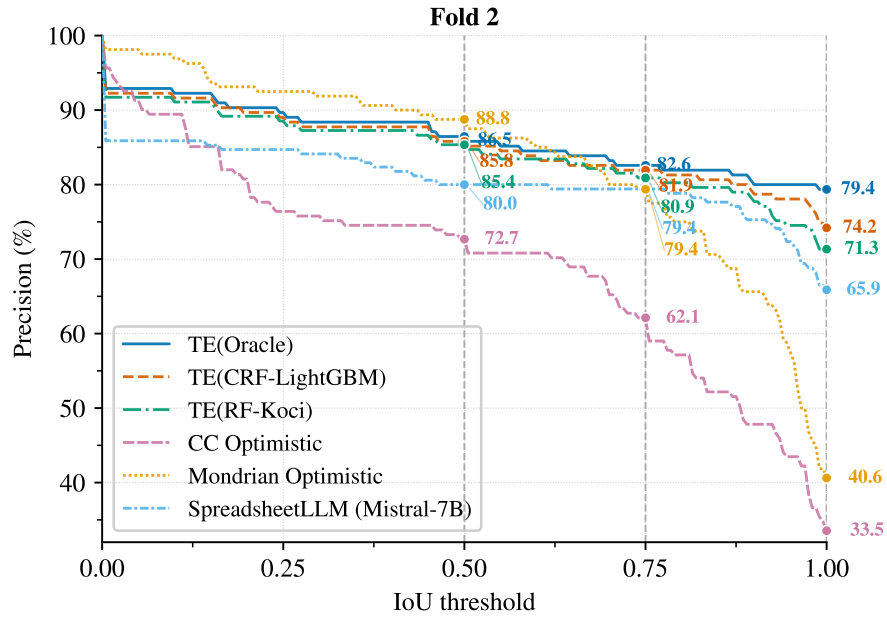


Figure 6.18: Micro-averaged precision as a function of IoU threshold τ for fold 2.

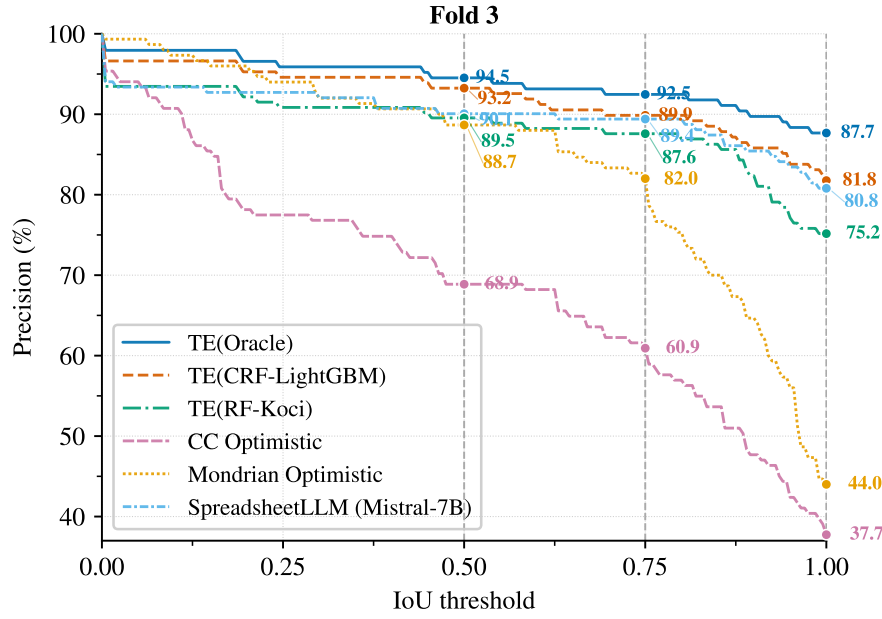


Figure 6.19: Micro-averaged precision as a function of IoU threshold τ for fold 3.

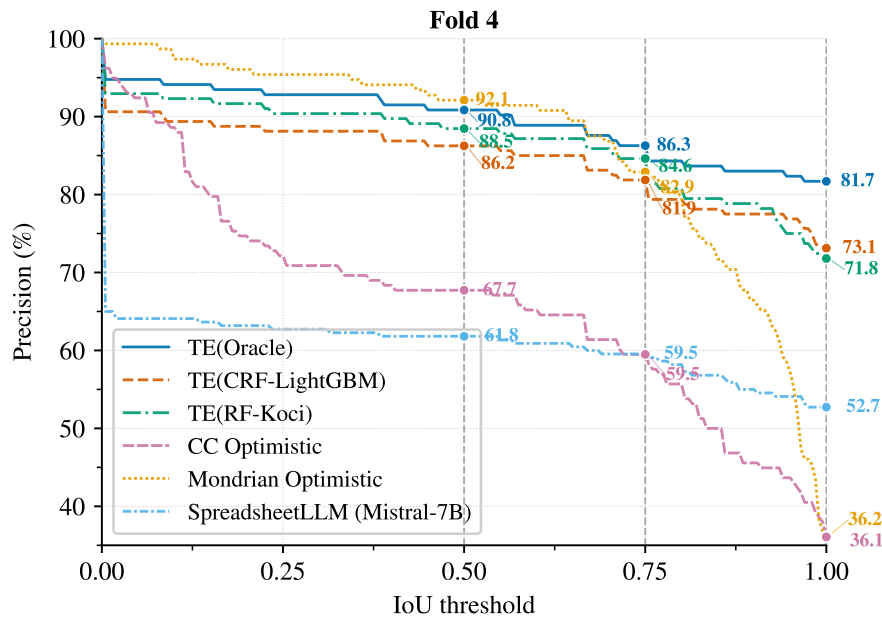


Figure 6.20: Micro-averaged precision as a function of IoU threshold τ for fold 4.

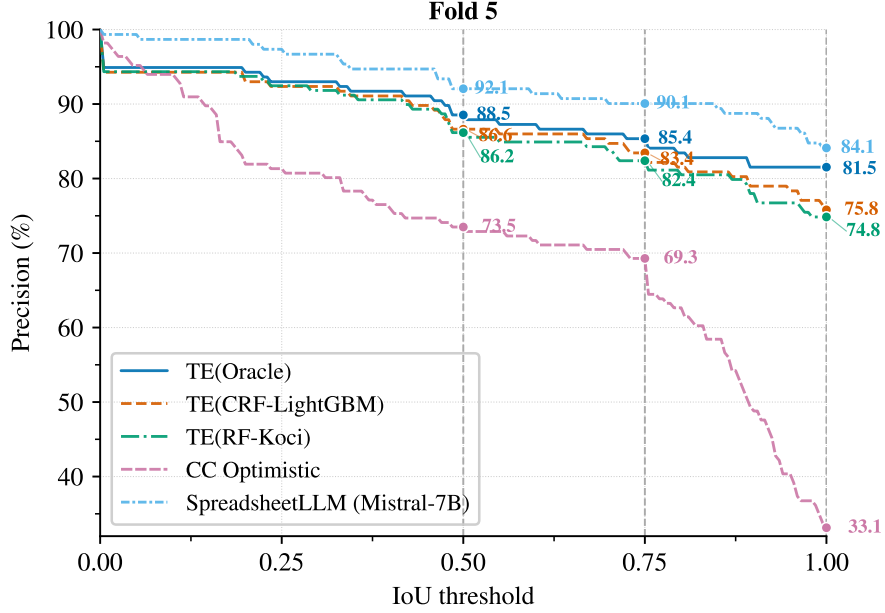


Figure 6.21: Micro-averaged precision as a function of IoU threshold τ for fold 5. MONDRIAN timed out on this fold and is excluded.

regions often overlap tables but fail to recover precise boundaries. These results suggest that *generic region extraction is insufficient for precise spreadsheet table extraction*.

Comparison with SPREADSHEETLLM. SPREADSHEETLLM is close to TE(CRF-LIGHTGBM) in macro precision (+0.6 pp at IoU = 1, 81.5 $\rightarrow$ 82.1%), but substantially worse on micro precision (−8.3 pp, 76.8 $\rightarrow$ 68.5%). This larger macro/micro gap suggests *stronger over-prediction on complex spreadsheets*. We also observe *higher variance across folds*, indicating greater sensitivity to the training distribution. In addition, model outputs are occasionally inconsistent with the requested output format and require post-processing. Overall, TE(CRF-LIGHTGBM) achieves competitive results while remaining substantially cheaper than SPREADSHEETLLM. The oracle results further indicate that additional gains remain possible through improved CTC predictions.

6.5.3 Experiment Conclusions

Our experiments show that our proposed CTC method, combining spatial, sequential, and non-linear features, and the deterministic, learning-free TE method, outperform (or perform similarly to) state-of-the-art DL competitors (TUTA for CTC, SPREADSHEETLLM for TE), for only a fraction of their computational cost and without requiring a GPU. Combined with the determinism of our TE method (in contrast to deep-learning approaches, which behave as black boxes), this advances the state of the art from both a practical and scientific perspective.

6.6 Summary

In this chapter, we addressed the problem of scalable spreadsheet understanding, motivated by the need for robust, cost-efficient extraction pipelines for large corpora of heterogeneous public spreadsheets. We introduced STATSHEETS, a publicly available dataset of 737 manually annotated sheets from 14 organizations across multiple countries, languages, and formats, which, unlike prior benchmarks, jointly supports both Cell-Type Classification (CTC) and Table Extraction (TE) under realistic conditions. We proposed an efficient two-stage pipeline combining a CRF-LIGHTGBM classifier that exploits a rich feature space of lexical, formatting, positional, and structural cues with a deterministic five-stage TE algorithm operating on the predicted cell-type grid. Experiments on STATSHEETS show that our pipeline achieves a Mean File-Macro F_1 of 0.937 on CTC, within 0.6 percentage points of the much heavier TUTA transformer, and consistently outperforms generic region-based methods on TE while remaining competitive with LLM-based systems at orders-of-magnitude lower computational cost and without requiring a GPU.

Directions for future work include extending STATSHEETS to further domains and spreadsheet practices, e.g., financial or scientific archives with non-standard layouts, to improve the generalization of learned features. A natural next step is also to integrate the pipeline more tightly with downstream tasks: the cell-type grid and table boundaries produced by our CTC+TE pipeline could directly seed statistical table discovery (Chapter 5) or other question-answering systems.

TabAgree: Spreadsheet Table Annotation

7.1 Introduction

Progress on the table understanding tasks studied in this thesis, from table discovery (Chapter 5) to cell-type classification and table extraction (Chapter 6), depends on a resource that is easy to overlook: high-quality annotated data, both to train models and to evaluate them fairly. Structure tasks need labels for table boundaries and cell roles; relevance tasks need human judgments of which tables answer a query.

Yet, annotating real-world tables, and spreadsheets in particular, is rarely straightforward. Structural labeling must account for heterogeneous layouts, sparse regions, merged cells, and complex multi-row or nested headers. Similarly, assessing the relevance of a table with respect to a question can require subtle, sometimes expert, judgment. As a result, annotators frequently disagree. Such disagreement provides useful information for evaluating annotation quality and characterizing task difficulty, both of which are important for the interpretation of system performance.

We present TABAGREE, a **spreadsheet-centered, agreement-aware annotation platform** designed to address the limitations of existing tools for table-structure and relevance annotation at scale. It supports both fine-grained cell- and table-structure annotation, carried out over fully formatted spreadsheets, and question–table relevance assessment. It is designed to report inter-annotator agreement through interpretable, task-specific metrics. TABAGREE has already been used in a real, multi-annotator deployment to build the statistical table discovery benchmark composed of *Eurostat* tables in Chapter 5, which directly motivated its agreement-aware design.

Contributions

In this chapter, our contributions are as follows.

- (1) We design and implement TABAGREE, a **client–server annotation platform** supporting simultaneous multi-annotator work, automated task assignment, and centralized annotation storage over fully formatted spreadsheets in all standard formats (XLS, XLSX, ODS, CSV, TSV).
- (2) We provide a **table relevance annotation workflow** for the TD and TQA tasks, offering browser-based multi-class relevance judgments over question–table pairs with full edit and review capabilities (Section 7.3.1).

- (3) We provide a **cell and structure annotation workflow** for the TSR, CTC and TE tasks, leveraging *LibreOffice Calc* (LO) to preserve full spreadsheet formatting during annotation, with automated post-processing (label normalization, sanity checks, and *NumPy* export) enabling annotation at scale including for files with millions of cells (Section 7.3.2).
- (4) We design **task-specific, interpretable inter-annotator agreement metrics**, as alternatives to Cohen’s and Fleiss’ Kappa (which are sensitive to class imbalance), with live monitoring and per-item agreement visualization (Section 7.3.3).

The TABAGREE source code, deployment instructions, sample data, and a link to the demonstration video are available at [84].

This chapter is organized as follows. Section 7.2 motivates the need for a dedicated annotation platform and surveys the limitations of existing tools. Section 7.3 describes the architecture and annotation workflows of TABAGREE. Section 7.4 presents the demonstration scenario.

7.2 Positioning with Respect to Existing Datasets and Tools

Despite rapid advances in large language models, manual annotation of tables (at the table or cell level) is still necessary. For instance, TQA requires (question, table) gold standard pairs, where the table is relevant for the question. One can prompt an LLM with a table and ask it to generate questions about it, producing *synthetic* (question, table) pairs. Such synthetic data, however, suffers from two main issues. The first is *quality*: LLMs frequently produce questions that are overly precise, uninteresting, or unrealistic, and direct prompting often yields low-quality results [11]. Workarounds such as generating SQL queries first and then converting them to NL still yield unrealistic questions, e.g., “Which address corresponds to the condition that September is 7626?” or “What is the average zip code for the state of IL?”. In high-dimensional, statistical datasets such as the *Eurostat* one in Chapter 5, the large number of columns and combinations of attributes lead to questions unlikely to be asked by a human. The second issue is *sensitivity to prompt formulations*: in [90], adding table context leads to questions containing the full table title, such as “What is the trend of passenger cars by unloaded weight from 1996 to 2023 in the table ‘Passenger cars by unloaded weight’?”. Such questions not only are unlikely to be posed by humans but also introduce biases for neural retrieval models, which will expect such titles in user questions. These examples highlight the continued need for human annotations, particularly when aiming for systems that can handle nuanced or domain-specific queries with confidence.

Existing annotation datasets also have significant limitations. Popular TD and TQA datasets such as FETAQA [139] and NQ-TABLES [95] contain only relational tables, where one row typically describes one real-world object; some are also very small, e.g., tables of [139] have between 3 and 34 rows (mean 13.8) and 3 to 7 columns. Real-world tables can not only be much larger, but they may also be non-relational, e.g., spreadsheets

with complex layout meant for human users, and/or multi-dimensional datasets where each column and each row describes one or several dimensions (recall Chapter 5). This underscores the need for *new, high-quality annotated datasets to train and evaluate systems effectively*. We developed an open-access, annotation-focused platform for this task.

Existing annotation tools are currently few, and limited. Most solutions for question–table relevance have limited UIs and are not shared publicly, e.g., that of FETaQA [139]. TSR, CTC and TE are even more challenging: some existing tools focus on automatic semantic annotation of columns or headers (e.g., mapping entities or types [141]) rather than cell-level labels or table boundaries. The closest work addressing TSR and CTC, to our knowledge, is the [annotation tool](#) associated with [113]. It provides a Java application embedding an *Excel* spreadsheet interface (via OLE). While similar to our approach in that it allows labeling cell ranges directly within a spreadsheet grid, it has several key limitations. First, it requires *Microsoft Excel*, which is proprietary, whereas our tool uses *LibreOffice Calc*, a free and open-source alternative. More importantly, it only supports *manual annotation of one spreadsheet at a time*; scaling to hundreds of files is time-consuming and cumbersome. Our approach overcomes this by *automatically managing file selection, opening, and closing*, enabling annotation at scale. Additionally, it offers no support for inter-annotator agreement, nor does it provide visualizations or statistics. Integration into our tool is impractical because it runs on Windows and stores annotations in a new sheet within each spreadsheet. This approach is *unsuitable for multi-annotator settings*: managing multiple annotators would require storing separate copies of each spreadsheet, which is unmanageable for already large datasets. Platforms such as *Label Studio* only accept CSV/TSV formats, provide minimal UI support, and lack support for error correction or annotator agreement; others (e.g., *Prodigy*) do not support tabular data. Advanced frameworks like *Streamlit*, *Flask*, or *Gradio* do not fully support spreadsheets, and tools like *Pyspread* support only limited formats (CSV and a simplified version of XLSX). At the same time, *annotation requires presenting the full spreadsheet, including all formatting features to distinguish headers, data cells, and contextual elements such as titles or sources*. Thus, these tools do not meet the needs of TSR annotation.

We overcome these limitations in an *open-source tool*, which supports *multiple users, small to very large files, rich formatting, and all standard formats (XLS, XLSX, ODS, CSV, TSV)*. Leveraging *LibreOffice Calc* as a macro-enabled spreadsheet editor, combined with a *Flask*-based Web application, our platform is *user-friendly*: it allows efficient annotation of cell roles and table boundaries, and computes inter-annotator agreement, addressing the gaps of existing datasets and tools. We rely on *LibreOffice Calc* because only *Microsoft Excel* and *LibreOffice Calc* render all formatting and support macros over every standard spreadsheet format; the lighter Python libraries and Web frameworks discussed above do not. Of the two, only *LibreOffice Calc* is free, open-source, and cross-platform, whereas *Microsoft Excel* is proprietary. It thus maximizes heterogeneity coverage while favoring open source and interoperability.

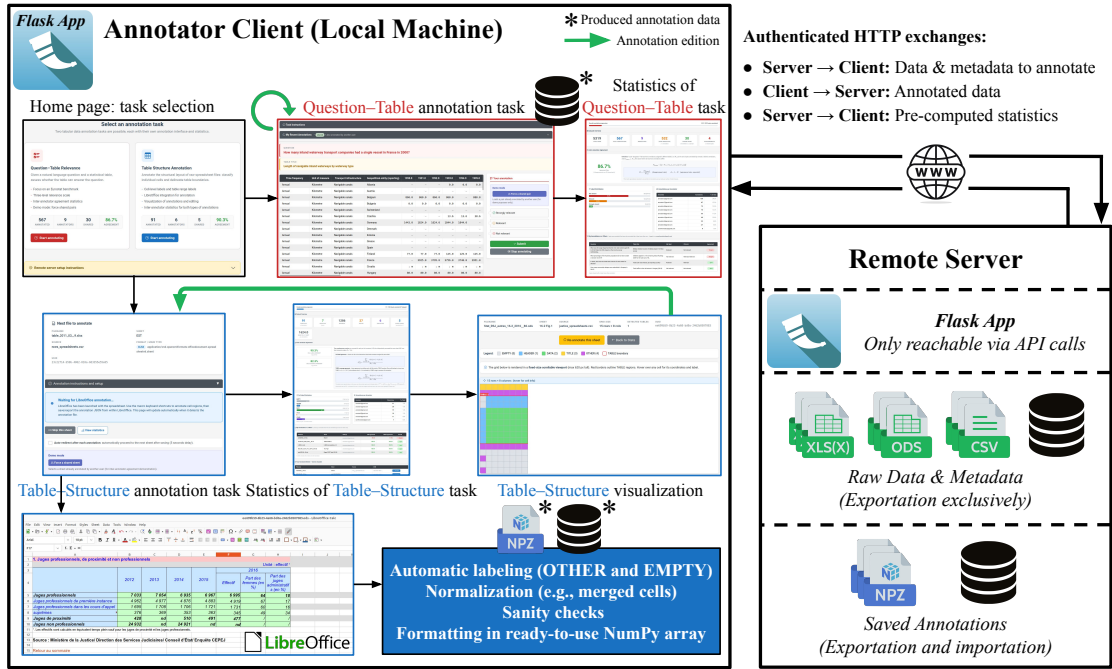


Figure 7.1: TABAGREE overall architecture

7.3 The TabAgree Annotation Platform

Figure 7.1 presents the architecture of TABAGREE, an annotation platform supporting inter-annotator agreement at scale. Multiple annotators can simultaneously work on shared data through a Web-based interface while annotations are centrally managed. TABAGREE follows a client-server architecture. Each annotator runs a local *Flask* Web GUI communicating via HTTP with a remote server exposing an API. Annotators authenticate with an email and a (shared) password. The remote server is responsible for:

- serving data and metadata for annotation or statistics computation (read-only operations); and
- storing submitted annotations (write operations).

Annotation information is stored in *SQLite* databases and as files with unique names combining data identifiers and annotator identity. Data are shared across annotators but separated per task, allowing the platform to support many different annotation efforts where various user sets collaborate. The server is also responsible for proposing to each user the next item to annotate (randomly chosen among those not annotated yet *by this user*).

After authentication, annotators access a homepage (Figure 7.1) with two panels corresponding to the supported tasks: **Question-Table annotation** and **Table-Structure**

annotation. Each panel displays key statistics. Navigation between pages is shown by black arrows (navigation bars omitted for clarity). The client interface also enables setting up the remote server.

7.3.1 Table Relevance Annotation

In this task, users assess the relevance of a table (content and title) to a natural language question. In our demonstration, three labels are available: *strongly relevant*, *relevant*, and *not relevant* (the label set is configurable). The annotation page is displayed once all required data and metadata are downloaded from the server. Most tables load instantly, but very large ones (hundreds of MBs in our scenario) may take several seconds. Reasonably sized tables are embedded in the webpage; for very large tables, embedding is inefficient and a download button enables inspection in the user's spreadsheet editor of choice. Detailed task instructions and data-specific notes are also provided. The interface presents (top-center in Figure 7.1):

- (i) the question (red),
- (ii) the table title (yellow), and
- (iii) the table (embedded or downloadable).

The annotators can interact with the page's buttons to:

- (i) select one of the three relevance labels,
- (ii) submit the annotation (which sends the annotation to the server and asks for the retrieval of a new item to annotate),
- (iii) stop annotating,
- (iv) force a shared annotation (for demonstration purposes only).

Two collapsible panels complement the interface:

- (i) annotation guidelines (editable in the HTML templates);
- (ii) a summary of previously annotated items by the current user, including title, question, chosen label, shared/non-shared status, and an edit button to choose another label.

Editing reloads the corresponding data and updates the stored annotation, allowing corrections or reconsideration, e.g., after consulting agreement statistics. A dedicated page provides statistics and inter-annotator agreement information (Section 7.3.3).

7.3.2 Table Structure Annotation

In this task, users assign structural labels to spreadsheet cells and identify table ranges within sheets. In the scenario where statistical datasets are annotated, we use five cell labels: HEADER, DATA, TITLE, OTHER, and EMPTY. For table ranges, we use the label TABLE. Again, the label set is configurable. Upon page loading (after complete retrieval), metadata are displayed (filename, source, sheet name, format, etc.), along with detailed setup instructions (middle-right in Figure 7.1). As motivated in Section 7.2, annotation is performed in LO to preserve full spreadsheet formatting. A downloadable macro file provides keyboard shortcuts to assign labels and define table ranges.

TABAGREE offers two annotation modes. In the *external mode*, *LibreOffice Calc* is automatically launched on the annotator’s machine and loads the correct sheet, driven by keyboard shortcuts, which we find most efficient on large sheets. To trigger these local commands, the *Flask* client starts a lightweight local HTTP server that listens for them. In the *embedded mode*, no external executable is needed: *LibreOffice Calc* runs directly in the browser through *WebAssembly* (WASM), with all its formatting features, and labels are assigned through buttons integrated in the HTML page that annotate in real time and update the embedded rendering. Both modes share the same server, storage, and post-processing; the external mode, in particular, motivates the separation between the remote API server and the local GUI client.

Annotations are produced via macros inside LO (bottom-left in Figure 7.1). When the annotator saves and closes the sheet, an annotation file is generated and detected by the *Flask* app, which post-processes it (bottom-right in Figure 7.1), as follows.

- (i) *Automatic labeling.* EMPTY is automatically assigned to cells without content; cells left unlabeled by users are assigned OTHER.
- (ii) *Normalization.* Merged cells are de-merged and labels are propagated to reflect the true cell grid. Consecutive $k \geq 2$ empty rows or columns are compressed into exactly two, reducing annotation size.
- (iii) *Sanity checks.* Only the last label assigned to a cell is kept, supporting on-the-fly corrections within LO. Structural constraints are verified (e.g., each HEADER or DATA cell must be associated with a unique TABLE range). If inconsistencies are detected, LO automatically reopens for correction, and a message is displayed on the task page to inform the annotator.
- (iv) *Formatting.* Final annotations are exported as NumPy arrays, convenient for machine-learning use.

Although post-processing may take a few seconds for very large sheets, it significantly accelerates annotation by automating repetitive labeling, enforcing consistency, and normalizing outputs for downstream research efforts. Additional interface controls include:

- (i) a skip button (useful for non-statistical sheets such as tables of contents or graphics),

- (ii) access to statistics and inter-annotator agreement information (Section 7.3.3),
- (iii) an option to automatically proceed to the next sheet after completion,
- (iv) a *Force shared sheet* button (as in Section 7.3.1, for demonstration purposes only).

7.3.3 Annotation Statistics and Inter-Annotator Agreement

Each task provides a dedicated statistics and inter-annotator agreement page. While task-specific metrics differ, both pages share a common structure. A progress bar summarizes the completion relative to the remotely stored data. Below it, key figures describe the current annotation state: number of annotated and remaining items, annotators involved, shared annotations, and task-specific indicators (e.g., mean number of cells per sheet for Table–Structure annotation). Annotation is blind by design: while annotating, users see only the item being labeled (a question and a table, or a spreadsheet), never others’ labels or the current agreement. These statistics live on this separate page, reachable only after submission, so that a choice cannot be anchored to an existing consensus. The page could even be restricted to a meta-annotator to remove any bias; we nonetheless make it available here, as observing agreement is central to understanding task difficulty. A dedicated panel reports the inter-annotator agreement score (computed remotely) with a concise description of the scoring function. Instead of relying on classical reliability measures such as Cohen’s or Fleiss’ Kappa, whose interpretation can be sensitive to class imbalance, we define task-specific agreement functions designed to be directly interpretable as proportions of agreement.

Question–Table Pair Agreement

This task is multi-annotator and multi-class, with a single label per pair. Let $P_{\geq 2}$ denote the set of pairs annotated by at least two distinct annotators, and $P_{\text{disagree}} \subseteq P_{\geq 2}$ the subset for which at least two labels differ:

$$P_{\text{disagree}} := \{p \in P_{\geq 2} \mid \exists a_i, a_j \text{ s.t. } \ell_{a_i}(p) \neq \ell_{a_j}(p)\}.$$

The agreement rate reported on the page is then defined as:

$$A := 1 - \frac{|P_{\text{disagree}}|}{|P_{\geq 2}|},$$

measuring the proportion of multi-annotated pairs without disagreement.

Table–Structure Agreement

Table–Structure annotations consist of (i) cell-wise labels forming a matrix and (ii) zero, one, or multiple TABLE ranges. Two complementary agreement scores are therefore

defined. Let a and b be two annotators of the same sheet, and $R_a \cap R_b$ their intersection region (identical in our current labeling scheme, as ensured by sanity checks, but generalized here for extensibility). The pairwise cell-label agreement is:

$$A_{a,b}^{\text{cell}} := \frac{\sum_{(r,c) \in R_a \cap R_b} \mathbb{1}[\ell_a(r,c) = \ell_b(r,c)]}{|R_a \cap R_b|}$$

where $\mathbb{1}[\cdot]$ is the indicator function. For each sheet u , pairwise agreements are averaged across annotator pairs, then across all sheets annotated by ≥ 2 annotators:

$$\overline{A^{\text{cell}}} := \frac{1}{|S|} \sum_{u \in S} \frac{1}{\binom{|M_u|}{2}} \sum_{\{a,b\}} A_{a,b}^{\text{cell}}$$

where S is the set of sheets annotated by ≥ 2 annotators, M_u the annotators of u , and the sum ranges over pairs of distinct annotators. For table-range annotations, we compute a coverage agreement score. Let $t_a(r,c) \in \{0,1\}$ indicate whether cell (r,c) belongs to a TABLE range in annotator a 's annotation (cells belong to at most one). The pairwise table-coverage agreement is:

$$A_{a,b}^{\text{tbl}} := \frac{\sum_{(r,c) \in R_a \cap R_b} \mathbb{1}[t_a(r,c) = t_b(r,c)]}{|R_a \cap R_b|}.$$

The overall table-coverage agreement $\overline{A^{\text{tbl}}}$ is computed analogously to $\overline{A^{\text{cell}}}$. Aggregated statistics summarize the number of annotated items per annotator and the label distribution for each task. A table lists items annotated by the current annotator that were also annotated by others, indicating whether label match and, if not, the differing labels for Question–Table pairs or the corresponding cell-wise and table-coverage scores for Table–Structure sheets. For the Table–Structure task, an additional table lists completed sheets with metadata and a *Visualize* button displays the saved label matrix (*NumPy* NPZ format) to inspect automatic labeling and normalization effects.

7.4 Interactive Demonstration and System Walkthrough

We demonstrate TABAGREE on a scenario covering both annotation tasks using data from real-world research sources, deployed via a local client GUI connected to a preconfigured remote server. The Question–Table annotation task relies on *Eurostat* tables, specifically the (question, table) pairs compiled in Chapter 5 for the STD problem, together with the annotation instructions and data-specific notes used in the original study. The Table–Structure annotation task uses the statistical spreadsheets collected for the construction of STATSHEETS (Chapter 6), from national agencies (e.g., *US Census Bureau*, *General Authority for Statistics* of Saudi Arabia, *Japan's Ministry of Internal Affairs*) and international organizations (e.g., *World Health Organization*, *OECD*). Files vary in language, length, layout, and format (more details in Section 6.4).

The server is pre-loaded with annotations from multiple annotators, so that even a *single* annotator is immediately in a multi-annotator setting. A typical walkthrough starts from a deliberately ambiguous item, such as a multi-header spreadsheet with merged cells, or a borderline (question, table) pair. After submitting, the statistics page shows *exactly where* and why the decision diverges from those of other annotators; the annotation can then be edited, and the agreement score updates live. For the Table–Structure task, the *Visualize* button further shows how automatic labeling and normalization reshaped the raw input. The takeaway goes beyond a binary agree/disagree view: it makes visible *where ambiguity lies* and how low agreement signals a harder task. A pre-annotated sample profile also allows exploring statistics, inter-annotator agreement, and Table–Structure visualizations without extensive prior annotation effort. A full walkthrough of the platform is available as a demonstration video (<https://youtu.be/fzCyXLUcowQ>).

7.5 Summary

In this chapter, we addressed a practical bottleneck in table understanding research: the lack of a unified, scalable annotation platform capable of handling the full complexity of real-world spreadsheets. We motivated the need for dedicated tooling by highlighting the limitations of existing solutions, both for table relevance labeling (TD and TQA) and for fine-grained cell-level annotation (CTC, TSR, and TE), and by arguing that synthetic data generation via LLMs remains insufficient for high-quality benchmarks. We presented TABAGREE, a client–server annotation platform built around *LibreOffice Calc* as a macro-enabled spreadsheet editor, preserving full formatting and enabling annotation at scale across all standard formats. TABAGREE provides two dedicated workflows: browser-based multi-label relevance annotation for question–table pairs, and cell- and range-level annotation within LO, with automated post-processing (label normalization, sanity checks, and *NumPy* export). It replaces classical inter-annotator agreement metrics (Cohen’s and Fleiss’ Kappa, sensitive to class imbalance) with task-specific, directly interpretable agreement functions, and surfaces them live to annotators.

A natural improvement for future work is to make the agreement signal active: rather than sampling items uniformly, TABAGREE could direct annotation effort toward the items where annotators most disagree, so as to improve the quality of the produced ground truth for a given annotation budget.

Conclusions and Perspectives

This chapter concludes the thesis by summarizing its main contributions and outlining directions for future research. Section 8.1 reviews the four core contributions developed in this thesis: focused Web crawling, spreadsheet table understanding, statistical table discovery, and the TABAGREE annotation platform. It discusses how these contributions address the acquisition, structuring, and retrieval of statistical data on the Web. Section 8.2 presents perspectives for future work, both within each research direction and across them. In particular, it discusses the integration of the proposed methods and systems into a unified end-to-end pipeline for statistical data acquisition and exploitation.

8.1 Summary of Contributions

Statistical datasets are a public good. Governments, international bodies, and national agencies release vast amounts of figures on population, employment, health, or the economy, and these figures inform policy debates, support journalistic fact-checking, and shape how citizens understand the world around them. Yet the very abundance of this data is also what makes it hard to use: statistics are scattered across hundreds if not thousands of websites, published in heterogeneous formats, and organized in ways that assume a human reader rather than a machine. This thesis started from that tension. Making statistical data usable in practice requires a chain of capabilities, from finding the data on the Web, to recovering its structure, to retrieving the right table for a given information need, and the work presented here addresses several links of that chain.

The starting point is acquisition: before anything can be structured or retrieved, the data has to be found. A full, exhaustive crawl of a website containing statistical datasets is far too costly, so in Chapter 4 we framed the search for statistical datasets within a site as a graph exploration problem, proved that solving it optimally is out of reach, and proposed SB-CLASSIFIER, a reinforcement-learning crawler that learns the shape of a website as it explores it. Its guiding idea is simple: links found on similar tag paths in a page tend to lead to similar content, so the crawler can follow the promising ones and retrieve many target files while making as few requests as possible, with no prior knowledge of the site and none of the parallel machinery of a search engine.

Finding the files is only useful if their content can be read. Statistical data are usually found as spreadsheets meant for human readability, where the structure of the tables is visual rather than explicit, so in Chapter 6 we worked on recovering it. We split

the problem into classifying the type of each cell and grouping cells into tables, and built a pipeline that pairs a non-linear classifier with a CRF that keeps neighboring decisions coherent, followed by a deterministic step that extracts the table boundaries. We experimentally showed that each component of the model contributes to its performance: spatial awareness, non-linearity, and sequential modeling each improve the results. We further showed that our approach performs comparably to, and sometimes better than, Transformer-based and LLM-based models, at a substantially lower cost and without requiring a GPU. So that future work can build on these tasks, we also constructed and released *STATSHEETS*, a benchmark of real statistical spreadsheets annotated for both of them. It is deliberately heterogeneous, covering a range of sources, languages, topics, sheet sizes, and file formats, so that methods can be evaluated on the diversity that statistical spreadsheets present in practice.

With the data found and its structure recovered, the question becomes which table answers a given need, expressed in natural language. Standard table discovery methods assume entity-oriented relational tables, which statistical data is not. Their cells are mostly numerical and carry no semantic content on their own, the semantic signal being concentrated in the dimension values, such as the time periods and locations the figures describe. They also raise a scalability concern: as multidimensional aggregates, their serialized representation is very large, up to millions of rows, on which traditional table discovery methods do not scale. In Chapter 5, we define Statistical Table Discovery, a variant of table discovery that accounts for these specificities. Our answer to this problem is *STAR*, a system that indexes each table along its temporal and spatial scope separately, and extracts those same elements from the query to match the two. We showed that this retrieves relevant statistical tables more accurately than traditional table discovery methods, while scaling to large corpora and running without a GPU, which end-to-end deep-learning and LLM-based methods require. The system is already used in practice: journalists at *Radio France* rely on it to find relevant figures in large collections of trusted sources. To support further research, we also released a dataset built from the statistical tables published by *Eurostat*, paired with natural-language queries and relevance judgments.

Both table discovery and spreadsheet understanding rely on labeled data, to train models and to evaluate them, and labeling real tables is difficult: spreadsheets have complex layouts, and deciding whether a table answers a question can require subtle, sometimes expert, judgment, so annotators often disagree. In Chapter 7, we presented *TABAGREE*, an annotation platform supporting simultaneous multi-annotator work on spreadsheets and on question–table pairs. The platform has two main characteristics. First, it annotates structure inside a real spreadsheet editor, *LibreOffice Calc*, which preserves the formatting that distinguishes headers, data, and titles and makes it possible to label files with millions of cells. Second, it reports inter-annotator disagreement as an interpretable signal, relying on simple agreement measures that read directly as proportions of agreement. By making annotation at scale and agreement tracking practical, *TABAGREE* is also meant to support further research, as it lowers the effort needed to build new benchmarks for these tasks.

Throughout this thesis, beyond the goal specific to each contribution, we pursued four transversal concerns. The first is efficiency under real constraints: every method was made to run on modest hardware and was judged not only on its accuracy but also on its cost, whether in HTTP requests, computation time, or annotation effort. This does not mean negligible resources, as the volumes involved remain those of large-scale data collections, but none of them requires heavy or specialized infrastructure. The second is heterogeneity: rather than assuming clean and uniform relational inputs, each component was built and tested against the variety of formats, topics, and languages that statistical data takes on the Web. The third is scalability: the methods are meant to operate within pipelines that ingest large amounts of Web data, up to tens of thousands of tables, and each was evaluated with that scale in mind. The fourth is reproducibility: the code developed throughout this thesis is publicly available as open source, and the datasets introduced in Chapters 5 and 6 are publicly shared to support transparent evaluation, comparison, and future research. Together, they show that open tabular data can be acquired, structured, and retrieved with efficient and scalable methods.

8.2 Perspectives

Each component opens specific directions for future work, which we discuss in turn before turning to a broader, transversal perspective.

Data Acquisition

Our crawler operates on the navigable Web and performs a single, one-shot acquisition of a website. Two extensions stand out. The first is the *hidden Web*: many statistical sources expose their data only through search forms or query interfaces, some even requiring creating an account and being logged in, which our crawler does not currently reach. The second, and to us the more promising, is *incremental re-crawling*. Statistical sources are not static; they publish new releases, sometimes revise their publications, and hence regularly update their pages. Rather than re-acquiring a site from scratch, a crawler could exploit the structure it learned during the first crawl to decide, efficiently, which parts of a site are most likely to have changed or to contain new targets. Incremental and re-crawling strategies are an active line of research [9, 53, 120, 147, 154], and connecting them to the knowledge already captured by SB-CLASSIFIER during its initial exploration would let the system keep an index fresh at a fraction of the cost of a full re-crawl.

Statistical Table Discovery

Two directions would strengthen STAR. The first is *user feedback*: relevance judgments collected during use, of the kind TABAGREE is designed to gather, could be fed back into the retrieval model to refine its ranking over time. The second, more ambitious, is *statistical table question answering*. We briefly mentioned a best-effort form of table question answering, but STAR’s design, with its separate indexing of time and space and its parsing of the linguistic elements of a query, could offer a foundation for efficient and

scalable TQA over statistical data, where the answer is not just the relevant table(s) but the (possibly aggregated) figure the user asked for. Pursuing this would extend the scope of the thesis into a complete pipeline, from acquiring data on the Web to answering a user’s question over it. A promising direction here is the use of small, locally run LLMs, whose capabilities on table-centric tasks are improving quickly: recent work shows that accurate table question answering is achievable with open-weight models that run on a desktop or laptop [103], and that cost can be controlled by routing questions adaptively between cheaper and more expensive computation [126]. For statistical data, where privacy and the ability to operate without sending queries to a third party matter, keeping the model local while preserving STAR’s scalability would be an especially valuable property.

Spreadsheet Understanding

Our cell-type classification and table extraction methods were evaluated as standalone tasks, but their natural role is as a first stage for downstream systems. STAR in particular needs exactly such a component to ingest data arriving in the heterogeneous formats and from the varied sources found on the Web. This suggests a tighter coupling in both directions: using structure recovery as a front-end for retrieval, and using signals from the downstream task to improve the representations or training of the CTC and TE models, so that the structure recovered is the structure that downstream use actually needs.

Annotation and Agreement

TABAGREE treats disagreement as a signal, and a natural next step is to make that signal active. Rather than sampling items uniformly, the platform could route the most contested, low-agreement items to additional annotators while retiring those that have already reached consensus, focusing human effort where it most improves data quality, following the principle of *selective repeated labeling* with multiple noisy labelers [157]. A complementary direction is to look for structure in where disagreement concentrates, for instance identifying topics or kinds of tables on which annotators systematically disagree, so that a meta-annotator can investigate, and possibly bring in domain experts for questions that require expert knowledge. A further step would be to measure how the resulting annotations improve downstream retrieval and question answering.

Connecting the Tasks

The perspective we find most compelling is one the thesis did not cover: linking the components so that each one feeds the others. The methods fit together. One could refine the reward of the crawler’s reinforcement-learning agent using the number of statistical tables that spreadsheet understanding recovers downstream, so that acquisition is driven by how many usable tables a site yields rather than by file type alone. Going further, retrieval could drive acquisition: when a question posed to STAR is only partially

answered by the tables in its index, or when the similarity of the most-relevant tables remains unsatisfactory, the system could search the Web for relevant dataset collections, for example using domain-specific dataset discovery techniques [183], pass the identified websites to our crawler to acquire the data efficiently, run the results through spreadsheet understanding, and add the recovered tables to STAR’s index. Such a loop would let the system expand its coverage and improve its answers over time, with very limited human involvement. It would also realize the end-to-end goal that motivated this thesis: a pipeline that runs from acquiring statistical data on the Web all the way to answering the questions users ask.

Other Scientific Contributions

The core contributions of this thesis, namely a focused Web crawler for scalable data acquisition (Chapter 4), the STAR system for statistical table discovery (Chapter 5), a pipeline and benchmark for spreadsheet table understanding (Chapter 6), and the TABAGREE annotation platform (Chapter 7), constitute the main research agenda of the thesis.

In parallel, I took part in several collaborative research projects that led to publications in international peer-reviewed conferences and journals. These works fall outside of the main scope of this thesis, but are additional scientific outputs produced during the PhD. This chapter gives an overview of three such directions: heterogeneous graph exploration and entity path finding (Section 9.1), multimodal information extraction from scientific documents (Section 9.2), and uncertainty reasoning and cooperative game theory (Section 9.3).

9.1 Heterogeneous Graph Exploration and Entity Path Finding

This section presents work published in [18–20], co-authored with Nelly Barret, Jia-Jean Law, and Ioana Manolescu.

CONNECTIONLENS [5] is a system developed within the *CEDAR team* at *Inria Saclay* to help non-expert users explore heterogeneous data sources. It unifies any collection of structured, semi-structured, or unstructured datasets into a single property graph. Each element of the original data (relational tuple, XML element, JSON map, RDF URI, etc.) becomes an internal node; each atomic value becomes a leaf node; and edges encode the connections present in the original datasets. Dedicated Information Extraction (IE) modules further identify Named Entities (NEs), such as people, organizations, locations, or emails, mentioned in the data, and add them as nodes connected to the data items they come from.

Entity path finding [18, 20] A key challenge for users exploring such a graph is to identify *connections between entities of interest*. For instance, a data journalist may want to know how a given person is connected to a given organization across a corpus of heterogeneous documents and datasets. We formalize this as the problem of *Named*

Entity paths: paths in the ConnectionLens graph whose endpoints are NEs and that traverse a sequence of intermediate data nodes. Each such path reveals a non-obvious connection between entities, and yields a structured, row-oriented view of the matching instances in the data.

We propose an interactive and efficient method to enumerate and compute NE paths [18]. The core contribution is an algorithm that identifies high-frequency path fragments to precompute, and reformulates incoming queries to use these cached subpaths, which keeps response times interactive even on large graphs. Experiments on real-world heterogeneous datasets confirm both the usefulness of NE paths for data exploration and the scalability of the computation.

The full research line is presented in a journal article [20], which extends the original conference contribution in two directions. First, it introduces a principled analysis of entity-to-entity paths of arbitrary structure contained in heterogeneous datasets. Second, it proposes a path ranking method that surfaces the most meaningful connections in dense or noisy graphs, using a dedicated IE module built on top of a large language model to assess path relevance. The experiments use datasets drawn from investigative journalism scenarios, and show the effectiveness of both the enumeration algorithm and the ranking method.

PathWays demonstration [19] PATHWAYS [19] is a working demonstration of the above approach. It lets users interactively select entity types of interest, enumerate the connecting paths across a heterogeneous data graph, and inspect the per-path tabular result sets.

9.2 Multimodal Information Extraction from Scientific Documents

This section presents work published in [68, 134–136]. I contributed to the first one with my supervisor Pierre Senellart alone; the other three are co-authored with Shrey Mishra, Neil Sharma, and Pierre Senellart.

Mathematical and scientific literature contains rich structured content (e.g., theorems, lemmas, propositions, proofs, definitions) that is not directly machine-accessible when articles are distributed in PDF form. The THEOREMKB [134] project in the *Valda team* aims to build a knowledge base of mathematical results by automatically extracting these environments from large corpora of *arXiv* papers.

Document class inference [68] Information extraction from scientific PDFs is much easier when the L^AT_EX *document class* used to produce the article is known, since each class imposes distinct typographic conventions for section headings, theorem environments, reference formatting, and column layout. In [68], we address the problem of automatically identifying the document class from the PDF alone, without access to the L^AT_EX sources. We propose two approaches: a hand-coded classifier based on five geometric features extracted from the PDF layout (margins, column width, and related measurements), and a CNN-based classifier trained on bitmap renderings of the first pages of articles.

We evaluate them on roughly 100 000 *arXiv* papers, for which ground-truth labels are obtained from the L^AT_EX sources. The CNN model reaches over 90% average F₁ across the most frequent document classes, well above the geometric-feature baseline. This work received a *Runner-up for Best Paper Award* at DocEng 2023.

Multimodal theorem and proof extraction [134–136] The extraction of theorem-like environments and their proofs from scholarly PDFs is addressed in [135] as a paragraph-level multimodal classification problem. Scientific PDFs carry three complementary information channels beyond raw text. The *font* information (font family and size) encodes structural conventions: theorem statements are typically set in italics, and proofs end with specific symbols. The *visual* rendering of a paragraph as a bitmap captures layout features that text-only models cannot see. And the *text* content itself carries the remaining information. Three independent unimodal models (a RoBERTa-based language model for text, an EfficientNetV2 CNN for paragraph images, and an LSTM over font sequences) are combined into a multimodal late-fusion architecture through cross-modal attention. A sliding-window Transformer then models the sequential dependencies across consecutive paragraphs, which lets the system handle multi-page proofs and page breaks. Evaluated on roughly 200 000 *arXiv* articles, the multimodal approach consistently outperforms unimodal baselines, and the sequential modeling gives further improvements. A key practical advantage is that the method requires neither OCR preprocessing nor access to the L^AT_EX sources at inference time.

THEOREMVIEW [136] demonstrates this pipeline interactively: given a raw PDF, it identifies all theorem-like environments and their proofs, and presents them in a structured, navigable interface.

9.3 Uncertainty Reasoning and Cooperative Game Theory

This section presents work published in [109], co-authored with Pratik Karmakar and Pierre Senellart.

The Shapley value is a classical solution concept from cooperative game theory. It distributes a collective payoff among players in proportion to their marginal contributions across all possible coalitions. Its axiomatic foundation (efficiency, symmetry, linearity, and null-player invariance) has made it a widely used attribution mechanism in economics, explainable AI, and, increasingly, data management, for example for data valuation and query responsibility attribution in probabilistic databases.

EShap = ShapE [109] A natural extension arises in *probabilistic* settings, where each player participates in a coalition independently with some probability p_i . Two notions of Shapley value arise in this context: the *expected Shapley value* (ESHAP), which averages the Shapley value over all randomized subgames, and the *Shapley value of the expected utility game* (SHAPE), which applies the Shapley axioms directly to the expected utility. Both capture a player’s average contribution under uncertainty. It was *a priori* unclear whether they coincide, since the former averages contributions across subgames while the latter applies the Shapley axioms to the expected utility directly. In [109], we prove that

$\text{ESHAP} = \text{SHAPE}$. This equality confirms the consistency of Shapley-based attribution in probabilistic settings, with direct implications for probabilistic databases and data valuation: the two seemingly distinct definitions yield identical attribution scores. The equality does *not* hold in general for the Banzhaf power index; however, for normalized Banzhaf indices, a separate proof recovers it, which extends the class of attribution tools with provable consistency guarantees in uncertain settings. This work was carried out in collaboration with Pratik Karmakar before and during my research visit to Singapore in 2025.



Bibliography

- [1] COG. <https://www.insee.fr/fr/information/7766585>, 2024. 84
- [2] NUTS. <https://ec.europa.eu/eurostat/documents/345175/629341/NUTS2021-NUTS2024.xlsx/2b35915f-9c14-6841-8197-353408c4522d?t=1702990824080>, 2024. 84
- [3] Elasticsearch. <https://www.elastic.co/>, 2024. 96
- [4] Ayesh Alshukri, Frans Coenen, and Michele Zito. Incremental Web-Site Boundary Detection Using Random Walks. In *Proceedings of the 7th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM 2011*, pages 414–427, 2011. 33
- [5] Angelos-Christos G. Anadiotis, Oana Balalau, Catarina Conceição, Helena Galhardas, Mhd Yamen Haddad, Ioana Manolescu, Tayeb Merabti, and Jingmao You. Graph integration of structured, semistructured and unstructured data for data journalism. *Inf. Syst.*, 104:101846, 2022. 165
- [6] Fatma Arslan, Naeemul Hassan, Chengkai Li, and Mark Tremayne. A Benchmark Dataset of Check-Worthy Factual Claims. In *ICWSM*, pages 821–829, 2020. 26
- [7] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, 47(2-3):235–256, 2002. 38, 39, 76
- [8] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, pages 722–735, 2007. 23
- [9] Konstantin Avrachenkov, Kishor Patil, and Gudan Thoppe. Online algorithms for estimating change rates of Web pages. *Perform. Evaluation*, 153:102261, 2022. 75, 161
- [10] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. Pneuma’s LLM-generated questions on the Chicago dataset. https://storage.googleapis.com/pneuma_open/pneuma_chicago_10K_questions_annotated.jsonl. 99

-
- [11] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. Pneuma: Leveraging LLMs for Tabular Data Representation and Retrieval in an End-to-End System. *Proc. ACM Manag. Data*, 3(3):200:1–200:28, 2025. 25, 77, 78, 79, 82, 90, 97, 99, 100, 108, 150
 - [12] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. Pneuma Code Repository. <https://github.com/TheDataStation/pneuma>, 2025. 97
 - [13] Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat, Antoine Deiana, Emilie Gautreau, Antoine Krempf, Thomas Pontillon, Gérard Roux, and Joanna Yakin. Fact-checking multidimensional statistic claims in French. In *Proceedings of the 2022 Truth and Trust Online Conference, TTO 2022*, pages 20–29, 2022. 26, 91, 92, 93, 95, 109
 - [14] Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat, Antoine Deiana, Emilie Gautreau, Antoine Krempf, Thomas Pontillon, Gérard Roux, and Joanna Yakin. Statistical Claim Checking: StatCheck in Action. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management, CIKM 2022*, pages 4798–4802, 2022. 26, 43, 77, 109
 - [15] Oana Balalau, Pablo Bertaud-Velten, Younes El Fraihi, Garima Gaur, Oana Goga, Samuel S. Guimarães, Ioana Manolescu, and Brahim Saadi. FactCheckBureau: Build Your Own Fact-Check Analysis Pipeline. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024*, pages 5185–5189, 2024. 26
 - [16] Oana Balalau, Simon Ebel, Helena Galhardas, Théo Galizzi, and Ioana Manolescu. STaR: Space and Time-aware Statistic Query Answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024*, pages 5190–5194, 2024. 27, 77, 78, 83, 90, 91, 92, 93, 95, 98, 109
 - [17] Titus Barik, Kevin Lubick, Justin Smith, John Slankas, and Emerson Murphy-Hill. FUSE: A Reproducible, Extendable, Internet-scale Corpus of Spreadsheets. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR)*, pages 486–489, 2015. 25
 - [18] Nelly Barret, Antoine Gauquier, Jia Jean Law, and Ioana Manolescu. Exploring Heterogeneous Data Graphs Through Their Entity Paths. In *Proceedings of the 27th European Conference on Advances in Databases and Information Systems, ADBIS 2023*, pages 163–179, 2023. 165, 166
 - [19] Nelly Barret, Antoine Gauquier, Jia-Jean Law, and Ioana Manolescu. PathWays: Entity-Focused Exploration of Heterogeneous Data Graphs. In *Proceedings of the 20th Extended Semantic Web Conference, ESWC 2023*, 2023. 166

- [20] Nelly Barret, Antoine Gauquier, Jia Jean Law, and Ioana Manolescu. Finding meaningful paths in heterogeneous graphs with PathWays. *Inf. Syst.*, 127:102463, 2025. [165](#), [166](#)
- [21] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. [36](#)
- [22] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA@KDD 2013*, pages 18–26, 2013. [24](#)
- [23] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. UbiCrawler: a scalable fully distributed Web crawler. *Softw. Pract. Exp.*, 34(8):711–726, 2004. [74](#)
- [24] Matko Bosnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. TwitterEcho: a distributed focused crawler to support open research with Twitter data. In *Proceedings of the 21st World Wide Web Conference, WWW 2012*, pages 1233–1240, 2012. [74](#)
- [25] Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics, COMPSTAT 2010*, pages 177–186, 2010. [40](#)
- [26] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. [119](#)
- [27] Leo Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, 2001. [119](#)
- [28] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: exploring the power of tables on the Web. *Proc. VLDB Endow.*, 1(1):538–549, 2008. [22](#), [24](#)
- [29] Rui Cai, Jiang-Ming Yang, Wei Lai, Yida Wang, and Lei Zhang. iRobot: an intelligent crawler for Web forums. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, pages 447–456, 2008. [75](#)
- [30] Tien Duc Cao, Ioana Manolescu, and Xavier Tannier. Searching for Truth in a Database of Statistics. In *Proceedings of the 21st International Workshop on the Web and Databases*, pages 4:1–4:6. ACM, 2018. [26](#), [77](#), [109](#)
- [31] Sandrine Ceurstemont. Controlling AI’s Growing Energy Needs. *Commun. ACM*, 68(3):15–17, 2025. [79](#)

-
- [32] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Comput. Networks*, 31 (11-16):1623–1640, 1999. [26](#), [29](#), [46](#), [47](#), [73](#)
- [33] Donald D. Chamberlin and Raymond F. Boyce. SEQUEL: A Structured English Query Language. In *Proceedings of 1974 ACM-SIGMOD Workshop on Data Description, Access and Control*, pages 249–264, 1974. [22](#)
- [34] Duen Horng Chau, Shashank Pandit, Samuel Wang, and Christos Faloutsos. Parallel crawling for online social networks. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 1283–1284, 2007. [74](#)
- [35] Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. Learning High-Quality and General-Purpose Phrase Representations. In *Findings of the Association for Computational Linguistics, EACL 2024*, pages 983–994, 2024. [98](#)
- [36] Si-An Chen, Lesly Miculicich, Julian Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. TableRAG: Million-Token Table Understanding with Language Models. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems, NeurIPS 2024*, 2024. [109](#)
- [37] Zhe Chen and Michael Cafarella. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*, pages 1126–1135, 2014. [23](#), [111](#), [113](#), [115](#)
- [38] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. Table Search Using a Deep Contextualized Language Model. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, pages 589–598, 2020. [25](#), [82](#), [108](#)
- [39] Zhoujun Cheng, Haoyu Dong, Ran Jia, Pengfei Wu, Shi Han, Fan Cheng, and Dongmei Zhang. FORTAP: Using Formulas for Numerical-Reasoning-Aware Table Pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 1150–1166, 2022. [114](#)
- [40] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. In *Proceedings of the 11th International Conference on Learning Representations, ICLR 2023*, 2023. [25](#), [108](#)
- [41] Junghoo Cho and Hector Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. In *Proceedings of 26th International Conference on Very Large Data Bases, VLDB 2000*, pages 200–209, 2000. [75](#)

- [42] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *Proceedings of the 11th International World Wide Web Conference, WWW 2002*, pages 124–135, 2002. [74](#)
- [43] Martin Pekár Christensen, Aristotelis Leventidis, Matteo Lissandrini, Laura Di Rocco, Renée J. Miller, and Katja Hose. Fantastic Tables and Where to Find Them: Table Search in Semantic Data Lakes. In *Proceedings of the 28th International Conference on Extending Database Technology, EDBT 2025*, pages 397–410, 2025. [25](#)
- [44] Christina Christodoulakis, Eric B. Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. Pytheas: Pattern-based Table Discovery in CSV Files. *Proc. VLDB Endow.*, 13:2075–2089, 2020. [113](#)
- [45] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. CoBase: A Scalable and Extensible Cooperative Information System. *J. Intell. Inf. Syst.*, 6(2/3):223–259, 1996. [24](#)
- [46] City of Chicago. Chicago Data Portal. <https://data.cityofchicago.org/>. [99](#)
- [47] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, 1970. [22](#)
- [48] Remi Coletta, Emmanuel Castanier, Patrick Valduriez, Christian Frisch, DuyHoa Ngo, and Zohra Bellahsene. Public data integration with WebSmatch. In *Proceedings of the First International Workshop on Open Data, WOD 2012*, pages 5–12, 2012. [23](#), [111](#), [114](#), [135](#)
- [49] Common Crawl Foundation. Common Crawl. <https://commoncrawl.org/>, 2026. [25](#)
- [50] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.*, 2006. [69](#)
- [51] Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Fine-grain web site structure discovery. In *Proceedings of the 5th ACM CIKM International Workshop on Web Information and Data Management, WIDM 2003*, pages 15–22, 2003. [76](#)
- [52] Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Clustering Web pages based on their structure. *Data Knowl. Eng.*, 54(3):279–299, 2005. [76](#)
- [53] Thi Kim Nhung Dang, Doina Bucur, Berk Atil, Guillaume Pitel, Frank Ruis, Hamid Reza Kadkhodaei, and Nelly Litvak. Look back, look around: A systematic analysis of effective predictors for new outlinks in focused Web crawling. *Knowl. Based Syst.*, 260:110126, 2023. [75](#), [161](#)
- [54] Brian D. Davison. Topical locality in the Web. In *Proceedings of the 23rd Annual International ACM Conference on Research and Development in Information Retrieval, SIGIR 2000*, pages 272–279, 2000. [26](#), [29](#), [73](#)

-
- [55] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proc. VLDB Endow.*, 17(8):1925–1938, 2024. [25](#)
 - [56] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused Crawling Using Context Graphs. In *Proceedings of 26th International Conference on Very Large Data Bases, VLDB 2000*, pages 527–534, 2000. [26](#), [29](#), [46](#), [47](#), [73](#)
 - [57] Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. Semantic Structure Extraction for Spreadsheet Tables with a Multi-task Learning Architecture. In *Proceedings of the 1st Workshop on Document Intelligence at NeurIPS 2019, DI 2019*, 2019. [111](#), [112](#), [113](#), [115](#)
 - [58] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. Table Sense: Spreadsheet Table Detection with Convolutional Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 69–76, 2019. [23](#), [25](#), [111](#), [112](#), [114](#), [115](#)
 - [59] Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. Encoding Spreadsheets for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pages 20728–20748, 2024. [9](#), [19](#), [23](#), [111](#), [112](#), [114](#), [115](#), [135](#), [136](#)
 - [60] Julian Eberius, Christopher Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner. DeXcelerator: a framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM 2013*, pages 2477–2480, 2013. [115](#)
 - [61] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007. [23](#)
 - [62] Hazem Elmeleegy, Jayant Madhavan, and Alon Y. Halevy. Harvesting Relational Tables from Lists on the Web. *Proc. VLDB Endow.*, 2(1):1078–1089, 2009. [22](#)
 - [63] Muhammad Faheem and Pierre Senellart. Adaptive Web Crawling Through Structure-Based Link Classification. In *Proceedings of the 17th International Conference on Asia-Pacific Digital Libraries, ICADL 2015*, pages 39–51, 2015. [26](#), [29](#), [46](#), [73](#), [74](#), [75](#), [76](#)
 - [64] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.*, 16(7):1726–1739, 2023. [25](#)

- [65] Juliana Freire, Grace Fan, Benjamin Feuer, Christos Koutras, Yurong Liu, Eduardo Peña, Aécio S. R. Santos, Cláudio T. Silva, and Eden Wu. Large Language Models for Data Discovery and Integration: Challenges and Opportunities. *IEEE Data Eng. Bull.*, 49(1):3–31, 2025. 25, 77
- [66] Tim Furche, Giovanni Grasso, Andrey Kravchenko, and Christian Schallhart. Turn the Page: Automated Traversal of Paginated Websites. In *Proceedings of the 12th International Conference on Web Engineering, ICWE 2012*, pages 332–346, 2012. 30
- [67] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979. 32
- [68] Antoine Gauquier and Pierre Senellart. Automatically Inferring the Document Class of a Scientific Article. In *Proceedings of the 2023 ACM Symposium on Document Engineering, DocEng 2023*, pages 11:1–11:10, 2023. 166
- [69] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawler for Scalable Web Data Acquisition. In *40th Annual French Conference on Databases, BDA 2024*, 2024. URL <https://bda2024.sciencesconf.org/resource/page/id/7>. 8, 18
- [70] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient and Focused Web Crawling for Statistical Data Sources Retrieval. In *2nd Workshop “Journées Infox sur Seine”*, 2024. URL <https://infoxsurseine.org/2024/efficient-and-focused-web-crawling-for-statistical-data-sources-retrieval/>. 8, 18
- [71] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. In *41st Annual French Conference on Databases, BDA 2025*, 2025. URL <https://bda2025.sciencesconf.org/resource/page/id/3>. 8, 18
- [72] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Supplementary material for the paper Efficient And Scalable Search For Statistics. https://github.com/AntoineGauquier/efficient_and_scalable_search_for_statistics, 2025. 9, 19, 79
- [73] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Code of the STAR–StatCheck system. <https://gitlab.inria.fr/cedar/star-statcheck>, 2025. 9, 19, 79
- [74] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawling for Scalable Web Data Acquisition (experiment reproducibility kit). https://github.com/AntoineGauquier/efficient_crawler_for_scalable_web_data_acquisition/, 2025. 9, 19, 30

-
- [75] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. TRES System Adapted to the Target Retrieval Task. https://github.com/AntoineGauquier/tres_modified_for_target_retrieval, 2025. 9, 19, 48
- [76] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. Efficient and Scalable Search for Statistics. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*, pages 1113–1126, 2026. 8, 18
- [77] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. STAR: Efficient, Scalable, and Modular Retrieval of Statistical Tables. *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, 2026. 8, 18
- [78] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. In *42nd Annual French Conference on Databases, BDA 2026*, 2026. 8, 18
- [79] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Efficient Crawling for Scalable Web Data Acquisition. In *Proceedings of the 29th International Conference on Extending Database Technology, EDBT 2026*, pages 372–385, 2026. 7, 18
- [80] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Reinforcement Learning-Based Focused Web Crawling for Robust and Scalable Data Acquisition, 2026. [Under Review]. 7, 18
- [81] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management, CIKM 2026*, 2026. 8, 18
- [82] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. TabAgree: An Agreement-Aware Platform for Reliable Spreadsheet Table Annotation, 2026. [Under Review]. 8, 19
- [83] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. StatSheets: dataset and reproducibility kit for spreadsheet table understanding. <https://doi.org/10.5281/zenodo.21967189>, 2026. 9, 19, 113, 126, 136
- [84] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. TabAgree: source code, deployment instructions, sample data, and demonstration video. <https://gitlab.inria.fr/cedar/tabagree>, 2026. 9, 20, 150
- [85] Majid Ghasemi-Gol, Jay Pujara, and Pedro A. Szekely. Tabular Cell Classification Using Pre-Trained Cell Embeddings. In *Proceedings of the 2019 IEEE International Conference on Data Mining, ICDM 2019*, pages 230–239, 2019. 111, 113, 115

- [86] Julius Gonsior, Josephine Rehak, Maik Thiele, Elvis Koci, Michael Günther, and Wolfgang Lehner. Active Learning for Spreadsheet Cell Classification. In *Proceedings of the SEA Data 2020 Workshop of the EDBT/ICDT 2020 Joint Conference*, 2020. 111, 113
- [87] Georges Gouriten, Silviu Maniu, and Pierre Senellart. Scalable, generic, and adaptive systems for focused crawling. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media, HT 2014*, pages 35–45, 2014. 26, 29, 73, 75
- [88] Yan Guo, Kui Li, Kai Zhang, and Gang Zhang. Board Forum Crawling: A Web Crawling Method for Web Forum. In *Proceedings of the 2006 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2006*, pages 745–748, 2006. 75
- [89] Yuxiang Guo, Zhonghao Hu, Yuren Mao, Baihua Zheng, Yunjun Gao, and Mingwei Zhou. Birdie Code Repository. <https://github.com/ZJU-DAILY/BIRDIE>, 2025. 97
- [90] Yuxiang Guo, Zhonghao Hu, Yuren Mao, Baihua Zheng, Yunjun Gao, and Mingwei Zhou. BIRDIE: Natural Language-Driven Table Discovery Using Differentiable Search Index. *Proc. VLDB Endow.*, 18(7):2070–2083, 2025. 25, 77, 78, 79, 82, 90, 97, 99, 100, 108, 150
- [91] Miyoung Han, Pierre-Henri Wuillemin, and Pierre Senellart. Focused Crawling Through Reinforcement Learning. In *Proceedings of the 18th International Conference on Web Engineering, ICWE 2018*, pages 261–278, 2018. 26, 29, 73, 75
- [92] Felienne Hermans and Emerson R. Murphy-Hill. Enron’s Spreadsheets and Related Emails: A Dataset and Analysis. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015*, pages 7–16, 2015. 112, 115
- [93] Inma Hernández, Carlos R. Rivero, and David Ruiz. Deep Web crawling: a survey. *World Wide Web*, 22(4):1577–1610, 2019. 75
- [94] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 4320–4333, 2020. 24, 82, 108, 109
- [95] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. Open Domain Question Answering over Tables via Dense Retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 512–519, 2021. 24, 108, 109, 150
- [96] Madelon Hulsebos, Wenjing Lin, Shreya Shankar, and Aditya G. Parameswaran. It Took Longer than I was Expecting: Why is Dataset Search Still so Hard? In

- Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics, HILDA 24*, pages 1–4, 2024. [25](#)
- [97] Marc Fisher II and Gregg Rothmel. The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *Proceedings of the 1st Workshop on End-User Software Engineering, WEUSE 2005*, pages 10:1–10:5, 2005. [115](#)
- [98] Gautier Izacard and Edouard Grave. Distilling Knowledge from Reader to Retriever for Question Answering. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021*, 2021. [97](#)
- [99] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. 2021. [40](#)
- [100] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2001. [2](#), [12](#), [22](#), [79](#)
- [101] Christian S. Jensen and Richard T. Snodgrass. Temporal Data Management. *IEEE Trans. Knowl. Data Eng.*, 11(1):36–44, 1999. [22](#), [79](#)
- [102] Xingyu Ji, Aditya Parameswaran, and Madelon Hulsebos. TARGET: Benchmarking Table Retrieval for Generative Tasks. In *Proceedings of the 3rd Table Representation Learning Workshop of NeurIPS 2024*. [25](#)
- [103] Yangfan Jiang, Fei Wei, Ergute Bao, Yaliang Li, Bolin Ding, Yin Yang, and Xiaokui Xiao. Accurate Table Question Answering with Accessible LLMs. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*, pages 169–182, 2026. [162](#)
- [104] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Mach. Learn.*, 77(1):27–59, 2009. [120](#)
- [105] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data*, 7(3):535–547, 2021. [97](#)
- [106] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502, 2004. [96](#)
- [107] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification. *Proc. VLDB Endow.*, 13(11):2508–2521, 2020. [26](#), [109](#)
- [108] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. Scrutinizer: Fact Checking Statistical Claims. *Proc. VLDB Endow.*, 13(12):2965–2968, 2020. [26](#), [109](#)

- [109] Pratik Karmakar, Antoine Gauquier, and Pierre Senellart. Expected Shapley Value is Shapley Value for Expected Utility Game. In *Proceedings of the 18th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2025*, pages 223–237, 2025. [167](#)
- [110] George Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-SQL. *VLDB J.*, 32(4):905–936, 2023. [25](#), [108](#)
- [111] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the 30th Conference on Neural Information Processing Systems, NIPS 2017*, pages 3146–3154, 2017. [119](#)
- [112] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Mach. Learn.*, 80:245–272, 2010. [39](#), [76](#)
- [113] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. A Machine Learning Approach for Layout Inference in Spreadsheets. In *Proceedings of the 8th International Conference on Knowledge Discovery and Information Retrieval, KDIR 2016*, pages 77–88, 2016. [23](#), [111](#), [113](#), [116](#), [129](#), [130](#), [151](#)
- [114] Elvis Koci, Maik Thiele, Wolfgang Lehner, and Oscar Romero. Table Recognition in Spreadsheets via a Graph Representation. In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems, DAS 2018*, pages 139–144, 2018. [111](#), [112](#), [114](#), [115](#)
- [115] Elvis Koci, Dana Kuban, Nico Luetting, Dominik Olwig, Maik Thiele, Julius Gonsior, Wolfgang Lehner, and Oscar Romero. XLIndy: Interactive Recognition and Information Extraction in Spreadsheets. In *Proceedings of the 19th ACM Symposium on Document Engineering, DocEng 2019*, pages 25:1–25:4, 2019. [115](#)
- [116] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. DECO: A Dataset of Annotated Spreadsheets for Layout and Table Recognition. In *Proceedings of the 15th International Conference on Document Analysis and Recognition, ICDAR 2019*, pages 1280–1285, 2019. [25](#), [112](#), [115](#)
- [117] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. A Genetic-Based Search for Adaptive Table Recognition in Spreadsheets. In *Proceedings of the 15th International Conference on Document Analysis and Recognition, ICDAR 2019*, pages 1274–1279, 2019. [111](#), [112](#), [114](#), [115](#)
- [118] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004. [120](#)
- [119] Andrey Kolobov, Yuval Peres, Cheng Lu, and Eric Horvitz. Staying up to Date with Online Content Changes Using Reinforcement Learning for Scheduling. In

- Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS 2019*, pages 579–589, 2019. [75](#)
- [120] Andrey Kolobov, Yuval Peres, Eyal Lubetzky, and Eric Horvitz. Optimal Freshness Crawl Under Politeness Constraints. In *Proceedings of the 42nd International ACM Conference on Research and Development in Information Retrieval, SIGIR 2019*, pages 495–504, 2019. [75](#), [161](#)
- [121] Andreas Kontogiannis, Dimitrios Kelesis, Vasilis Pollatos, Georgios Paliouras, and George Giannakopoulos. Tree-based Focused Web Crawling with Reinforcement Learning. *arXiv*, abs/2112.07620, 2021. [9](#), [19](#), [26](#), [29](#), [47](#), [48](#), [73](#), [75](#)
- [122] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A Large Public Corpus of Web Tables containing Time and Context Metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web, WWW 2016*, pages 75–76, 2016. [25](#)
- [123] Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM Already Serve as A Database Interface? A BIG Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems, NeurIPS 2023*, 2023. [25](#), [108](#)
- [124] Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adrià de Gispert, and Gonzalo Iglesias. An Inner Table Retriever for Robust Table Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 9909–9926, 2023. [109](#)
- [125] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024. [83](#)
- [126] Yang Liu, Mengyi Yan, Jiao Xue, Weilong Ren, Yutong Ye, Haoyi Zhou, Zhumin Chen, and Jianxin Li. SPARQ: A Cost-Efficient Framework for Offline Table Question Answering via Adaptive Routing. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*, pages 2668–2682, 2026. [162](#)
- [127] Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. Web-scale Knowledge Collection. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining, WSDM 2020*, pages 888–889, 2020. [23](#)
- [128] Yury A. Malkov and Dmitry A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020. [37](#)

- [129] Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. <https://ai.meta.com/blog/meta-llama-3/>, 2024. 97
- [130] Robert Meusel, Peter Mika, and Roi Blanco. Focused Crawling for Structured Data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014*, pages 1039–1048, 2014. 26, 29, 73, 74, 75
- [131] Gengxin Miao, Jun’ichi Tatemura, Wang-Pin Hsiung, Arsany Sawires, and Louise E. Moser. Extracting data records from the web using tag path clustering. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009*, pages 981–990, 2009. 30
- [132] Microsoft. Official Code Release for TUTA Table Understanding. https://github.com/microsoft/TUTA_table_understanding/tree/main/tuta, 2021. 130
- [133] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations, ICLR 2013*, 2013. 91
- [134] Shrey Mishra, Yacine Brihmouche, Theo Delemazure, Antoine Gauquier, and Pierre Senellart. First Steps in Building a Knowledge Base of Mathematical Results. In *Proceedings of the 4th Workshop on Scholarly Document Processing, SDP 2024*, pages 165–174, 2024. 166, 167
- [135] Shrey Mishra, Antoine Gauquier, and Pierre Senellart. Modular Multimodal Machine Learning for Extraction of Theorems and Proofs in Long Scientific Documents. In *Proceedings of the 24th ACM/IEEE Joint Conference on Digital Libraries, JCDL 2024*, pages 14:1–14:5, 2024. 167
- [136] Shrey Mishra, Neil Sharma, Antoine Gauquier, and Pierre Senellart. TheoremView: A Framework for Extracting Theorem-Like Environments from Raw PDFs. In *Proceedings of the 47th European Conference on Information Retrieval (ECIR)*, 2025. 166, 167
- [137] Gordon Mohr, Michael Stack, Igor Rnıtovic, Dan Avery, and Michele Kimpton. Introduction to Heritrix. In *IWAW*, 2004. 75
- [138] Andreas C. Muller and Sven Behnke. PyStruct: learning structured prediction in python. *J. Mach. Learn. Res.*, 15(1):2055–2060, 2014. 119
- [139] Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryscinski, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir R. Radev. FeTaQA: Free-form Table Question Answering. *Trans. Assoc. Comput. Linguistics*, 10:35–49, 2022. 25, 150, 151

-
- [140] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. Table Union Search on Open Data. *Proc. VLDB Endow.*, 11(7):813–825, 2018. 24
- [141] Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. SemTab 2021: Tabular Data Annotation with MTab Tool. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference, ISWC 2021*, pages 92–101, 2021. 151
- [142] OpenAI. Documentation of the GPT 3.5 Turbo model. <https://platform.openai.com/docs/models/gpt-3.5-turbo>, . 98
- [143] OpenAI. Documentation of the GPT 4 “Omni” model. <https://platform.openai.com/docs/models/gpt-4o>, . 98
- [144] Taylor Orth. From millionaires to Muslims, small subgroups of the population seem much larger to many Americans. <https://today.yougov.com/politics/articles/41556>, 2022. 1, 11
- [145] Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. Dataset Discovery and Exploration: A Survey. *ACM Comput. Surv.*, 56(4):102:1–102:37, 2024. 25, 108
- [146] Pew Research Center. Mobile Fact Sheet. <https://www.pewresearch.org/internet/fact-sheet/mobile/>, 2024. 80
- [147] Kien Pham, Aécio S. R. Santos, and Juliana Freire. Learning to Discover Domain-Specific Web Content. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining, WSDM 2018*, pages 432–440, 2018. 26, 75, 161
- [148] Rakesh Pimplikar and Sunita Sarawagi. Answering Table Queries on the Web using Column Keywords. *Proc. VLDB Endow.*, 5(10):908–919, 2012. 24
- [149] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1994. 36
- [150] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001. 23
- [151] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 3980–3990, 2019. 98
- [152] Carsten Rother, Vladimir Kolmogorov, Victor S. Lempitsky, and Martin Szummer. Optimizing Binary MRFs via Extended Roof Duality. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2007*, 2007. 119, 120

- [153] Mohammed Saeed and Paolo Papotti. Fact-Checking Statistical Claims with Tables. *IEEE Data Eng. Bull.*, 44(3):27–38, 2021. 26
- [154] Peter Schulam and Ion Muslea. Improving the Exploration/Exploitation Trade-Off in Web Content Discovery. In *Companion Proceedings of the ACM Web Conference 2023, WWW 2023*, pages 1183–1189, 2023. 75, 161
- [155] SDMX Community. SDMX Technical Specifications. https://sdmx.org/?page_id=5008, 2024. 2, 12, 22, 29, 79
- [156] Pierre Senellart. Identifying Websites with Flow Simulation. In *Proceedings of the 5th International Conference on Web Engineering, ICWE 2005*, pages 124–129, 2005. 26, 33
- [157] Victor S. Sheng, Foster J. Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008*, pages 614–622, 2008. 162
- [158] Solomon Eyal Shimony. Finding MAPs for Belief Networks is NP-Hard. *Artif. Intell.*, 68(2):399–410, 1994. 119
- [159] Roei Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. Web Table Retrieval using Multimodal Deep Learning. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, pages 1399–1408, 2020. 25, 82, 108
- [160] Kristinn Sigurdsson. Incremental crawling with Heritrix. In *IWAW*, 2005. 75
- [161] Marijan Sorić, Cécile Gracianne, Ioana Manolescu, and Pierre Senellart. Benchmarking Table Extraction from Heterogeneous Scientific PDF Documents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2026*, 2026. 3, 13, 65
- [162] Jannik Strötgen and Michael Gertz. HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010*, pages 321–324, 2010. 85
- [163] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 697–706, 2007. 23
- [164] Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. TAP4LLM: Table Provider on Sampling, Augmenting, and Packing Semi-structured Data for Large Language Model Reasoning. In *Findings of the Association for Computational Linguistics, EMNLP 2024*, pages 10306–10323, 2024. 24, 109

-
- [165] Kexuan Sun, Harsha Rayudu, and Jay Pujara. A Hybrid Probabilistic Approach for Table Understanding. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 4366–4374, 2021. 111, 112, 113
 - [166] Yibo Sun, Zhao Yan, Duyu Tang, Nan Duan, and Bing Qin. Content-based table retrieval for web queries. *Neurocomputing*, 349:183–189, 2019. 108
 - [167] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018. 35, 38, 76
 - [168] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 809–819, 2018. 26
 - [169] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large Margin Methods for Structured and Interdependent Output Variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005. 120
 - [170] VIGINUM. RRN: A Complex and Persistent Information Manipulation Campaign, 2023. https://www.sgdsn.gouv.fr/files/files/Publications/20230719_NP_VIGINUM_RAPPORT-CAMPAGNE-RRN_EN.pdf. 2, 12
 - [171] Gerardo Vitagliano, Lan Jiang, and Felix Naumann. Detecting Layout Templates in Complex Multiregion Files. *Proc. VLDB Endow.*, 15:646–658, 2021. 65, 111, 114, 135
 - [172] Gerardo Vitagliano, Lucas Reisener, Lan Jiang, Mazhar Hameed, and Felix Naumann. Official Code Release for Mondrian. <https://github.com/HPI-Information-Systems/Mondrian>, 2021. 135
 - [173] Andreas Vlachos and Sebastian Riedel. Identification and Verification of Simple Claims about Statistical Properties. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 2596–2601, 2015. 26
 - [174] Qiming Wang and Raul Castro Fernandez. Solo Code Repository. <https://github.com/thedatastation/solo>, 2023. 97
 - [175] Qiming Wang and Raul Castro Fernandez. Solo: Data Discovery Using Natural Language Questions Via A Self-Supervised Approach. *Proc. ACM Manag. Data*, 1(4):262:1–262:27, 2023. 25, 77, 78, 79, 82, 90, 97, 100, 108
 - [176] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2021*, pages 1780–1790, 2021. 111, 113, 115, 130

- [177] Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *Proceedings of the 12th International Conference on Learning Representations, ICLR 2024*, 2024. 24, 108
- [178] Dimitri Watel and Marc-Antoine Weisser. A practical greedy approximation for the directed Steiner tree problem. *J. Comb. Optim.*, 32(4):1327–1370, 2016. 32
- [179] Gerhard Weikum, Xin Luna Dong, Simon Razniewski, and Fabian M. Suchanek. Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases. *Found. Trends Databases*, 10(2-4):108–490, 2021. 23
- [180] WHATWG. DOM: Living Standard. <https://dom.spec.whatwg.org/>, 2024. 34
- [181] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the 11th International Conference on Learning Representations, ICLR 2023*, 2023. 109
- [182] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large Language Models are Versatile Decomposers: Decomposing Evidence and Questions for Table-based Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023*, pages 174–184, 2023. 24, 108
- [183] Haoxiang Zhang, Aécio S. R. Santos, and Juliana Freire. DSDD: Domain-Specific Dataset Discovery on the Web. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management, CIKM 2021*, pages 2527–2536, 2021. 26, 73, 74, 75, 163
- [184] Haoxiang Zhang, Yurong Liu, Aécio Santos, Wei-Lun Hung, and Juliana Freire. AutoDDG: Automated Dataset Description Generation using Large Language Models. *Proc. ACM Manag. Data*, 4(1), 2026. 25
- [185] Shuo Zhang and Krisztian Balog. Ad Hoc Table Retrieval using Semantic Similarity. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*, pages 1553–1562, 2018. 24, 108
- [186] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. TableLlama: Towards Open Large Generalist Models for Tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2024*, pages 6024–6044, 2024. 24, 108
- [187] Yunjia Zhang, Jordan Henkel, Avriila Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. ReAcTable: enhancing ReAct for Table Question Answering. *Proc. VLDB Endow.*, 17(8):1981–1994, 2024. 24, 109

RÉSUMÉ

Les données statistiques publiées par les gouvernements, les instituts statistiques et les organisations internationales constituent un bien commun : elles éclairent le débat public, soutiennent la recherche et servent de fondement au fact-checking. Leur abondance rend toutefois leur exploitation difficile. Dispersées sur des centaines de sites Web, publiées dans des formats hétérogènes et organisées pour des lecteurs humains, elles sont souvent peu exploitables par les machines. Les rendre exploitables implique de résoudre une chaîne de problèmes : localiser les données sur le Web, en reconstruire la structure, puis retrouver les tableaux pertinents pour répondre à une question donnée. Cette thèse traite plusieurs maillons de cette chaîne en se concentrant sur les tableaux statistiques, une catégorie de données tabulaires à la fois multidimensionnelle et relativement peu étudiée.

Nous présentons quatre contributions. Premièrement, pour l'acquisition des données, nous formalisons le crawl ciblé d'un site Web comme un problème d'exploration de graphe, en démontrons sa NP-difficulté et proposons SB-CLASSIFIER, un crawler guidé par l'apprentissage par renforcement. Celui-ci apprend quels hyperliens mènent aux fichiers cibles et en récupère une grande partie tout en ne visitant qu'une faible portion du site. Deuxièmement, pour la recherche de tableaux, nous définissons le problème de découverte de tableaux statistiques et présentons STAR, une méthode qui indexe séparément le contenu temporel, spatial et linguistique des tableaux et des requêtes. STAR surpasse les systèmes génériques de découverte de tableaux tout en passant à l'échelle sur de grands corpus sans GPU. Troisièmement, pour la reconstruction de structure, nous proposons une chaîne de compréhension des feuilles de tableur couvrant la classification du type des cellules et l'extraction de tableaux. Elle s'exécute sur une seule machine sans GPU tout en demeurant compétitive avec des systèmes bien plus coûteux, fondés sur des transformeurs ou des LLM. Quatrièmement, pour l'annotation, nous développons TABAGREE, une plateforme permettant de produire à grande échelle des annotations fiables pour ces tâches et d'exploiter le désaccord entre annotateurs comme un signal interprétable. Nous publions deux jeux de données construits avec TABAGREE, dédiés à la découverte de tableaux statistiques et à la compréhension des feuilles de tableur.

Nos méthodes combinent efficacité, sobriété matérielle et passage à l'échelle. Robustes à l'hétérogénéité des données du monde réel, elles rendent possible l'acquisition, la compréhension et la recherche efficaces de données statistiques ouvertes à grande échelle.

MOTS CLÉS

Données statistiques ouvertes ★ Données tabulaires ★ Crawl ciblé du Web ★ Découverte de tableaux ★ Compréhension des tableurs ★ Annotation de données ★ Gestion de données à grande échelle

ABSTRACT

Statistical data published by governments, statistical institutes, and international organizations are a public good: they inform public debate, support research, and provide the foundation for journalistic fact-checking. Yet their abundance is also what makes it hard to use. Statistics are scattered across hundreds of websites, published in heterogeneous formats, and organized for human readers rather than machines, so that the data, though available, are rarely usable in practice. Making it usable is not a single problem but a chain of them, from finding the data on the Web, to recovering its structure, to retrieving the few tables that answer a given question. This thesis addresses several links of that chain, with a particular focus on statistical tables, a multidimensional and comparatively neglected category of tabular data.

We make four contributions. First, for data acquisition, we formalize focused crawling within a website as a graph exploration problem, prove its NP-hardness, and propose SB-CLASSIFIER, a reinforcement-learning crawler that learns which hyperlinks lead to target files and retrieves a large fraction of them while visiting only a small part of the website. Second, for retrieval, we define Statistical Table Discovery and present STAR, a method that indexes the temporal, spatial, and remaining content of tables and queries separately, outperforming general-purpose table discovery systems while scaling to large corpora without a GPU. Third, for structure recovery, we propose an end-to-end spreadsheet understanding pipeline for cell-type classification and table extraction that runs on a single machine without a GPU, yet remains competitive with much heavier Transformer- and LLM-based systems. Fourth, for annotation, we build TABAGREE, an agreement-aware platform that produces reliable labels for these tasks at scale and treats inter-annotator disagreement as an interpretable signal. We also release two public benchmarks built with this platform, on statistical table discovery and on spreadsheet understanding.

Across these contributions, our methods are efficient on modest hardware, robust to the heterogeneity of real data, and scalable to the volumes reached by Web-scale statistical corpora, showing that open statistical data can be acquired, understood, and retrieved efficiently and at scale.

KEYWORDS

Open Statistical Data ★ Tabular Data ★ Focused Web Crawling ★ Table Discovery ★ Spreadsheet Understanding ★ Data Annotation ★ Scalable Data Management