Università degli Studi di Milano Bicocca

Department of Computer Science, Systems and Communication

PhD program in Computer Science Cycle XXXVI

Semantic Enrichment of Tabular Data with Machine Learning Techniques

Supervisor:
Prof. Flavio De Paoli

Tutor:
Prof. Daniela Micucci

Candidate:
Roberto Avogadro

Coordinator: Prof. Leonardo Mariani

Contents

Abstract 1

Acknowledgments 2

1 Introduction 4
- 1.1 Contributions . . . 11
- 1.2 Thesis Structure . . . 12
- 1.3 Reproducibility . . . 14

2 Preliminaries 15
- 2.1 Knowledge Graph . . . 15
- 2.2 Information Retrieval . . . 17
- 2.3 Tables . . . 17
  - 2.3.1 Simple Table . . . 18
  - 2.3.2 Vertical Table . . . 19
  - 2.3.3 Nested Table . . . 19
  - 2.3.4 Relational Tables . . . 19
- 2.4 Semantic Table Interpretation (STI) . . . 20
- 2.5 Neural Networks . . . 22
  - 2.5.1 Structure of Neural Networks . . . 22
  - 2.5.2 Training of Neural Networks . . . 22
  - 2.5.3 Applications of Neural Networks . . . 23
  - 2.5.4 Diverse Types of Neural Networks: An Overview . . . 23
  - 2.5.5 Delving into Feedforward Neural Networks . . . 23
- 2.6 Data Pipelines . . . 24
  - 2.6.1 Components of a Data Pipeline . . . 24
  - 2.6.2 Types of Data Pipelines . . . 24
  - 2.6.3 Importance of Data Pipelines . . . 25
- 2.7 Data Pipeline Example . . . 25

3 Related work (SoTa) 26
- 3.1 Introduction . . . 26
- 3.2 ER: Entity Retrieval . . . 27
- 3.3 EL: Entity Linking . . . 31
- 3.4 NIL Entities Problem . . . 33
- 3.5 Human-in-the-Loop . . . 35
  - 3.5.1 Introduction . . . 35
  - 3.5.2 Research Trends and Classification . . . 36

3.5.3 Types of Interactions 36
3.5.4 Challenges and Limitations 36
3.5.5 Conclusion 37
3.6 Data Pipelines 37
3.6.1 Scalability in Data Processing 38
3.6.2 Architecture and Components 38
3.6.3 Technologies, Methodologies, and Tools in Data Pipelines 39
3.6.4 Challenges and Limitations 40
3.6.5 Summary 40

4 Information Retrieval System **41**
4.1 Knowledge Graph Indexing 41
4.2 Data Retrieval 42
4.3 Information Retrieval System for Dataset with LamAPI 43
4.4 Evaluation of LamAPI Performance 49

5 Semantic Enrichment of Tabular Data **54**
5.1 Overview of the Proposed Approach 54
5.1.1 Data Preparation & Candidate Retrieval 55
5.1.2 The Entity Linking Algorithm 56
5.1.3 Decision 56
5.2 Entity Linking over Tabular Data 57
5.2.1 Feature Engineering 57
5.2.2 The Machine Learning Model 62
5.2.3 Decision: Uncertainty Estimation and Metrics 64
5.3 Implementation and Evaluation of the Algorithms 66
5.3.1 The Experimental Campaign 68
5.3.2 Examining Variations in Scores 70
5.3.3 Further Discussion and Analyses of Score Distributions 71
5.3.4 Benchmarking Tools Across Tables 73

6 Human In The Loop (HITL) over Tabular Data **79**
6.1 Interactive Human Revision 79
6.1.1 User Review through a Jupyter Notebook 80
6.1.2 User Review through a User Interface 80
6.2 Types of Human Feedback 86
6.2.1 Human's Rules 86
6.2.2 Decision Rule Revision 87
6.2.3 Model Revision 87
6.3 Feedback from Human Revision 87
6.4 Validation of Human Feedback 88

7 Entity Linking in Large-Scale Data Environments **95**
7.1 Data Pipeline: An Overview 95
7.2 Data pipelines 96
7.3 Entity Linking at scale using Alligator Scalable Version 99
7.4 Evaluation of Alligator Scalable Version 102
7.4.1 Evaluating Pipeline Stepwise Time Performance 104
7.4.2 Collection of Metrics for Pipeline Performance Evaluation 105
7.4.3 Assessing Pipeline Scalability 105

8 Conclusion and Future Directions **108**

Appendices **109**

A First Appendix **110**
A.1 F1 score and AUC curve for the all Datasets 110
A.2 Examining Variations in Scores over all Datasets 110
A.3 Examining Variations in Scores using Mammotab10K 113
A.4 Further Discussion and Analyses of Score Distributions 114

List of Figures

1.1 Motivations and research value of semantic table interpretation. 5
1.2 ChatGPT 4 output to request table annotation using Wikidata. 9
1.3 ChatGPT 4 output to request table annotation using Wikipedia. 10

2.1 An example of KG 16
2.2 A subset of wikidata Knowledge Graph (KG) 17
2.3 Components of an information retrieval system 18
2.4 Skeleton of a table 18
2.5 An example of annotated table 21
2.6 A typical structure of a Feedforward Neural Network 24

3.1 Number of publications of paper related to Semantic Table Interpretation (STI) [46]. 32
3.2 Distribution of the topics of the papers related to STI [46]. 32
3.3 Number of publications of paper related to NIL entities 33
3.4 The increasing research interest in the human-in-the-loop (image from [77]) . . 36

4.1 LamAPI documentation page 43
4.2 LamAPI Lookup service 44
4.3 The high level workflow to build LamAPI 49
4.4 Coverage trends across different considered datasets 52
4.5 Average position of the correct candidate across different considered datasets . 53

5.1 Linking workflow with Human-In-The-Loop feedback. 54
5.2 Example of feature vector. 58
5.3 Overview of the Alligator Swagger Page 67
5.4 Focus on the POST Method for Performing Entity Linking (EL) on Tabular Data in Alligator 67
5.5 Scores variations for *HardTableR2* dataset 76
5.6 Distribution of ρ and δ over *HardTableR2* 77
5.7 Benchmark comparison of various solutions across four distinct tables 78

6.1 Reconciled table displaying only Wikidata QIDs. 82
6.2 Reconciled table augmented with additional metadata: name, description, and confidence score. 83
6.3 Display list of the tables within a dataset. For each table, it is possible to see some statistics like the number of columns and rows. 84
6.4 Display page with content of a table. 84
6.5 Detailed page for the analysis result of reconciling a mention. 85

6.6 Page for downloading annotations in different formats. 85
6.7 Annotated table without threshold adjustment. 90
6.8 Annotated table with threshold value highlighted. 91
6.9 Annotated table after the application of the adjusted threshold. 92
6.10 F1 and AUC computed for the training dataset. 93
6.11 F1 and AUC computed for the test dataset. 93

7.1 Architecture of enRichMyData Toolbox 98
7.2 Example of data pipeline combining different tools 101
7.3 Representation of the Alligator pipeline's steps as visualized in the Argo Workflows interface. 103
7.4 Representation of the Alligator pipeline's steps as observed from the command-line interface (CLI). 103
7.5 Temporal distribution of each step in the Alligator pipeline, as visualized in the Argo Workflows user interface. 104
7.6 Central Processing Unit (CPU) utilization during the complete dry run of the Alligator pipeline. 105
7.7 Memory utilization during the complete dry run of the Alligator pipeline. . . . 106
7.8 Network bandwidth utilization during the complete dry run of the Alligator pipeline. 106

A.1 F1 and AUC computed for train *Round1_T2D* and *Round3* 110
A.2 F1 and AUC computed for train *2T-2020* and *Round4* 111
A.3 F1 and AUC computed for train *HardTableR2* and *HardTableR3* 111
A.4 F1 and AUC computed for test *Round1_T2D* and *Round3* 111
A.5 F1 and AUC computed for test *2T-2020* and *Round4* 112
A.6 F1 and AUC computed for test *HardTableR2* and *HardTableR3* 112
A.7 Scores variations for Round1_T2D dataset 115
A.8 Scores variations for Round3 dataset 116
A.9 Scores variations for Round4 dataset 117
A.10 Scores variations for 2T-2020 dataset 118
A.11 Scores variations for HardTableR3 dataset 119
A.12 Scores variations for Mammotab10K dataset using the model from Round1_T2D dataset 120
A.13 Scores variations for Mammotab10K dataset using model from Round3 dataset 121
A.14 Scores variations for Mammotab10K dataset using model from Round4 dataset 122
A.15 Scores variations for Mammotab10K dataset using model from 2T-2020 dataset 123
A.16 Scores variations for Mammotab10K dataset using model from HardTableR2 dataset 124
A.17 Scores variations for Mammotab10K dataset using model from HardTableR3 dataset 125

A.18 Distribution of ρ and δ over *Round1_T2D dataset* 126
A.19 Distribution of ρ and δ over *Round3* 126
A.20 Distribution of ρ and δ over *Round4* 128
A.21 Distribution of ρ and δ over *2T-2020* 128
A.22 Distribution of ρ and δ over *HardTableR3* 129

Abstract

Semantic Table Interpretation (STI) is one of the most widely used methods for identifying entities in tabular data. In this work, a methodology is delineated for implementing entity linking on a large scale using machine learning techniques, focusing on the challenges and solutions associated with handling vast amounts of data. The methodology is based on the concept of table-to-Knowledge Graph (KG) matching, which is a key step to enrich and extend Knowledge Graphs (KGs) from semi-structured data. Moreover, the intricacies of STI, EL, and the emerging landscape of Large Language Models (LLMs) and their potential application in STI and EL have also been touched upon. In addition, the impact of employing distinct KnowledgeGraphs, such as Wikidata and Wikipedia, in the context of EL using ChatGPT 4 has been demonstrated. In particular, the role of Human-in-the-Loop (HITL) techniques in enhancing model performance is explored. This work outlines the foundational groundwork for the development of a scalable approach that adapts the existing STI framework to handle enrichment pipelines in large-scale data scenarios, thereby enhancing its operational efficiency and applicability.

Acknowledgments

This research journey began during my Bachelor's degree and continued throughout my PhD, spanning many years of dedicated focus on this topic. Over time, I have cultivated a deep expertise in this area, thoroughly enjoying the process of delving into the intricate challenges it presents.

I would like to express my gratitude to my supervisor, Prof. Flavio De Paoli. Special thanks go to Prof. Michele Ciavotta and Prof. Matteo Palmonari, who have been excellent reference points for me. I am also grateful to Prof. Dumitru Roman, who supervised my work during my time abroad at SINTEF Digital in Oslo, Norway — a division of the largest independent research organization in Scandinavia. His ability to stimulate my intellectual curiosity was invaluable.

My heartfelt appreciation goes to my family, sisters, and friends, who have provided unwavering support throughout my academic journey.

Equally important, I wish to acknowledge the contributions of my colleagues in "the lab." Special thanks go to Marco Cremaschi for his consistent assistance, to David Chieregato for his patience and invaluable advice, and to Fabio D'Adda for being a pleasant companion during break times. I also extend my gratitude to Elia Guarnieri, aptly described as 'the teacher,' for his pedagogical contributions. Lastly, I thank Stefano Fiorini and Riccardo Pozzi for being not only thoughtful collaborators in academic discussions but also enjoyable companions outside of the work environment.

This work received partial funding from *InterTwino* - National programme "European Research Networks" by the Bulgarian Ministry of Education and Science - (grant number O1-89/02.04.2021), and the European Commission Horizon 2020 enRichMyData project (grant number 101070284).

Acronyms

AI Artificial Intelligence.

CEA Cell Entity Annotation.

CPA Column Property Annotation.

CR Candidate Retrieval.

CTA Column Type Annotation.

ED Entity Disambiguation.

EL Entity Linking.

IR Information Retrieval.

KG Knowledge Graph.

KGs Knowledge Graphs.

LLM Large Language Model.

ML Machine Learning.

NLP Natural Language Processing.

RDF Resource Description Format.

STI Semantic Table Interpretation.

UI User Interface.

1. Introduction

In the era of information, the large amounts of available data represent both a challenge and an opportunity for organizations and researchers. These structured and unstructured data can be used to create supervised ML models within the realm of Artificial Intelligence (AI), which extract knowledge by identifying patterns and relationships in the data. However, to build effective models, it is important that the data is of good quality. Organizations must therefore pay attention to data collection and management, with the ultimate goal of obtaining accurate and reliable datasets.

In 2023, OpenAI released GPT-4, a large-scale multimodal language model trained using an unprecedented scale of compute and data. Unlike GPT-3.5, this new version is more reliable, creative, and capable of handling more complex user instructions. Although it does not yet consider many real-world scenarios, GPT-4 shows performance comparable to that of a human in various domains.

The construction and use of structured data in tables is common in multiple contexts; it is worth noting that in the current version of Wikipedia, it is possible to identify 2,803,424 tables [49]. This suggests that tables convey a huge amount of information that can be used in data analytics or AI solutions.

The large amount of already available tables and datasets makes interoperability strategic, and an answer is to identify the same entities across disparate datasets. The identification process is known as the table-to-Knowledge Graph (KG) matching problem also referred to as Semantic Table Interpretation (STI), which has recently received much attention in the research community [38, 34, 12]. This is a key step to enrich data [13, 55] and construct and extend Knowledge Graphs (KGs) from semi-structured data [75, 41]. The knowledge of entities with their types and properties allows for further operations as illustrated in Figure 1.1. In this work we concentrate on data enrichment, which means being able to interpret (reconcile) a table using a KG to enable for extension with information taken from the KG end/or from other tables.

STI is a research field in continuous evolution with increasing interest over time. The growing interest in the topic is proved by the international *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* (SemTab) that has been proposed since 2018[1] [38, 34, 12]. This challenge is repeated annually. Over the years, several approaches, datasets, and related Gold Standards have been released. The outcomes of the various editions of the challenge, including datasets with associated Gold Standard and the methods developed by participants, provide the valuable framework to support and validate the work proposed in this thesis.

An example of a tabular data source is presented in Table 1.1. This table can be reconciled against Wikidata as illustrated in Table 1.2. Wikidata is the largest collaboratively-edited database of structured data that serves as a repository of general-purpose linked data. The reconciliation process added Wikidata unique identifiers to movie titles, directors, and distributors,

[1] cs.ox.ac.uk/isg/challenges/sem-tab

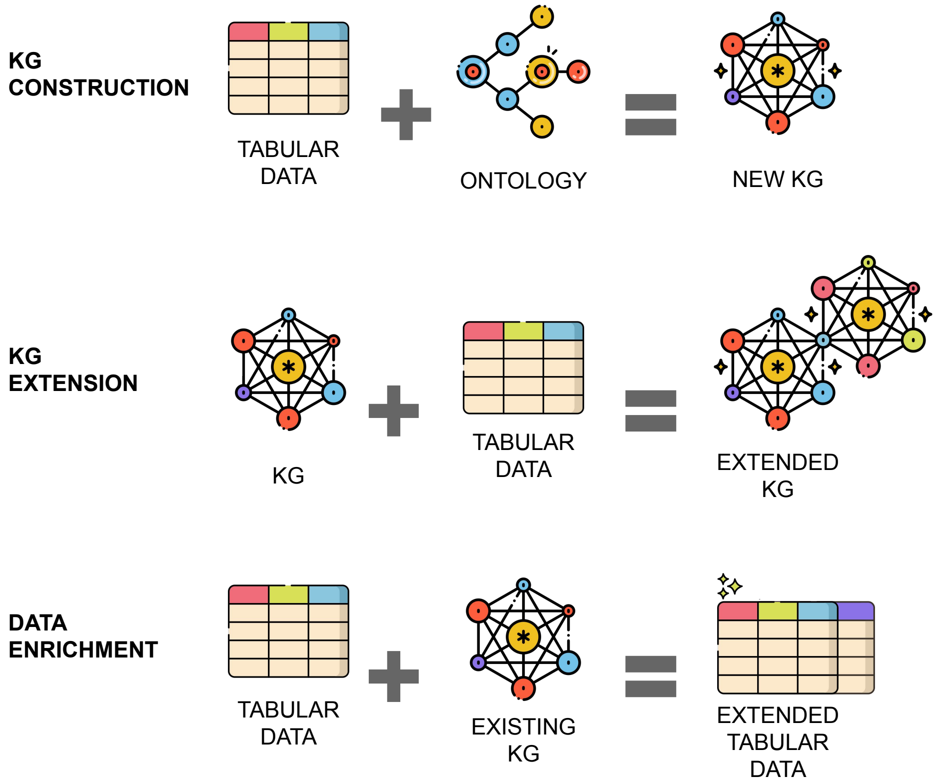


Figure 1.1: Motivations and research value of semantic table interpretation.

Table 1.1: An example of tabular data about movies

Title	**Director**	**Release date**	**Distributor**	**Length in min**	**Worldwide gross (USD)**
X-Men	Bryan Singer	14/07/2000	Twentieth Century Fox	104	296300000
Batman Begins	Christopher Nolan	17/06/2005	Warner Bros.	140	371853783
Superman Returns	Bryan Singer	28/06/2006	Warner Bros.	154	391081192
Avatar	James Cameron	15/01/2009	Twentieth Century Fox	162	2744336793
Jurassic World	Colin Trevorrow	11/06/2015	Universal Pictures	124	1670400637

which enables for extensions as illustrated in Table 1.3. IMDb identifiers (with gray background in the figure) are taken from Wikidata and used to extract further information, the genre in this case (with pink background). IMDb (Internet Movie Database) is an example of a domain-specific dataset that defines unique identifies without being a KG.

The example illustrates the effect of executing an *enrichment pipeline* that brings a table and, after the application of a sequence of algorithms, produces a final result, the enriched table, as the one in Table 1.3. The overall approach taken in this work foresees two phases: a preliminary phase, and a production phase. In the former, a sample table is enriched by manually applying algorithms and involving datasets to set up an enrichment pipeline, i.e., the sequence of actions to be applied to get the final enriched table. In the latter, the defined pipeline is executed on the full dataset at scale.

Table 1.2: An example of tabular data about movies (annotated using Wikidata)

Title	Tittle (QID)	Director	Director (QID)	Release date	Distributor	Distributor (QID)	Length in min	Worldwide gross (USD)
X-Men	Q106182	Bryan Singer	Q220751	14/07/2000	Twentieth Century Fox	Q434841	104	296,300,000
Batman Begins	Q166262	Christopher Nolan	Q25191	17/06/2005	Warner Bros.	Q126399	140	371,853,783
Superman Returns	Q166262	Bryan Singer	Q220751	28/06/2006	Warner Bros.	Q126399	154	391,081,192
Avatar	Q24871	James Cameron	Q42574	15/01/2009	Twentieth Century Fox	Q434841	162	2,744,336,793
Jurassic World	Q3512046	Colin Trevorrow	Q5145625	11/06/2015	Universal Pictures	Q168383	124	1,670,400,637

Table 1.3: An example of tabular data about movies (extended using IMDb)

Title	Tittle (QID)	Director	Director (QID)	Release date	Distributor	Distributor (QID)	Length in min	Worldwide gross (USD)	IMDB ID	Genre
X-Men	Q106182	Bryan Singer	Q220751	14/07/2000	Twentieth Century Fox	Q434841	104	296300000	tt0120903	Action, Adventure, Sci-Fi
Batman Begins	Q166262	Christopher Nolan	Q25191	17/06/2005	Warner Bros.	Q126399	140	371853783	tt0372784	Action, Crime, Drama
Superman Returns	Q166262	Bryan Singer	Q220751	28/06/2006	Warner Bros.	Q126399	154	391081192	tt0348150	Action, Adventure, Sci-Fi
Avatar	Q24871	James Cameron	Q42574	15/01/2009	Twentieth Century Fox	Q434841	162	2744336793	tt0499549	Action, Adventure, Fantasy
Jurassic World	Q3512046	Colin Trevorrow	Q5145625	11/06/2015	Universal Pictures	Q168383	124	1670400637	tt0369610	Action, Adventure, Sci-Fi

Most of the approaches proposed in the literature for entity linking and table interpretation are based on the use of heuristics, which are techniques that involve the definition of matching rules and similarity measures that are combined linearly or with different weights. These approaches are often manually adapted through the analysis of results or comparison with a Gold Standard to annotate data from specific domains or cases. However, the inherent specificity of these approaches gives rise to biases, limiting their ability to generalize effectively.

In general, the robustness and generalization of heuristic-based approaches are difficult to assess, as it is not possible to understand how the approaches were adapted to achieve better performance on a specific dataset. Supervised approaches, provided they have high-quality training data, tend to be easily generalizable because of their ability to predict model update cycles to reduce bias [45, 67].

In this work, an approach that combines heuristic-based and machine-learning-based (supervised) approaches has been adopted. Moreover, a human-in-the-loop approach is exploited to assess the results of the automatic annotations to both modify incorrect annotations, and provide feedback to fine-tune the algorithms.

A comprehensive STI approach must consider and adequately balance the different characteristics of a table (or a set of tables). The annotation involves several key challenges: i) *disambiguation*: the type of the entities described in a table are not known in advance, and those entities may correspond to more than one type in the KG. ii) *homonymy*: this issue is related to the presence of different entities with the same name and type. iii) *matching*: the mention in the table may be syntactically different from the label of the entity in a KG (i.e. use of acronyms, aliases, and typos). iv) *NIL-mentions*: the approach much also consider strings that refer to entities for which a representation has not yet been created within the KG, namely NIL-mentions. v) *literal and named-entity*: in a table, there can be columns that contain named-entity mentions (NE-column) and columns containing strings (L-column). vi) *missing context*: it is often easier to extract the context from textual documents than from tables due to the amount of content to be processed. Moreover, the header, the first row of a table, which usually contains descriptive attributes for the columns, may or may not be present. So, the relationships (properties) between the columns are not known in advance. vii) *amount of data*: the approach must consider large tables with many rows and columns, and tables with very few mentions. viii) *different domains*: the tables can belong to generic of specific domains.

In STI, there are essentially three tasks: Column Type Annotation (CTA), Column Property Annotation (CPA), and Cell Entity Annotation (CEA), where the last one represents the core task. This task is referred to as Entity Linking (EL), which comes in different flavors depending on the considered data formats but shares some common features. Due to the extensive search space of entities, a majority of EL methodologies employ a two-step process. The first step involves Candidate Retrieval (CR) [58], where potential entities corresponding to the input string are gathered. The second step focuses on Entity Disambiguation (ED) [57], wherein one or none of the candidate entities are ultimately selected to represent the input string. In general, CR yields a ranked list of candidate entities, which is subsequently re-ranked during the ED phase.

The retrieval of a set of candidate entities is typically achieved through Information Retrieval (IR) techniques. However, recent advancements have introduced dense approaches [76, 66] to CR. These dense methods leverage dense vector representations to perform both retrieval and ranking in a single step. This not only simplifies the process but also potentially enhances the accuracy and efficiency of the system since the role of features is pivotal, as they serve as the key elements that the model uses for both retrieval and ranking. Yet, these methods show some limitations, since the representation of features in multi-dimensional spaces remains a critical factor, and feature extraction often depends on reference knowledge graphs. Additionally, the performance of these approaches is closely tied to the quality and quantity of the training data.

Classic IR approaches based on search engines still provide valuable solutions to support entity search, mainly because they do not require training, can work with any KG, and easily adapt to changes in the reference KG. Although IR-based entity search has been used extensively, especially in table-to-kg matching, their use has been frequently left to custom optimisations and not adequately discussed or documented in scientific papers [51, 44, 8, 63]. As a result, such systems must be developed from scratch to address new problems, including data indexing techniques, query formulation and service set-up.

Listing 1.1: Prompt used to request table annotation using Wikidata entities.

```
Title, Director, Release date, Distributor, Length in min, Worldwide gross
X-Men, Bryan Singer, 14/07/2000, Twentieth Century Fox, 104, 296,300,000
Batman Begins, Christopher Nolan, 17/06/2005, Warner Bros., 140, 371,853,783
Superman Returns, Bryan Singer, 28/06/2006, Warner Bros., 154, 391,081,192
Avatar, James Cameron, 15/01/2009, Twentieth Century Fox, 162, 2,744,336,793
Jurassic World, Colin Trevorrow, 11/06/2015, Universal Pictures, 124, 1,670,400,637

Can you perform Entity Linking over the table above using Wikidata as
reference Knowledge Graph?
```

ED is central to an EL process, and various approaches exist that exploit different types of format and features of the input [67]. However, to be successful, the correct candidate must be included in the set of candidates returned by CR, but this is still a challenge. Moreover, although it is theoretically possible to examine the list of candidates to an arbitrary depth, practical considerations of computational efficiency necessitate to consider a limited number of candidates.

More recently, generative AI has introduced novel solutions to any search problems. Today, many platforms allow the generation of texts, videos, and images based on textual input provided by a user. The hype raised in the last months of 2022, thanks to some OpenAI products, Dall-E and ChatGPT above all, is due to the curiosity effect triggered precisely by the ease of access and use of these systems. ChatGPT can produce an almost perfect text from a reasonably detailed request (the prompt). The user experience and the quality of the produced output have allowed these systems to spread widely, even adopted by large organizations [69].

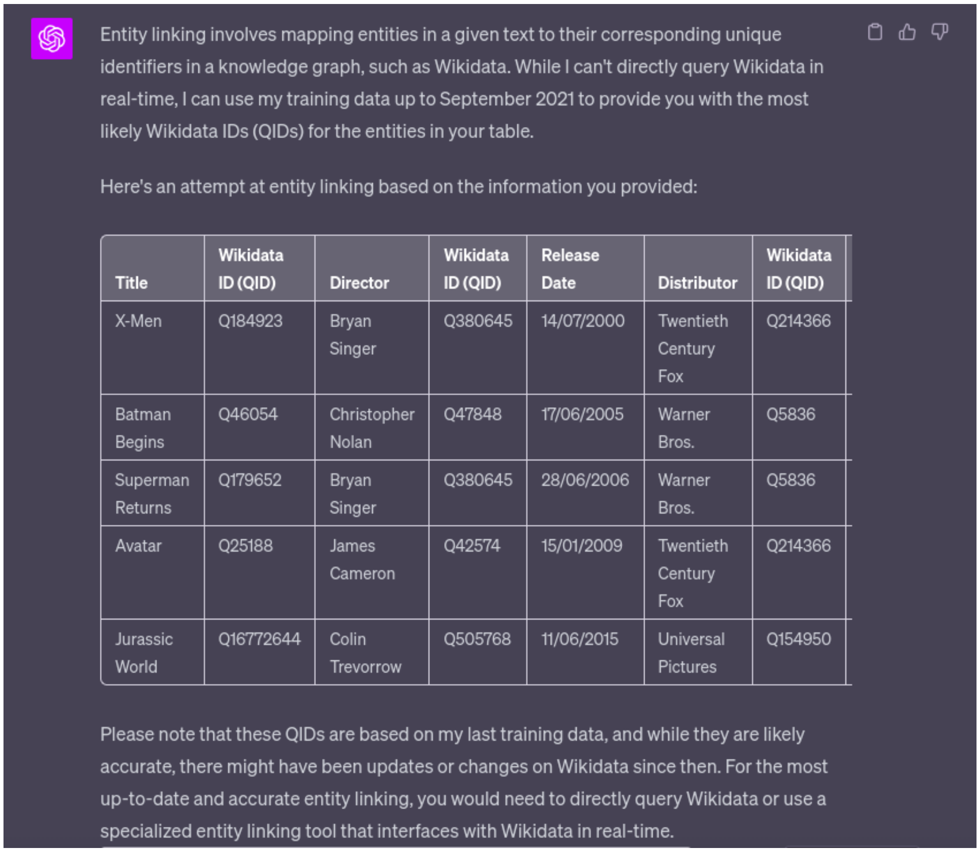
OpenAI released version 4 of GPT with the claim: 'while [GPT-4 is] less capable than humans in many real-world scenarios, exhibits human-level performance on various professional and academic benchmarks. It passes a simulated bar exam with a score around the top 10% of test takers[2]'. The GPT scores stand out in the top ranks for at least 34 tests of ability in fields as diverse as macroeconomics, writing, math, and vinology [54]. The news that GPT-4 beats 90% of the lawyers trying to pass the bar caused a stir [3]. If Large Language Model (LLM) are used in disparate contexts, experiments relating to STI demonstrate that these models are not always successful to annotate tabular data. For instance, considering the Table 1.1, providing the prompt in the Listing 1.1, the result proposed by ChatGPT 4 is shown in Figure 1.2.

The evidence is that most of the proposed Wikidata IDs are incorrect: `Q184923` refers to the entity 'Zeilhofen', a human settlement in Germany; `Q25188` to the movie 'Inception' (in this case, at least the context is correct since it refer to a movie); `Q16772644` is the Wikimedia category '1983 in diving'. The only one which is correct is `Q42574` that refers to the director 'James Cameron' while `Q46054` does not even exist. What is described is an example of using a off-the-shelf LLM. The results can improve using an adequately trained LLM. However, obtaining the levels of precision and recall of current STI approaches with these models is currently hard to generalize.

In contrast with the previous observations, it is noteworthy that substantial improvements

[2]https://openai.com/research/gpt-4

[3]https://www.forbes.com/sites/johnkoetsier/2023/03/14/gpt-4-beats-90-of-lawyers-trying-to-pass-the-bar/



Entity linking involves mapping entities in a given text to their corresponding unique identifiers in a knowledge graph, such as Wikidata. While I can't directly query Wikidata in real-time, I can use my training data up to September 2021 to provide you with the most likely Wikidata IDs (QIDs) for the entities in your table.

Here's an attempt at entity linking based on the information you provided:

Title	Wikidata ID (QID)	Director	Wikidata ID (QID)	Release Date	Distributor	Wikidata ID (QID)
X-Men	Q184923	Bryan Singer	Q380645	14/07/2000	Twentieth Century Fox	Q214366
Batman Begins	Q46054	Christopher Nolan	Q47848	17/06/2005	Warner Bros.	Q5836
Superman Returns	Q179652	Bryan Singer	Q380645	28/06/2006	Warner Bros.	Q5836
Avatar	Q25188	James Cameron	Q42574	15/01/2009	Twentieth Century Fox	Q214366
Jurassic World	Q16772644	Colin Trevorrow	Q505768	11/06/2015	Universal Pictures	Q154950

Please note that these QIDs are based on my last training data, and while they are likely accurate, there might have been updates or changes on Wikidata since then. For the most up-to-date and accurate entity linking, you would need to directly query Wikidata or use a specialized entity linking tool that interfaces with Wikidata in real-time.

Figure 1.2: ChatGPT 4 output to request table annotation using Wikidata.

are observed when utilizing Wikipedia as the reference Knowledge Graph instead of Wikidata with ChatGPT-4. Listing 1.2 provides details of the employed prompt, and Figure 1.3 illustrates the resulting output. Notably, in this instance, the request explicitly included a confidence score for each annotation. Remarkably, the results achieved a perfect one hundred percent accuracy.

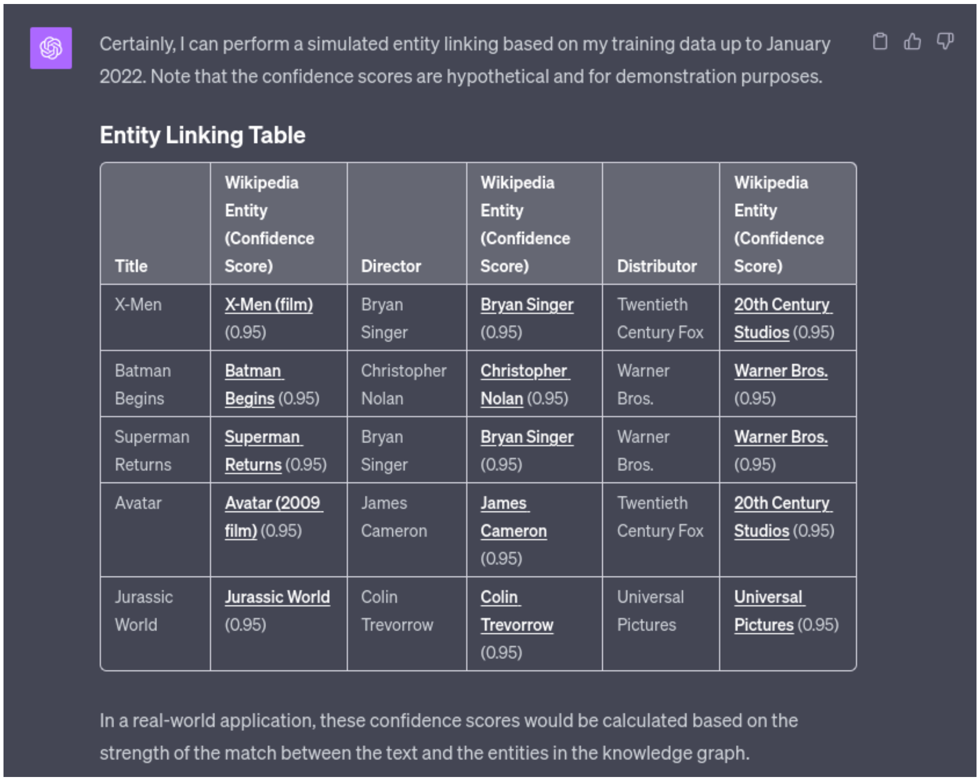
To summarize and conclude the discussion, ChatGPT-4 has shown considerable efficacy in addressing the problem of entity linking from a human-understandability standpoint by providing Wikipedia links. However, it currently falls short in offering a machine-processable solution, as it does not provide Wikidata IDs. This limitation highlights the need for further advancements to bridge the gap between human-readable and machine-processable entity linking.

In this introductory chapter, the stage has been set for the exploration of the field encom-

Listing 1.2: Prompt used to request table annotation using Wikipedia entities.

```
Title, Director, Release date, Distributor, Length in min, Worldwide gross
X-Men, Bryan Singer, 14/07/2000, Twentieth Century Fox, 104, 296,300,000
Batman Begins, Christopher Nolan, 17/06/2005, Warner Bros., 140, 371,853,783
Superman Returns, Bryan Singer, 28/06/2006, Warner Bros., 154, 391,081,192
Avatar, James Cameron, 15/01/2009, Twentieth Century Fox, 162, 2,744,336,793
Jurassic World, Colin Trevorrow, 11/06/2015, Universal Pictures, 124, 1,670,400,637

Can you perform Entity Linking over the table above using Wikipedia
as reference Knowledge Graph, providing a confidence score for each entity as well?
```



Certainly, I can perform a simulated entity linking based on my training data up to January 2022. Note that the confidence scores are hypothetical and for demonstration purposes.

Entity Linking Table

Title	Wikipedia Entity (Confidence Score)	Director	Wikipedia Entity (Confidence Score)	Distributor	Wikipedia Entity (Confidence Score)
X-Men	X-Men (film) (0.95)	Bryan Singer	Bryan Singer (0.95)	Twentieth Century Fox	20th Century Studios (0.95)
Batman Begins	Batman Begins (0.95)	Christopher Nolan	Christopher Nolan (0.95)	Warner Bros.	Warner Bros. (0.95)
Superman Returns	Superman Returns (0.95)	Bryan Singer	Bryan Singer (0.95)	Warner Bros.	Warner Bros. (0.95)
Avatar	Avatar (2009 film) (0.95)	James Cameron	James Cameron (0.95)	Twentieth Century Fox	20th Century Studios (0.95)
Jurassic World	Jurassic World (0.95)	Colin Trevorrow	Colin Trevorrow (0.95)	Universal Pictures	Universal Pictures (0.95)

In a real-world application, these confidence scores would be calculated based on the strength of the match between the text and the entities in the knowledge graph.

Figure 1.3: ChatGPT 4 output to request table annotation using Wikipedia.

passing Semantic Table Interpretation and Entity Linking. The significance of structured data, particularly tables, within the contemporary data-driven landscape has been emphasized. The proliferation of tables across diverse domains has prompted substantial interest in the challenge of associating table mentions with entities present in Knowledge Graphs.

The introduction of the SemTab challenge, a recurring benchmark, has catalyzed research and innovation in the field of STI.

Furthermore, the limitations of heuristic-based approaches and the need for techniques characterized by enhanced robustness and generalization have been discussed. Supervised approaches show promise in this context, depending on the availability of high-quality training data.

The evolving landscape of Large Language Models (LLMs) and their potential application in STI and EL have also been touched upon. While LLMs like ChatGPT 4 offer impressive capabilities, attaining precision and recall levels commensurate with state-of-the-art STI approaches presents challenges.

Lastly, the impact of employing distinct Knowledge Graphs, such as Wikidata and Wikipedia, in the context of EL using ChatGPT 4 has been demonstrated. This underscores the subtleties and potential enhancements in performance.

With this foundational groundwork established, the subsequent chapters of this work will delve into the intricacies of STI, EL, and the experimental outcomes, illuminating the strengths and weaknesses inherent in various approaches. Comparative analyzes will be conducted, and new perspectives within this rapidly evolving field will be explored.

1.1 Contributions

This thesis contributes to the advancement of Information Retrieval Systems in the context of KG and STI, with a particular focus on the integration of Machine Learning (ML) techniques. The work enriches existing methodologies in *KG construction*, *KG extension*, and *data enrichment*, thereby extending the current state of the art.

The thesis contends that STI provides a crucial framework for facilitating user-assisted annotation, particularly in expansive data environments. The specific contributions of this research are as follows:

1. Proposal and development of a generalized Information Retrieval System for datasets, emphasizing KG:

    - Introduction and definition of an innovative approach for indexing and retrieving data across multiple datasets, with particular attention to the nuances of Knowledge Graphs (KGs);
    - Empirical validation of the proposed indexing approach through systematic testing on benchmark datasets to demonstrate efficacy, efficiency and robustness.

2. Formulation and implementation of algorithms for annotation tasks in the context of STI:

    - Design and definition of a novel EL algorithm that exploits ML methodologies, aimed at improving existing approaches by generating human-interpretable confidence scores;
    - Design and definition of additional heuristic-based algorithms for Column Type Annotation (CTA) and Column Property Annotation (CPA);
    - Adoption of a data representation format for annotations to promote interoperability among different outputs of STI annotation tools, namely adopting W3C standards;
    - Foresee a human-in-the-loop approach to enable review cycles to refine the results produced by the algorithms, and drive the refinement of algorithms toward domain specialization;
    - Empirical validation of the proposed algorithms, substantiated through experimentation on benchmark datasets.

3. Implementation of the algorithms at scale

    - Development of a scalable approach that adapts the existing STI framework to handle enrichment pipelines in large-scale data scenarios, thereby enhancing its operational efficiency and applicability.

1.2 Thesis Structure

This thesis is organized to provide a coherent and logical exposition of the research undertaken. The document starts with an introductory chapter that sets the stage for the research questions and objectives. This is followed by a review of the relevant literature, providing the theoretical background necessary for the study. Subsequent chapters are dedicated to the presentation of the research methodology, empirical findings, and analyses. The thesis concludes with a summary chapter that synthesizes the key contributions and oulines future research activities.

Chapter 2: Preliminaries. This chapter serves as a foundational guide to the key concepts and terminologies that are integral to the thesis. It introduces the reader to the essential definitions and frameworks in the domains of Knowledge Graphs (KGs), Information Retrieval, Tables, Semantic Table Interpretation (STI), Neural Networks, and Data Pipelines. Formal definitions and structures are provided to offer a rigorous understanding of these areas. The chapter also includes illustrative figures and examples to aid in conceptual clarity. This preparatory chapter aims to equip the reader with the necessary background to engage deeply with the subsequent chapters.

Chapter 3: Related Work. This chapter offers a comprehensive review of the state-of-the-art methodologies and technologies that are relevant to the scope of this research. The chapter commences with an introduction to Entity Retrieval, highlighting both conventional and contemporary strategies. Subsequently, the focus shifts to Entity Linking, particularly as it pertains to Semantic Table Interpretation. Here, both heuristic-based and machine learning-based methods are summarized. The chapter also addresses the specific challenges and existing solutions concerning NIL entities. Additionally, the role of Human-in-the-Loop (HITL) techniques in enhancing model performance is explored. The chapter concludes with an overview of current best practices in the construction and management of data pipelines. Each section culminates with a summary that outlines the advantages and limitations of the existing work in the respective area.

Chapter 4: Information Retrieval Systems. In this chapter, an approach is delineated for data retrieval across diverse datasets, with a particular emphasis on Knowledge Graphs (KG). The chapter concentrates on the nuances of entity retrieval, as well as the extraction of other data types such as types associated with entities and relationships (properties) between them.
The research questions addressed in this chapter are:

Q4.1: What are the most effective methods for indexing a Knowledge Graph to optimize entity retrieval?

Q4.2: What strategies can be employed to ensure that the correct candidate entity is consistently included in the retrieval result set?

Q4.3: What validation methods can be applied to assess the reliability and effectiveness of the proposed information retrieval system for Knowledge Graphs?

Chapter 5: Semantic Enrichment of Tabular Data. In this chapter, a methodology is outlined for performing entity linking in tabular data using machine learning techniques. The chapter discusses how the problem has been addressed and concludes with the validation of the methodology.
The research questions addressed in this chapter are:

Q5.1: How can machine learning techniques be effectively applied to the problem of entity linking in tabular data?

Q5.2: What methodologies can be employed to optimize the ranking algorithm, such that the correct candidate entity is most frequently positioned first?

Q5.3: What validation strategies can be implemented to ensure the robustness of the proposed entity linking technique?

Chapter 6: Human In The Loop (HITL) over tabular data. This chapter discusses the role of humans to review and assess uncertain annotations. In particular, the use of user feedback to learn and improve the performance of the algorithms for future use in table of the same domain.
The research questions addressed in this chapter are:

Q6.1: Can the human effort be limited to ensure a feasible approach to uncertainty revision?

Q6.2: How can a feedback-driven life-cycle for annotations be designed to improve the algorithms in domain-specific applications?

Chapter 7: Entity Linking in Large-Scale Data Environments. This chapter outlines a methodology for implementing entity linking on a large scale, focusing on the challenges and solutions associated with handling vast amounts of data. The chapter concludes with an empirical evaluation of the methodology's time efficiency.
The research questions addressed in this chapter are:

Q7.1: How can entity linking be effectively scaled to handle large volumes of tabular data?

Q7.2: What validation metrics can be employed to assess the time efficiency gains of the proposed large-scale entity linking methodology?

Chapter 8: Conclusions and Future Directions. This final chapter synthesizes the key findings of the research, discussing their implications and contributions to the field. It also outlines potential avenues for future research and provides a summary of the study's limitations. The chapter serves to encapsulate the research journey, offering a comprehensive view of its significance and impact.

1.3 Reproducibility

Reproducibility is a fundamental aspect of scientific research, ensuring the validity and reliability of findings [65]. The methodological approach used in this study is accessible for replication and verification. The list of repositories, which contain the implementation details of the approaches proposed in this thesis, along with usage instructions, is provided as follows:

- LamAPI

  [code]: https://bitbucket.org/disco_unimib/lamapi

- Alligator (api version)

  [code]: https://github.com/roby-avo/alligator

- Alligator (scalable version)

  [code]: https://github.com/roby-avo/alligator-scalable

2. Preliminaries

In this chapter, we introduce fundamental definitions to enhance the reader's understanding of the topic. We will cover the concepts of knowledge graph (KG), table, semantic table interpretation (STI), neural network, and data pipeline.

2.1 Knowledge Graph

A knowledge graph (KG) is a structured representation of knowledge that organizes information about entities, their attributes, and relationships between them. It serves as a valuable resource for capturing and representing real-world knowledge in a machine-readable format. In a knowledge graph, entities are represented as nodes, while the relationships between entities are represented as edges connecting the nodes. Each entity and relationship typically have additional attributes or properties associated with them, providing further context and descriptive information. Knowledge graphs leverage semantic connections and ontological reasoning to enable powerful information retrieval, inference, and knowledge discovery. By modeling knowledge in a graph-like structure, knowledge graphs facilitate efficient data integration, knowledge representation, and reasoning capabilities, enabling more effective analysis and interpretation of complex information domains.

Within the context of a knowledge graph, an ontology refers to a formal representation of domain knowledge, providing a conceptual framework for organizing and structuring knowledge. An ontology defines a vocabulary of concepts and relationships that capture the entities and their connections within the knowledge graph. It specifies the types of entities and relationships that can exist in the graph, along with any constraints or rules governing their usage. By incorporating an ontology into a knowledge graph, it enhances the semantic richness and enables more precise querying, reasoning, and inferencing capabilities. Ontologies facilitate knowledge representation, semantic connections, and logical deductions, serving as a foundational component for knowledge-based systems and contributing to the overall structure and meaning within a knowledge graph.

Let's to introduce the formal definitions of Knowledge Graph and Ontology.

Definition 1 (Knowledge Graph) A knowledge graph, denoted as KG, is a directed graph $G = (V, E)$, where V represents a set of entities and E represents a set of directed edges. Each entity $v \in V$ corresponds to a unique identifier and may have associated attributes or properties. Each directed edge $(v_i, r, v_j) \in E$ represents a relationship r between entities v_i and v_j. Additionally, each edge may have associated attributes or properties. The knowledge graph KG serves as a structured representation of real-world knowledge, enabling efficient data integration, reasoning, and knowledge discovery.

Definition 2 (Ontology) An ontology, denoted as O, is a formal representation of domain knowledge, providing a conceptual framework for organizing and structuring knowledge. An ontology consists of a set of concepts, relationships, and axioms that define the vocabulary and rules governing the domain. Formally, an ontology O is defined as a tuple $O = (C, R, A)$, where C represents a set of concepts, R represents a set of relationships, and A represents a set of axioms. Each concept $c \in C$ represents an entity or a class of entities in the domain. Each relationship $r \in R$ represents a binary or n-ary association between concepts. Axioms in A define logical rules or constraints governing the ontology. Ontologies facilitate knowledge representation, semantic connections, and reasoning capabilities, serving as a foundation for knowledge-based systems and knowledge graphs.

RDF (Resource Description Framework) is the most popular format to represents information in the form of triples, which consist of $< subject, predicate, object >$ statements. For example, $< Da\ Vinci, painted, Mona\ Lisa >$. RDF data forms a graph structure, where resources are nodes, and relationships between them are edges. This graph model allows for the representation of complex and interconnected data. A example of KG is illustrated in 2.1.

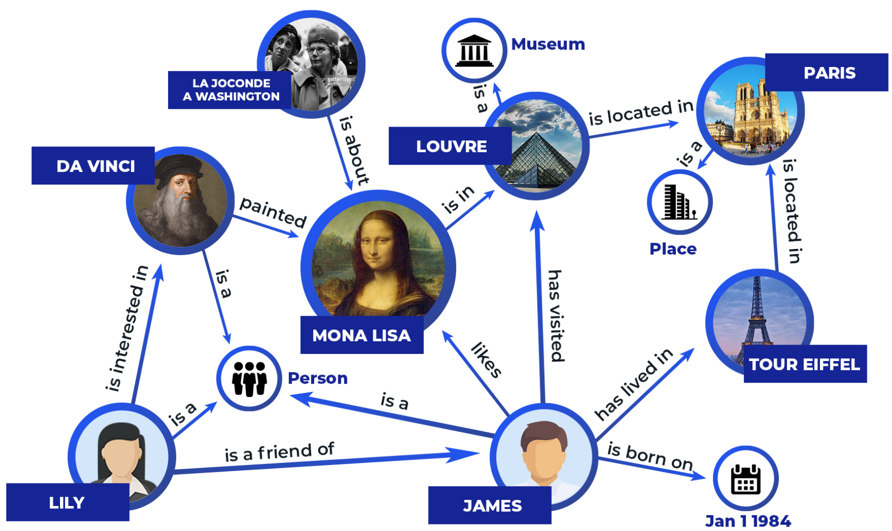


Figure 2.1: An example of KG

As of now, there is a rich landscape of knowledge graphs that exist, encompassing a diverse range of domains and applications. These knowledge graphs serve as powerful resources for organizing and representing structured information, enabling efficient data integration, semantic connections, and knowledge discovery. Some notable examples include the Google Knowledge Graph, DBpedia, Wikidata, YAGO and Freebase. Each of these knowledge graphs offers unique characteristics and strengths, providing valuable insights into different aspects of the world's knowledge. Together, these knowledge graphs contribute to a growing interconnected web of information, fueling advancements in various fields such as NLP, IR and AI. The continuous evolution and expansion of these knowledge graphs demonstrate the ongoing efforts and collaborations in building comprehensive and accessible repositories of knowledge. In Figure 2.2, a subset of the Wikidata Knowledge Graph (KG) is presented, which is derived from the annotation process applied to Table 1.1.

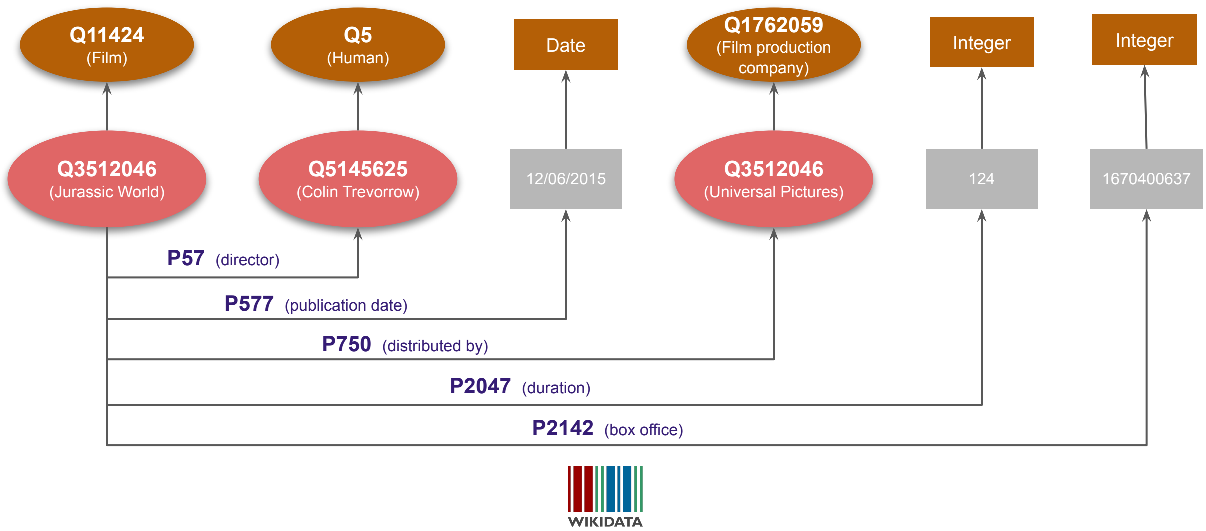


Figure 2.2: A subset of wikidata KG

2.2 Information Retrieval

Information retrieval is a fundamental aspect of knowledge management and plays a crucial role in extracting relevant and meaningful information from wide and diverse data sources. It encompasses the techniques and methodologies used to retrieve information that satisfies a user's information needs. The process involves matching user queries with indexed data or documents and ranking the results based on their relevance. Figure 2.3 illustrates the typical components of an information retrieval system. These include the user query, the indexing process, the retrieval model, and the ranking algorithm. The indexing process involves analyzing and organizing the data or documents into a searchable form, such as an inverted index. The retrieval model defines the mathematical framework used to calculate the relevance between the user query and the indexed data. The ranking algorithm determines the order in which the results are presented to the user. Information retrieval techniques have been widely applied in various domains, including web search engines, document retrieval systems, and recommendation systems.

2.3 Tables

A table is a mathematical structure consisting of a set of rows and a set of columns. It is denoted as $T = (R, C, D)$, where R represents the set of rows, C represents the set of columns, and D represents the set of data values contained within the cells of the table. Each row in the table is a sequence of values from a predetermined domain, and each column represents a distinct attribute or field. The table structure is such that each cell in the table uniquely corresponds to an intersection of a row and a column, containing a single data value. Tables provide a systematic and organized way to represent and store data, facilitating efficient data retrieval, manipulation,

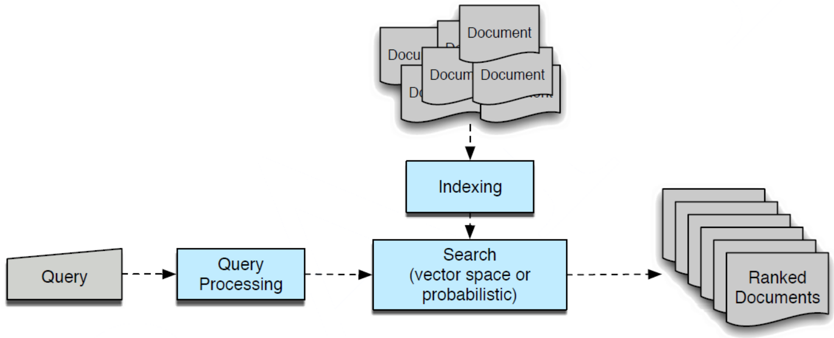


Figure 2.3: Components of an information retrieval system

and analysis. They serve as a fundamental data structure in various fields, including databases, spreadsheets, and data analysis, enabling effective data organization and communication.

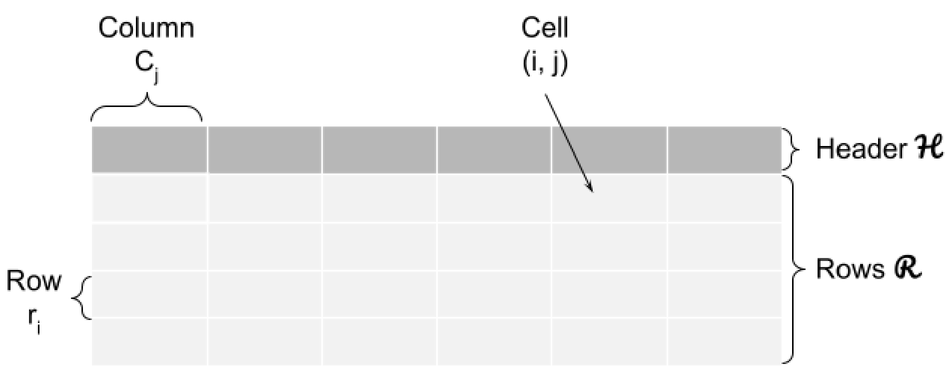


Figure 2.4: Skeleton of a table

Tables come in various forms and structures, each tailored to represent data in distinct ways. This section will discuss and showcase a few examples of these different table types.

2.3.1 Simple Table

A simple table typically consists of rows and columns that represent data in a grid format. Each cell in the grid contains a piece of information, as demonstrated in Table 2.1.

Table 2.1: Example of a Simple Table

Header1	**Header2**
Data1	Data2

2.3.2 Vertical Table

Vertical tables, often called transposed tables, have their headers on the left, running vertically. Each row represents a different attribute or field, as illustrated in Table 2.2.

Table 2.2: Example of a Vertical Table

Header1	Data1
Header2	Data2

2.3.3 Nested Table

Nested tables involve having a table inside another table. This structure is used when one wants to represent hierarchical or multi-part data within a single cell of the outer table, as presented in Table 2.3.

Table 2.3: Example of a Nested Table

<table>
<tr><th>Header1</th><th colspan="2">Header2</th></tr>
<tr><td rowspan="2">Data1</td><td>Nested1</td><td>Nested2</td></tr>
<tr><td>Nested3</td><td>Nested4</td></tr>
</table>

2.3.4 Relational Tables

Relational tables represent the foundation of relational databases. They consist of multiple tables interconnected through keys, ensuring data consistency and integrity. Tables 2.4 and 2.5 exemplify how the *Users* table and the *Orders* table are related through the *UserID* key.

Table 2.4: Users Table

UserID	**UserName**
1	Alice
2	Bob

Table 2.5: Orders Table

OrderID	**UserID**	**Product**
001	1	Book
002	2	Pen
003	1	Notebook

While all these table types have their uses and significance in various scenarios, the primary focus and interest lie in well-formed and relational tables due to their structured representation and capacity to ensure data integrity.

2.4 Semantic Table Interpretation (STI)

STI algorithms include a variety of tasks designed to address different challenges, like identifying the subject column of a table (which describes the primary entities within the table), determining the semantic types of columns, disambiguating values within cells, and recognizing relationships between values in different columns.

Before describing the different tasks, a clear distinction about the possible semantic types of a column is needed. For the proposed applications, the main distinction operated by STI is the identification of three different kinds of columns, which are S-columns, NE-columns and LIT-columns:

- *Subject column (S-column):* is the one that describes the primary entities or objects that the table is primarily about. It is often the most important column in terms of conveying the main focus or subject matter of the table. For example, in a table about films, the subject column might be 'Film Title';
- *Named-Entity column (NE-column):* column that refer to specific, named concepts or entities in the real world. These can include names of people, organizations, locations, dates, product names, and more. Recognizing and categorizing named entities is crucial for understanding the structured data, extracting meaningful information from it, and reconciling them against a KG;
- *Literal column (LIT-column):* columns that contains cells which are numerical values, date values or generic text, for instance description text. These values are often descriptive or quantitative in nature and don't correspond to named entities or objects.

As shown in Figure 2.5, the identification of types (Column Type Annotation, CTA) is just one of the possible tasks in the context of a STI application. An STI algorithm takes a table and a reference KG as inputs, generating a set of annotations that elucidate the table's semantics. These annotations establish a mapping between the source table and the KG, usually referred to as Semantic Table Annotations. These annotations can be categorized into two distinct levels:

- *Instance-level annotation:* which is mainly covered by entity linking algorithms for tables; the instance-level annotation maps the disambiguated content of the table to entities in the reference KG. The most used approaches to the instance-level annotation are: cell-based annotation, where the content of a cell is mapped to an entity (e.g., given DBpedia as the reference KG, the cells containing the labels Einstein and General Theory of Relativity can be mapped to the entities Albert Einstein and General Relativity, respectively); and row-based annotation, where a table row is mapped to an entity, usually referring to the one-entity-per-row assumption;

- *Schema-level annotation:* arises from interpreting the table schema and mapping schema elements to types and properties within the reference KG. Annotation strategies might map column headers to classes and create associations between pairs of column headers and properties. One possible strategy involves mapping the entire table to a type within the ontology and associating each column header with a property from the ontology, particularly when operating under the one-entity-per-row assumption.

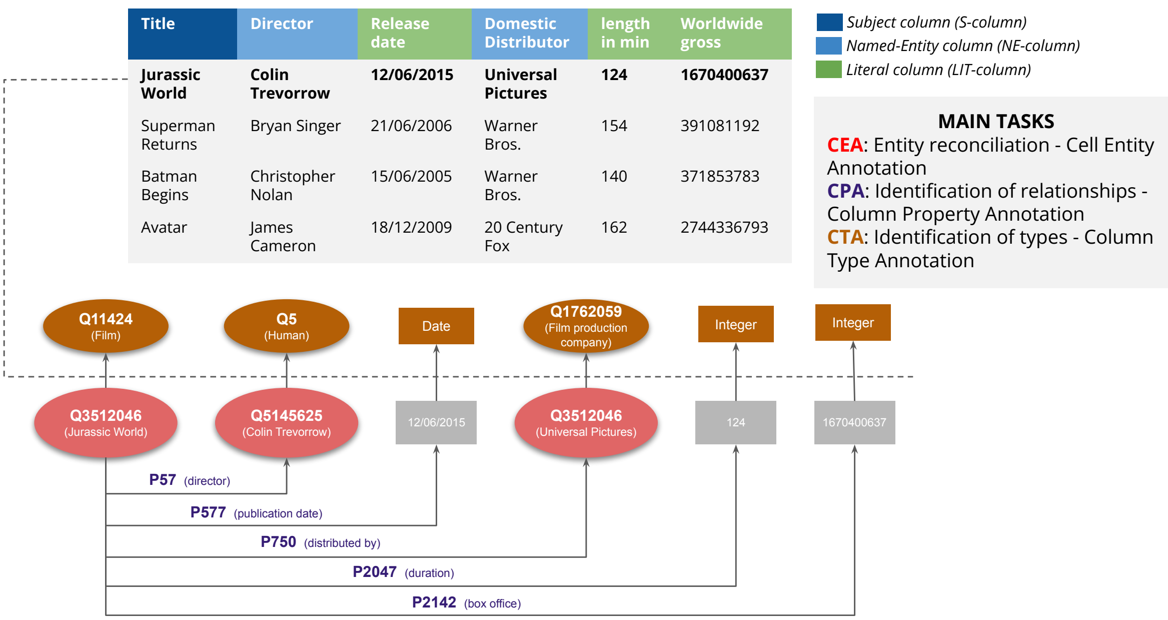


Figure 2.5: An example of annotated table

Diverse annotations can arise from a single STI process, potentially leading to distinct output based on the chosen strategy. A given STI approach might generate one or more of the aforementioned annotations. Depending on the specific application context (for example, whether it involves fully automated matching or interactive matching), the annotation tasks can be tackled collectively, enabling mutual reinforcement and interaction between them. Approaches designed to include all STI tasks often involve intricate pipelines to fine-tune the collection and propagation of evidence across these tasks. The typical procedure for such algorithms consists of three primary steps:

1. *Retrieval of Potential Candidates:* this step involves identifying possible candidates that align with KG facts;

2. *Ranking of Retrieved Entities:* the retrieved candidates are then ranked based on a scoring mechanism or similarity measure;

3. *Annotation Decision:* the final step entails deciding whether to annotate an entity or not. For instance, this decision could be made based on a threshold applied to the chosen similarity measure.

These steps collectively form the standard procedure for these types of algorithms.

2.5 Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the human brain. They are composed of interconnected processing elements, or 'neurons', which are used to estimate functions that depend on a large number of inputs.

2.5.1 Structure of Neural Networks

A typical neural network consists of three types of layers: input, hidden, and output layers.

- The *input layer* receives various forms of input data. This layer passes the data onto the hidden layers without any computation.
- The *hidden layers* perform computations and transfer information from the input nodes to the output nodes. A network can have any number of hidden layers.
- The *output layer* provides the final output. It interprets and classifies the input pattern appropriately.

Each neuron in a neural network uses a set of learnable weights and a bias to convert an input into an output. This is typically done through a non-linear function known as an activation function.

2.5.2 Training of Neural Networks

The training process of a neural network involves adjusting the weights and biases of the network based on the error of the output. This error is calculated using a loss function, which measures the difference between the predicted output and the actual output.

The most common training algorithm for neural networks is the backpropagation algorithm. This involves propagating the error backwards through the network, starting from the output layer and moving through each layer in reverse. At each layer, the weights and biases are updated in a way that minimizes the loss function.

2.5.3 Applications of Neural Networks

Neural networks have a wide range of applications, including image recognition, speech recognition, natural language processing, and many others. They are also fundamental to deep learning, a subfield of machine learning, where neural networks are expanded into neural architectures that are several layers deep.

This is a very broad overview. Depending on your focus, you may want to add subsections about specific types of neural networks (like Convolutional Neural Networks for image tasks, Recurrent Neural Networks for sequence tasks, etc.), or about specific aspects of training (like gradient descent, different types of loss functions, etc.).

2.5.4 Diverse Types of Neural Networks: An Overview

Neural networks come in various forms, each designed to address specific types of problems. One of the simplest forms is the Feedforward Neural Network (FNN), where information flows unidirectionally from the input layer to the output layer without any loops. Convolutional Neural Networks (CNNs) are predominantly used in image processing tasks, as they can identify spatial hierarchies or patterns in the input data. Recurrent Neural Networks (RNNs) and their advanced version, Long Short-Term Memory networks (LSTMs), excel at processing sequential data, making them suitable for tasks such as language modeling or time series analysis. Autoencoders are specialized neural networks used for data compression, noise reduction, and feature extraction. Generative Adversarial Networks (GANs) consist of two networks that compete with each other to generate new data resembling the input data. This is just the tip of the iceberg; numerous other specialized architectures continue to emerge in the ever-evolving field of neural networks.

2.5.5 Delving into Feedforward Neural Networks

Feedforward Neural Networks, also referred to as 'Multi-layer Perceptrons (MLP)', are the simplest type of artificial neural network. In a feedforward network, information moves in only one direction, forward, from the input layer, through the hidden layers, and to the output layer. There are no loops in the network; information is always fed forward, never fed back. Each neuron in a layer is connected to all neurons located in the previous layer and the following layer. Neurons in the same layer have no connections with each other. This type of network is widely used in pattern recognition and classification tasks where the input data is stationary or does not change with time, such as handwriting recognition or image recognition. Figure 2.6 provides an illustrative example of a feedforward neural network:

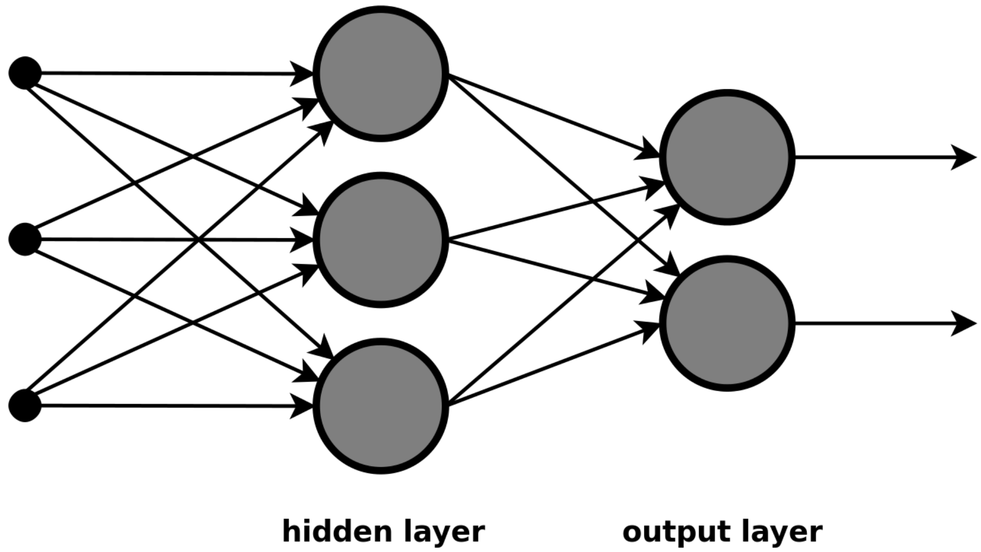


Figure 2.6: A typical structure of a Feedforward Neural Network

2.6 Data Pipelines

Data pipelines are a sequence of data processing steps, where data is extracted from sources, transformed into a format suitable for analysis, and loaded into a final storage destination.

2.6.1 Components of a Data Pipeline

The main components of a data pipeline are:

- **Data Extraction**: This involves gathering data from various sources such as databases, servers, text files, APIs, or other data repositories.
- **Data Transformation**: The extracted data is transformed into a suitable format for analysis. This could involve data cleaning, normalization, aggregation, and data integration.
- **Data Loading**: This is the process of loading the transformed data into a final destination like a data warehouse, database, or a data lake.

2.6.2 Types of Data Pipelines

Data pipelines can be categorized into two types:

- **Batch Processing**: Data is collected over time and processed all at once. This type of processing is suitable for applications where real-time data processing is not required.
- **Real-time Processing**: Here, data is continuously processed and made available for analysis almost instantly after being collected. This is suitable for applications where immediate insight from data is crucial.

2.6.3 Importance of Data Pipelines

Data pipelines are an essential part of modern data architecture. They automate the data flow process, making it faster and reducing the risk of errors. By ensuring that high-quality data is always available for analysis, data pipelines enable organizations to make data-driven decisions, forecast trends, and improve operational efficiency.

2.7 Data Pipeline Example

In the context of entity linking over tabular data, a typical data pipeline may include the following steps:

1. **Data Extraction**: The initial step in any data pipeline involves extracting data from its source. For tabular data, this source could be a relational database, a CSV file, an Excel spreadsheet, or a Google Sheets document. During extraction, the raw data, comprising both entities and the text where they're mentioned, is collected.

2. **Data Cleaning**: Extracted data might not always be in a suitable format for processing. As such, it becomes necessary to perform cleaning operations, such as converting text to lowercase, removing punctuation or other special characters, and handling missing values.

3. **Named Entity Recognition (NER)**: This process involves identifying the named entities present in the text data. These entities can be organizations, persons, locations, expressions of times, quantities, monetary values, percentages, etc.

4. **Candidate Generation**: For each identified entity, a list of candidate entities from the knowledge graph is generated. These candidates, which could potentially correspond to the mentioned entity, are identified through methods like looking for exact matches, partial matches, or synonyms.

5. **Feature Extraction**: During this step, features are extracted for each candidate entity. These features, which can aid in determining whether the candidate is a match for the mentioned entity, could include text similarity between the candidate and the mention, the candidate's popularity in the knowledge graph, or other contextual clues.

6. **Entity Linking**: This step uses the extracted features to decide which candidate entity (if any) is a match for each mentioned entity. This decision could be made using machine learning models or heuristic methods.

7. **Data Storing**: Upon completion of the entity linking, the results are stored in a suitable format for further analysis or usage. This storage could be a database, a file, or another data storage system.

This structured approach to handling data ensures a systematic flow from raw data to meaningful insights and serves as a robust foundation for any data-driven decision-making process.

3. Related work (SoTa)

3.1 Introduction

Given a KG containing a set of entities E and a collection of named-entity mentions M, the goal of EL is to map each entity mention $m \in M$ to its corresponding entity $e \in E$ in the KG. As described above, a typical EL service consists of the following modules [67]:

1. Candidate Retrieval (CR). In this module, for each entity mention $m \in M$, irrelevant entities in the KG are filtered out to return a set E_m of candidate entities: entities that mention m may refer to. To achieve this goal, state-of-the-art techniques have been used, such as name dictionary-based techniques, surface form expansion from the local document, and methods based on search engines.

2. Entity Disambiguation (ED). In this module, the entities in the set E_m are more accurately ranked to select the correct entity among the candidate ones. In practice, this is a re-ranking activity that considers other information (*e.g.*, contextual information) besides the simple textual mention m used in the CR module.

According to the experiments conducted in [25, 14] and LamAPI [4], the role of the CR module is critical since it should ensure the presence of the correct entity in the returned set to let the ED module to find it. Hence, the main contribution of this work is to discuss retrieval configurations, *i.e.*, query and filtering strategies, for retrieving entities.

Name dictionary-based techniques are the main approaches to CR; such techniques leverage different combinations of features (*e.g.*, labels, alias, Wikipedia hyperlinks) to build an offline dictionary D of links between string names and mapping entities to be used to generate the set of candidate entities. The most straightforward approach considers exact matching between the textual mention m and string names inside D. Partial matching (*e.g.*, fuzzy and/or n-grams search) can also be considered.

Besides pure string matching, type constraints (using types/classes of the KG) associated with string mentions can be exploited to filter candidate entities. In such a case, the dictionary needs to be augmented with types associated with linked entities to enable hard or soft filtering. Listing 3.1 and 3.2 report an example of how type constraints can influence the result of the candidate entity retrieval for 'manchester' textual mention. The former shows the result without constraint: cities like Manchester situated in England or Parish in Jamaica are reported (note the similarity score equal to 1.00). The latter shows the result when type constraints are applied: types like 'SoccerClub' and 'SportsClub' allows for the promotion of soccer clubs such as 'Manchester United F.C.', which is now ranked first (similarity score 0.83).

Similar approaches have been proposed in this domain, such as the MTab [51] entity search, where keyword search, fuzzy search and aggregation search are provided. Another relevant approach is EPGEL [44], where the candidate entity generation uses both a keyword and a fuzzy search. This approach also uses BERT [21] to create a profile for each entity to improve the search results. The LinkingPark [8] method proposes a weighted combination of keywords, tri-grams and fuzzy search to maximise recall during the candidate generation process. In addition, this approach involves verifying the presence of typos before generating candidates. Concerning the other work, LamAPI provides a n-grams search and the possibility to include type constraints in the candidate search to apply type/concept filtering in the CR. Furthermore, LamAPI provides several services to help researchers in tasks like EL.

Listing 3.1: DBpedia lookup without type constraints.

```
{
"id": Manchester
"label": Manchester
"type": City Settlement ...
"ed_score": 1
},
{
"id": Manchester_Parish
"label": Manchester
"type": Settlement PopulatedPlace
"ed_score": 1
}
```

Listing 3.2: DBpedia lookup with type constraints.

```
{
"id": Manchester_United_F.C.
"label": Manchester U
"type": SoccerClub SportsClub ...
"ed_score": 0.833
},
{
"id": Manchester_City_F.C.
"label": Manchester C
"type": SoccerClub SportsClub ...
"ed_score": 0.833
}
```

3.2 ER: Entity Retrieval

When it comes to performing data retrieval, several options are available as existing solutions. These include:

- **DBpedia SPARQL endpoint**[1]: Allows performing SPARQL queries against the DBpedia KG using full-text search capabilities through the 'bif:contains' operator;
- **Wikidata SPARQL endpoint**[2]: Allows performing SPARQL queries against the Wikidata KG using exact match of the string;
- **DBpedia Lookup**[3]: Full-text search service to perform lookup over DBpedia KG;
- **Wikidata Lookup by Name**[4]: Full-text search service to perform lookup over Wikidata KG;
- **Wikidata Reconciliation service**[5]: Full-text search service to perform lookup over Wikidata KG made by W3C entity reconciliation group.

Regarding the existing solutions introduced above, it is important to consider some considerations related to SPARQL endpoints such as DBpedia and Wikidata. These endpoints are

[1] https://dbpedia.org/sparql
[2] https://query.wikidata.org/
[3] https://dbpedia.org/sparql
[4] https://www.schemaapp.com/wikidata-lookup-by-name/
[5] https://wikidata.reconci.link/en/api

primarily designed as triple storage systems and may not have robust full-text search capabilities. In particular, Wikidata, being a large knowledge graph, does not provide operators like ‘bif:contains’ for performing full-text search.

The ‘bif:contains’ operator in SPARQL performs a basic full-text search by matching tokens using OR or AND operations between them. However, it does not provide a specific ranking as output. This means that the search results are not effectively ranked based on relevance or other ranking algorithms, making it less effective compared to full-text search mechanisms that utilize advanced ranking functions like BM25.

Therefore, while SPARQL endpoints like DBpedia and Wikidata can be useful for certain types of queries and retrieval tasks, they may not be the most efficient or effective solutions when it comes to comprehensive full-text search capabilities and ranking of search results.

The following example showcases SPARQL queries for DBpedia and Wikidata, specifically targeting the mention ‘Albert Einstein’. Listings 3.3 and 3.4 provide a demonstration of these queries.

It is essential to note that the ‘bif:contains’ operator, as mentioned before, is not considered highly effective for comprehensive full-text search. Its functionality is limited and may not yield desired or accurate results.

In the case of the Wikidata query, the expression $<?item\ ?label\ "Albert\ Einstein"@en>$ denotes an exact match for the mention. It is important to highlight that this exact match is case-sensitive. For example, attempting to match $<?item\ ?label\ "Albert\ einstein"@en>$ with ‘einstein’ in lowercase will not produce any results.

Listing 3.3: Search example using a SPARQL query with DBpedia enpoint.

```
SELECT * WHERE {
  ?sub a dbo:Person .
  ?sub rdfs:label ?label.
  ?label bif:contains "Albert AND Einstein" .
  filter(langMatches(lang(?label), "en"))
}
LIMIT 100
```

Listing 3.4: Search example using a SPARQL query with Wikidata enpoint.

```
SELECT ?item ?itemLabel WHERE {
  ?item wdt:P31 wd:Q5.
  ?item ?label "Albert Einstein"@en .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
LIMIT 100
```

The W3C entity reconciliation specification[6] defines the format for representing entities. Each entity has a unique identifier (*id*), an official name (*name*), associated types with their unique identifiers and labels, and a description.

To leverage the full-text search capability of the lookup endpoint, requests are executed in a specific manner. For instance, when using the Wikidata Reconciliation Service, a GET request is sent to a designated URL, optionally supplemented with parameters like ’type’ and ’limit’. An example of such a request is as follows:

[6] reconciliation-api.github.io/specs/latest/

https://wikidata.reconci.link/en/api?queries={"q0":{"query":"AlbertEinstein","type":"Q5","limit":100}}

The two examples below illustrate the results obtained from the Wikidata Reconciliation service. In both listings, the objective is to reconcile the name ‘Albert Einstein.’ Listing 3.5 presents the reconciliation results without imposing any type constraint. In contrast, Listing 3.6 employs a type constraint of ‘Q5,’ which corresponds to ‘Human.’

A salient distinction is evident between the outcomes of the two listings, predominantly due to the incorporation of the type constraint. Specifically, when the type constraint ‘Q5’ (Human) is applied, as in Listing 3.6, ‘Q937’ (indicative of ‘Albert Einstein’) appears as the sole candidate, securing a score of 100. This score strongly suggests that the entity is the most probable match in the context of human entities.

Listing 3.5: Lookup without type constraint: returned data from Wikidata.

```
{
"id": Q937,
"label": Albert Einstein,
"description": German-born...,
"type": Q5,
"score": 100
},
{
"id": Q2253683,
"label": Albert Einstein,
"description": train service,
"type": Q753050 Q67454740...,
"score": 100
},
{
"id": Q13426745,
"label": Albert Einstein,
"description": 2013 studio album...,
"type": Q482994,
"score": 100
}
```

Listing 3.6: Lookup with type constraint (‘Q5’): returned data from Wikidata.

```
{
"id": Q937,
"label": Albert Einstein,
"description": German-born...,
"type": Q5,
"score": 100
},
{
"id": Q123371,
"label": Hans Albert Einstein,
"description": Swiss-American...,
"type": Q5...,
"score": 86
},
{
"id": Q468357,
"label": Lieserl (Einstein),
"description": 2013 studio album...,
"type": Q482994,
"score": 52
}
```

Entity retrieval is a critical task in the field of information retrieval, which aims to identify a set of candidate entities from a knowledge graph (KG) based on a given mention or query. The task of entity retrieval is particularly challenging in large-scale knowledge graphs such as Wikidata, where the number of entities is enormous and the structure of the graph is complex.

When it comes to retrieval from (semi-)structured data, including knowledge graphs (KGs), two commonly used approaches are Information Retrieval (IR) and dense retrieval based on embeddings. A recent survey conducted by [26] discusses the current state-of-the-art models, including methods based on terms, semantic retrieval, and neural approaches. KGs play a crucial role in this context because entity linking approaches heavily rely on structured data resources, such as KGs, to perform effective linking. Let’s take a closer look at each approach:

1. Information Retrieval (IR): Information Retrieval is a traditional approach for data retrieval that focuses on matching query terms with the textual representation of data. In this approach, documents or entities are represented as bags of words or terms. IR techniques use various algorithms such as vector space models, Boolean models, or probabilistic models to rank the documents or entities based on their relevance to a given query. In the context of datasets or KGs, IR-based approaches typically involve indexing the textual information associated with the data, such as entity names, descriptions, or attributes. Queries are formulated using

keywords or structured queries, and the retrieval process involves matching these queries against the indexed data. The retrieved results are ranked based on their relevance scores, which are usually computed using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or BM25 (Best Matching 25). Information retrieval (IR) techniques have reached a stage of being well-established and standardized. However, the challenge lies in optimizing the indexing and querying processes [5]. One approach that addresses this issue is LamAPI [4], which focuses on studying the impact of various configurations. LamAPI aims to improve the efficiency and effectiveness of indexing and querying by analyzing and experimenting with different setups

2. Dense Retrieval based on Embeddings: Dense retrieval is a relatively recent approach that leverages the power of dense vector embeddings for data retrieval. In this approach, each document or entity in the dataset is encoded into a dense vector representation (embedding) in a high-dimensional space. These embeddings are learned using techniques like word2vec, GloVe, or transformer-based models like BERT [[72] or GPT [22]. The retrieval process in dense retrieval involves encoding the query into an embedding as well. Then, the similarity between the query embedding and the document/entity embeddings is computed using metrics like cosine similarity or dot product. The top-k most similar documents or entities are retrieved based on their embedding similarities. Dense retrieval has gained popularity due to its ability to capture semantic relationships and similarities between data elements. It can handle more nuanced queries and is less reliant on exact keyword matches. Dense retrieval models can be trained using large-scale datasets and can be fine-tuned or optimized for specific retrieval tasks. In summary, Dense retrieval is currently a highly studied area of research. It has been extensively utilized to support text linking algorithms such as BLINK [76] and GENRE [18]

There are several approaches to entity retrieval, including syntactic matching, knowledge-based methods, and machine learning-based methods.

One of the most popular approaches for entity retrieval is based on information retrieval (IR) techniques. In this approach, each entity in the KG is represented by a set of features, such as its name, description, and attributes, and the query or mention is also represented in a similar way. The matching between the query and the candidate entities is then performed using standard IR techniques, such as cosine similarity or BM25. This approach has been widely used in many entity retrieval systems, such as Spotlight[7].

Another approach for entity retrieval is based on graph-based techniques. In this approach, the KG is treated as a graph, where the entities are nodes and the relations between them are edges. The candidate entities are then identified based on their proximity to the query node or mention node in the graph. There have been several recent studies on the use of graph-based techniques for entity retrieval. One such study is [20] which examines text-based approaches and how they evolved to leverage entities and their relations in the retrieval process. The study covers multiple aspects of graph-based models for entity-oriented search, providing an overview on link analysis and exploring graph-based text representation and retrieval, leveraging knowledge graphs for document or entity retrieval, building entity graphs from text, using graph matching

[7]https://www.dbpedia-spotlight.org/api

for querying with subgraphs, exploiting hypergraph-based representations, and ranking based on random walks on graphs. Another study is [23] which shows that graph embeddings are useful for entity-oriented search tasks. The study demonstrates empirically that encoding information from the knowledge graph into (graph) embeddings contributes to a higher increase in effectiveness of entity retrieval results than using plain word embeddings.

Recently, neural network-based approaches have shown promising results for entity retrieval. In these approaches, the KG and the query are represented as embeddings in a continuous vector space, and the matching between the query and the candidate entities is performed based on their proximity in the embedding space. These approaches include models like KGEmb[6] and K-Adapter[74].

Furthermore, some recent approaches leverage the context around the mention, such as the sentence or the paragraph in which the mention appears, to improve entity retrieval performance. These approaches, such as HIBERT[79] and HBM[47], use pre-trained language models such as BERT to encode the context and combine it with the mention representation to identify candidate entities.

Overall, entity retrieval is a challenging task that has received a significant amount of attention in recent years, and several approaches have been proposed to address it. Each approach has its strengths and weaknesses, and the choice of approach depends on various factors, such as the size of the KG, the type of queries, and the available computational resources.

3.3 EL: Entity Linking

According to [46], there has been a growing interest and popularity in Semantic Table Interpretation over the years, as evidenced by Figure 3.1. Additionally, Figure 3.2illustrates the distribution of papers by topic.

In the field of table annotation, various approaches have been proposed, which can be broadly categorized into two main categories: heuristic-based and machine learning-based methods. However, the most important task in table annotation is entity linking (CEA), which involves associating a mention in a table with a corresponding entity in a knowledge graph.

Upon performing entity linking over mentions in tabular data, two critical annotations can be obtained: Column Type Annotations (CTA) and Column Predicate Annotations (CPA). CTA provides information about the types of entities extracted for each column, while CPA identifies the predicates or relationships between the columns, which are extracted based on the relationships between the entities.

In summary, entity linking is a crucial step in table annotation, providing valuable insights into the data represented in the table. CTA and CPA can help us better understand the relationships and connections between the entities represented in the table, supporting various tasks such as information retrieval, data integration, and knowledge discovery. The STI approaches can be split into two main categories as following: **Heuristic based** The majority of approaches are children of the Semantic Table Interpretation Challenge and most of them address the problem

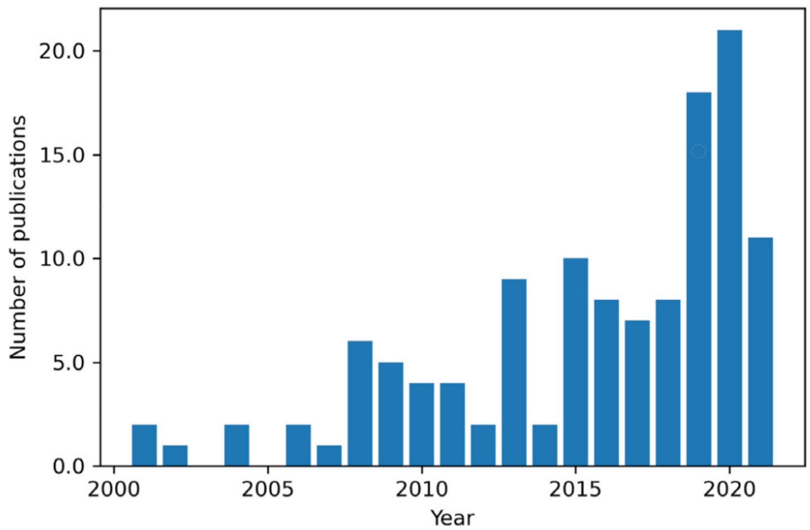


Figure 3.1: Number of publications of paper related to STI [46].

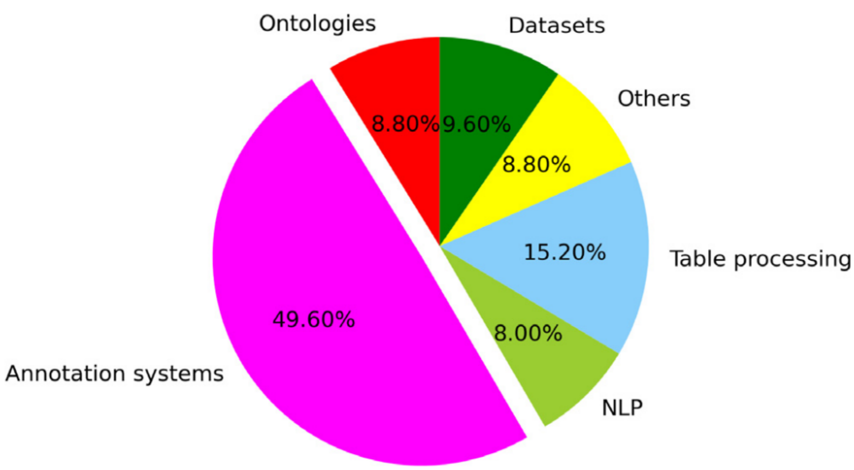


Figure 3.2: Distribution of the topics of the papers related to STI [46].

of column classification (CEA, CTA and CPA). Many approaches like [52, 31, 11, 8, 1, 68, 70] have been proposed during international Semantic Web Challenge on Tabular Data to Knowledge Graph Matching The most relevant is MTAB [52] that proposed an approach based on a framework which consist in two phases Structural Annotations (identify structure of the table) and Semantic Annotations (perform the annotation steps CEA, CTA and CPA). DAGOBAH [31] proposes an approach that is a two-stages annotation system consisting of an entity lookup step, followed by an entity scoring step. Finally, SELBAT [11] proposes an approach that basically provides an iterative process that performs Entity Linking (EL) on tables. The approaches cited above took part in different editions of the Semantic Web Challenge and improved their systems as a result.

Machine learning based Most of the approaches discussed here benefit from recent progress in machine learning and focus on column classification (task CTA). These approaches can also be useful for obtaining preliminary schema annotations of a table.

Chen et al. [7] proposed ColNet an approach based on embeddings techniques to predict the type annotations (CTA). This work uses a CNN to predict the type of a synthetic column, inspired by its successful application in text classification [9]. [71] proposed a new framework called Duduo for tabular column annotation, which achieved state-of-the-art performance on two benchmark datasets. They used pre-trained Transformer language models and multi-task learning to improve performance. Duduo was also shown to be data-efficient, achieving competitive performance using only 8 tokens per column or about 50% of the training data. Their approach outperformed previous state-of-the-art methods, including Sherlock [30], TURL [19] and Sato [78].

A recent survey presented in [46] enumerates all the main STI approaches. Distinguishingly, the approach presented in this thesis integrates both heuristic and machine learning methods, aiming to capitalize on the strengths of each. Furthermore, the proposed method assigns a confidence score, facilitating the interpretability of the method and allowing humans to actively participate in the process by providing feedback.

3.4 NIL Entities Problem

When it comes entity linking another important aspect to consider is regarding the NIL entities which are entities that are out of KG. Thus, it means that for those entities, representations are not available in the KG that is being used. According to the bar plot reported in Figure 3.3 the number of publications that taking into consideration this problem are significant over the years.

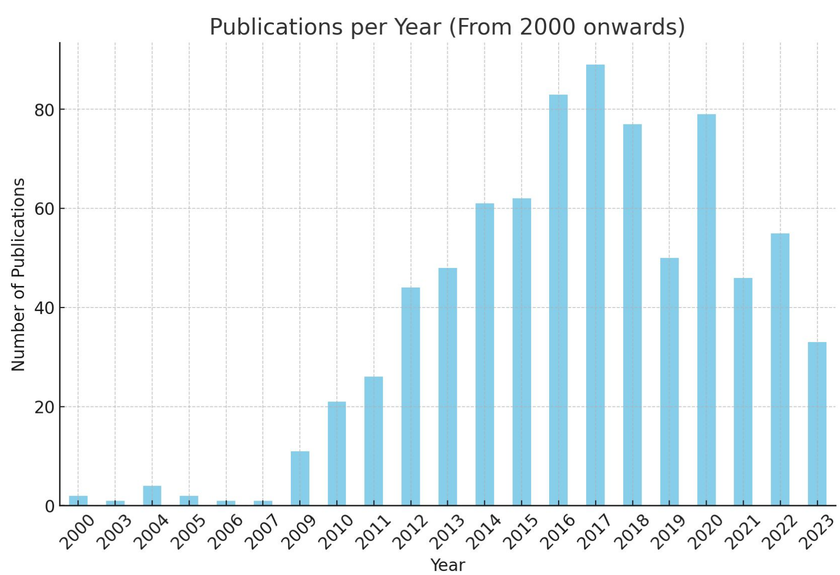


Figure 3.3: Number of publications of paper related to NIL entities

The problem regarding the terminology of NIL entities was first discussed by [32] in their paper titled ‘The Role of Knowledge in Determining Identity of Long-Tail Entities’. This study brought attention to NIL entities, which often pose unique challenges for entity linking systems due to their absence from the system’s knowledge resources, their non-redundancy, lack of frequency priors, potential for extreme ambiguity, and sheer numerosity. The author proposed an innovative method to tackle these challenges, involving the imputation of identifying knowledge to NIL entities from generalized characteristics. To do this, they utilized profile models based on background knowledge from Wikidata, a substantial database of structured data. Two profiling machines were developed using state-of-the-art neural models, a testament to the authors’ innovative approach to this problem. The study’s evaluation of these profiling machines, both in terms of their intrinsic behavior and their impact on determining the identity of NIL entities, is particularly noteworthy. Such insights are crucial for understanding the behavior of these models and their effectiveness in resolving NIL entities, and they represent a significant contribution to the field.

Others very recent approaches are presented in [80, 28, 33, 62]. [80] offers a significant contribution to the ongoing discourse on the challenges of NIL prediction in Entity Linking (EL). Despite the impressive strides made in EL models with the use of pre-trained language models, the issue of predicting NIL entities, which are mentions that do not correspond to an entity in the knowledge base, remains insufficiently addressed. The authors take a novel approach to this problem by dividing mentions linking to NIL into two categories: Missing Entity and Non-Entity Phrase. They then introduce a new dataset, NEL, designed specifically to address the NIL prediction problem. To assemble this dataset, they selected ambiguous entities as seeds and gathered relevant mention context from the Wikipedia corpus. They ensured the inclusion of mentions linking to NIL through a combination of human annotation and entity masking. The study’s methodology involves conducting experiments using widely used bi-encoder and cross-encoder EL models. The results shed light on the significant influence of both types of NIL mentions in the training data on the accuracy of NIL prediction. Their work, including the code and dataset, is openly accessible, providing valuable resources for further exploration in this field.

As mentioned, a significant aspect of entity linking is the handling of NIL entities, text mentions that do not have a corresponding entity in the associated knowledge base. This phenomenon has two main sub-tasks: NIL-detection, which involves identifying NIL-mentions in the text, and NIL-disambiguation, which seeks to ascertain if different NIL-mentions refer to the same out-of-knowledge base entity. This is the focus of a novel study in [33] that introduced a new dataset known as NILK for NIL-linking processing. The paper, deriving NILK from WikiData and Wikipedia dumps from two different timestamps, offers a substantial contribution to the NIL-linking task. In particular, it marks NIL-mentions for NIL-detection by extracting mentions which are associated with entities added newly to the Wikipedia text. For the purpose of NIL-disambiguation, it provides an entity label by marking NIL-mentions with WikiData IDs from the newer dump. This represents a pioneering effort as while there exist multiple datasets that can be adapted for NIL-detection, none of them sufficiently address the problem of NIL-disambiguation. The availability of the annotated NILK dataset, along with the code, presents a significant resource for researchers studying entity linking, particularly with regard to NIL entities. The NILK dataset can be accessed at: `https://zenodo.org/record/66075142`. This unique approach presented in the paper forms an integral part of the current state-of-the-art in entity linking research, specifically concerning the handling of NIL entities.

The presence of unlinkable (NIL) entities poses a significant challenge to the performance of Named Entity Linking methods and the downstream models that depend on them. While there have been numerous approaches proposed to handle NIL entities, most have focused primarily on clustering and prediction for general entities. However, the existence of NIL entities is not limited to general domains but extends into specialized fields like biomedical sciences, given the ever-expanding nature of scientific literature. This domain-specific challenge has been the focus of a recent study by Ruas and Couto [62]. In this paper, the authors introduce NILINKER, a model that incorporates a candidate retrieval module for biomedical NIL entities, along with a neural network that exploits the attention mechanism to identify the top-k relevant concepts from targeted Knowledge Bases (including MEDIC, CTD-Chemicals, ChEBI, HP, CTD-Anatomy, and Gene Ontology-Biological Process) that may partially represent a given NIL entity. In addition to NILINKER, the authors also present a new evaluation dataset named EvaNIL, which is designed to train and evaluate models focusing on the NIL entity linking task. This extensive dataset contains 846,165 documents (including abstracts and full-text biomedical articles), featuring 1,071,776 annotations across six different partitions: EvaNIL-MEDIC, EvaNIL-CTD-Chemicals, EvaNIL-ChEBI, EvaNIL-HP, EvaNIL-CTD-Anatomy, and EvaNIL-Gene Ontology-Biological Process. By integrating NILINKER into a graph-based Named Entity Linking model (REEL), the authors demonstrate its effectiveness in enhancing the performance of the Named Entity Linking model. This work provides valuable insights and resources for tackling the challenge of NIL entities in specialized domains such as biomedical sciences.

3.5 Human-in-the-Loop

3.5.1 Introduction

Human-in-the-Loop (HITL) is an interdisciplinary field of research that intersects machine learning, human-computer interaction, and data science. The core premise of HITL approaches is the symbiotic relationship between humans and machine learning models, aiming to leverage human expertise for model improvement while also reducing the manual effort required for tasks such as data labeling, validation, and interpretation. GPT models serve as a prime example of HITL in action, where reinforcement learning techniques have been employed to fine-tune the model using human feedback [22, 40]. The overarching objective is to create a harmonious interplay between human expertise and machine efficiency, thereby achieving superior model performance with minimized human effort.

3.5.2 Research Trends and Classification

According to a recent survey [77], there has been a surge in research interest in HITL methodologies, as illustrated in Figure 3.4. The survey categorizes existing works into three progressively related domains: (1) improving model performance through data processing, (2) enhancing model performance via interventional model training, and (3) developing system-independent human-in-the-loop approaches.

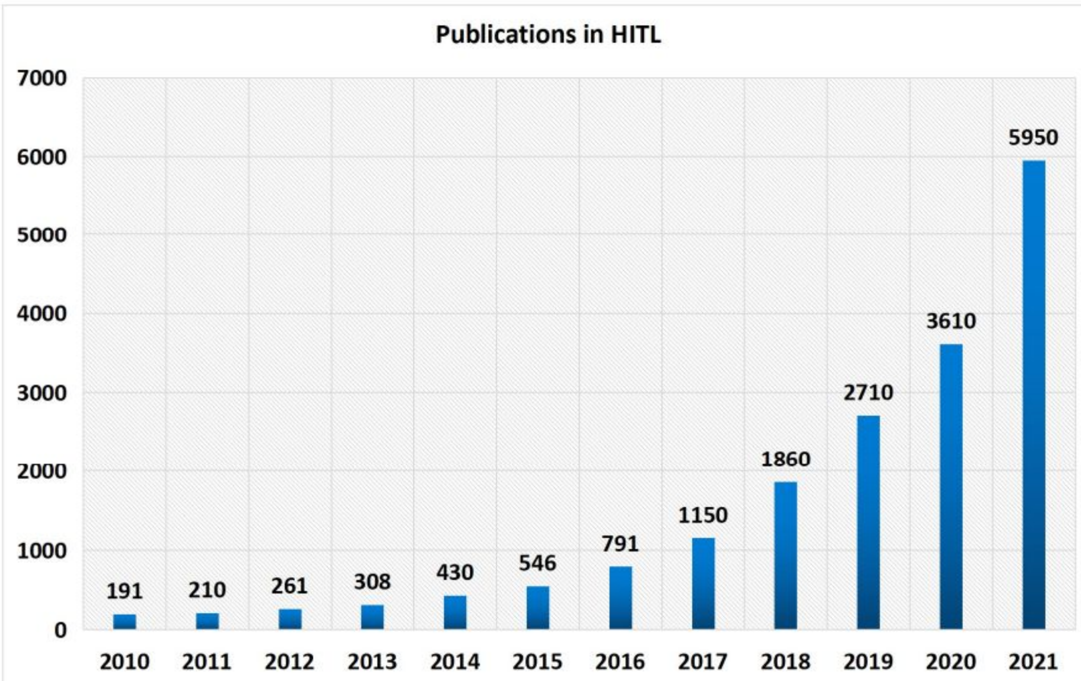


Figure 3.4: The increasing research interest in the human-in-the-loop (image from [77])

3.5.3 Types of Interactions

In a seminal paper [50], the authors delineate the various forms of interactions between humans and machine learning algorithms. They classify these interactions into three categories based on the locus of control in the learning process: (1) active learning, where the machine learning system maintains control, (2) interactive machine learning, which involves a more collaborative interaction between the human and the system, and (3) machine teaching, where human domain experts exert control over the learning process.

3.5.4 Challenges and Limitations

HITL (Human-In-The-Loop) methodologies, while offering significant potential for enhancing model performance, come with inherent challenges and limitations. One of the most prominent concerns is scalability. The degree of human engagement required may not be sustainable, especially when dealing with extensive or intricately detailed datasets. Furthermore, integrating human feedback into machine learning models opens the door to the incorporation of human biases. This can inadvertently compromise the model's fairness and its capacity to generalize across diverse scenarios. A particularly pressing challenge emerges when deploying HITL techniques for domain-specific adaptation. In such cases, models are prone to the phenomenon of 'catastrophic forgetting' This means that while the model becomes intensely attuned to the specificities of a given domain, it risks overshadowing or losing previously acquired knowledge. The model's hyper-focus on domain-specific data can lead it to underperform even on datasets where it had previously been trained effectively. [43, 42, 24, 56, 27].

3.5.5 Conclusion

In summary, the field of Human-in-the-Loop is gaining traction as a viable approach to enhance machine learning models by incorporating human expertise. Various tools in data processing, such as Trifacta Wrangler, KNIME, and Talend, are commonly used for tasks like data enrichment. However, these tools often lack support for HITL methodologies, thereby limiting their ability to fully leverage the benefits of human-machine collaboration. This has been acknowledged as a significant challenge by both the data management and machine learning communities [77, 50].

3.6 Data Pipelines

Data pipelines serve as the backbone of modern data architectures, playing a pivotal role in the orchestration of data flow across various stages of data processing and analytics. These pipelines are instrumental in the transformation of raw unstructured data into a more organized and usable form, enabling data-driven decision making processes [53, 2, 61]. The concept of a data pipeline is not new; however, the advent of big data, cloud computing, and advanced analytics has significantly expanded its scope and complexity. Today, data pipelines are not merely a set of data movement and transformation operations, but are complex systems that integrate various functionalities such as data ingestion, cleaning, transformation, enrichment, storage, and analysis. They are often designed to be highly scalable, fault-tolerant, and capable of handling a wide range of data types and formats [64, 10].

The importance of data pipelines is further accentuated by the increasing volume, velocity, and variety of data generated by a myriad of sources, including IoT devices, social media platforms, and enterprise systems. As organizations strive to become more data-centric, the role of data pipelines in automating labor-intensive tasks, ensuring data quality, and facilitating real-time analytics has become indispensable.

This section aims to provide a comprehensive overview of the state-of-the-art in data pipeline technologies, methodologies, and challenges, thus offering insight into the current landscape and future directions of this critical field.

3.6.1 Scalability in Data Processing

Executing cleaning, transformation, and linking at a large scale requires infrastructural components that allow for scalability. A definition of scalability is given by [29], which states that "scalability is the ability of a system to sustain increasing workloads by making use of additional resources." The implementation of a system with this characteristic is an essential step in a big data pipeline to avoid common performance bottlenecks. Usual issues arise in the following three areas:

- **CPU Usage**: This is the most common bottleneck. This issue occurs when the pipeline works correctly but the processing power of the CPU is not sufficient to handle the entire process.
- **Memory Usage**: The server does not have enough memory to organize the pipeline flow. This issue can also indicate a memory leak in the process.
- **Disk Usage**: This happens when the volume of disk space is fully occupied by the processed data.

Another common reason for implementing a scalable process is the flexibility of the new infrastructure. It allows one to change the priorities of the process, for example, by focusing more on one step rather than another, without losing the initial investment. In addition, a scalable process is future-proof for an eventual resize in the future.

After analyzing the need for a scalable process, there are two main ways of scaling:

- **Scaling Up (Vertical Scaling)**: This means using more powerful hardware and more memory. This method offers the best performance since everything works on the same machine. A possible limitation could be related to the speed of growth of the process; for a fast process, it represents just a short-term solution, and frequent updates become more and more expensive due to hardware limitations.
- **Scaling Out (Horizontal Scaling)**: This means adding new power across the infrastructure and not on the same machine. This solution uses parallel computing to increase the performance of the infrastructure and is valid also in the long term. At the same time, moving from a single machine to a distributed system leads to lower speed and higher complexity.

3.6.2 Architecture and Components

Modern data pipelines are often designed as a series of interconnected stages, each responsible for a specific task in the data processing workflow. These stages typically include:

1. **Data Ingestion**: The initial stage where data is collected from various sources. Technologies like Apache Kafka[8], Amazon Kinesis[9], and Flume[10] are often used for real-time data ingestion.

2. **Data Storage**: Once ingested, data is stored in a repository that could range from traditional databases like MySQL[11] to distributed file systems like Hadoop HDFS[12] or cloud-based solutions like Amazon S3[13].

3. **Data Processing**: This stage involves the transformation of data, often using batch or stream processing frameworks such as Apache Spark[14] or Flink [15].

4. **Data Analysis and Machine Learning**: Advanced analytics and machine learning models may be applied to the processed data. Libraries like scikit-learn[16], TensorFlow[17], and PyTorch[18] are commonly used.

5. **Data Visualization and Reporting**: The final processed data is often visualized using tools like Tableau[19], Power BI[20], or custom dashboards.

3.6.3 Technologies, Methodologies, and Tools in Data Pipelines

The landscape of data pipeline technologies is undergoing a significant transformation, primarily driven by the adoption of cloud-native architectures and managed services. Major cloud service providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform, offer specialized managed services for data pipelines—AWS Data Pipeline, Azure Data Factory, and Google Cloud Dataflow, respectively. These managed services are designed to offer scalable, reliable, and efficient data processing capabilities, thereby reducing the operational burden on organizations.

In terms of methodologies, the DevOps paradigm is increasingly influential in the development and maintenance of data pipelines. This approach integrates software development (Dev) and IT operations (Ops) to shorten the system development life cycle and provide continuous delivery. Key practices within this methodology, such as Continuous Integration and Continuous

[8] https://kafka.apache.org
[9] https://aws.amazon.com/kinesis
[10] https://flume.apache.org
[11] https://www.mysql.com/
[12] https://hadoop.apache.org
[13] https://aws.amazon.com/s3
[14] https://spark.apache.org/
[15] https://flink.apache.org/
[16] https://scikit-learn.org/
[17] https://www.tensorflow.org
[18] https://pytorch.org
[19] https://www.tableau.com
[20] https://powerbi.microsoft.com

Deployment (CI/CD), are being employed to facilitate rapid development cycles and to ensure the reliability and robustness of deployed pipelines.

To further streamline the management of data pipeline infrastructure, Infrastructure as Code (IaC) tools are gaining prominence. Tools like Terraform and Ansible enable the codification of infrastructure, allowing for version control, repeatability, and automated provisioning and de-provisioning of resources. This not only enhances the manageability of complex data pipeline architectures but also introduces an additional layer of reliability and auditability.

In summary, the state-of-the-art in data pipeline technologies is marked by a synergistic interplay between cloud-native solutions, DevOps methodologies, and Infrastructure as Code tools. These elements collectively contribute to the creation of more scalable, reliable, and manageable data pipelines.

3.6.4 Challenges and Limitations

Despite advancements, several challenges persist in the development and operation of data pipelines:

1. **Scalability**: Handling large volumes of data in real-time remains a challenge, requiring optimized algorithms and distributed computing resources.

2. **Data Quality**: Ensuring the accuracy and reliability of data as it moves through the pipeline is crucial, necessitating robust data validation and cleansing mechanisms.

3. **Security and Compliance**: With increasing regulations around data privacy, pipelines must incorporate strong security measures, including encryption and access controls.

4. **Complexity**: The growing complexity of data and analytics tasks requires sophisticated pipeline architectures, which can be difficult to manage and maintain.

3.6.5 Summary

Data pipelines are a critical component in modern data architectures, enabling the efficient and automated flow of data from source to insights. The state-of-the-art is characterized by cloud-native technologies, DevOps methodologies, and a focus on scalability and reliability. However, challenges related to data quality, security, and complexity remain areas for ongoing research and development.

4. Information Retrieval System

As discussed in Chapter 1, entity linking relies on a reliable source of data for reconciliation. In many cases, this data source is represented as a knowledge graph (KG). However, efficiently accessing and retrieving information from a knowledge graph poses unique challenges. To address these challenges, a robust information retrieval system specifically designed for knowledge graphs is required. Such a system enables the indexing of data within the knowledge graph, facilitating efficient lookup and retrieval of entities and their associated information, including relationships between graph elements.

This chapter focuses on the design and implementation of information retrieval systems tailored to knowledge graphs. Various techniques were explored and methodologies for indexing knowledge graph data, taking into consideration factors such as scalability, query performance, and support for complex queries.

By the end of this chapter, readers will gain a comprehensive understanding of building an information retrieval system for knowledge graphs. It will discuss the challenges involved and the techniques used to overcome them. This knowledge will provide readers with insights into the intricacies of information retrieval systems for knowledge graphs, enabling them to navigate the complexities of building and optimizing such systems for efficient data retrieval and knowledge exploration.

4.1 Knowledge Graph Indexing

The primary objective of indexing in Knowledge Graphs (KGs) is to facilitate efficient and accurate data retrieval for Information Retrieval Systems. This section outlines a general methodology that is adaptable to various types of KGs. The methodology focuses on two core challenges: Entity Lookup by Name and Relationship Retrieval. While this section provides a foundational understanding of the challenges in KG indexing, the subsequent section on Data Retrieval will explore more advanced Information Retrieval techniques.

Entity Lookup by Name: One of the core challenges in KG indexing is the efficient lookup of entities by their unique identifiers or names. Given the vast and heterogeneous nature of KGs, traditional lookup methods may not suffice. Information Retrieval techniques offer a promising avenue for addressing this challenge. For instance, an inverted index could serve as a foundational layer for more advanced techniques, such as tokenization and n-grams, which will be elaborated upon in the subsequent section.

Relationship Retrieval: Another significant challenge is the efficient retrieval of relationships between entities. Relationships in KGs are multifaceted, often involving various types and layers of connections. Efficiently indexing these relationships is crucial for any Information Retrieval

System tailored for KGs. As an example, one could employ an inverted index where each entity is mapped to a list of connected entities, categorized by the type of relationship. This would allow for quick, targeted queries, setting the stage for more advanced querying techniques.

4.2 Data Retrieval

Data retrieval plays a crucial role in performing entity linking tasks, especially when dealing with knowledge graphs (KG). The primary objective of data retrieval is to obtain relevant candidates from a given dataset, which, in this case, constitutes a KG. This step serves as the foundation for subsequent entity linking processes, allowing us to identify and associate textual mentions with their corresponding entities in the KG.

In the context of entity linking within tabular data, data retrieval becomes even more significant. Tabular data often contains structured information, such as columns and rows, and the mentions to be linked are embedded within these tables. Therefore, the data retrieval process involves searching and extracting relevant candidates from the tabular data based on the textual mentions encountered.

When performing data retrieval, various types of queries can be employed. These queries are designed to capture the essential characteristics of the textual mentions and locate potential candidate entities within the KG. Depending on the specific requirements of the task, queries can consider factors such as semantic similarity, contextual information, or domain-specific knowledge.

One crucial consideration in data retrieval is the management of the output size. Retrieving a large number of candidates can pose challenges in terms of processing time and computational resources. Therefore, it is important to define an appropriate output size that can be effectively managed and processed. This may involve setting limits on the number of candidates retrieved or implementing filtering mechanisms to prioritize the most relevant and promising candidates.

It is worth noting that data retrieval is not limited to tabular data. It can also be applied to structured and unstructured data in various domains. Whether dealing with textual documents, web pages, or other types of structured data, the goal remains the same – to retrieve relevant candidates that can be further processed and linked to entities in the KG.

Overall, data retrieval is a fundamental component of entity linking, enabling the identification and retrieval of relevant candidates from a given dataset. By leveraging appropriate querying techniques and considering output size limitations, data retrieval enhances the efficiency and effectiveness of the entity linking process, leading to accurate and reliable results.

Now, let's discuss the key characteristics in detail that define a data retrieval component:

- **limit**: An integer value that specifies the number of entities to retrieve. The default value is 100, which has been empirically demonstrated to provide a good level of coverage;

- **token**: Represents a word or a piece of text. By default, a retrieval system performs an exact match with the tokens of the input string.

- **fuzzy**: A common operator in data retrieval services that allows matching with a certain error tolerance in the input string. It may require slightly more processing time compared to exact token matching;

- **n-grams**: Given a value of ‘n,’ it represents a group of letters taken from a piece of text in a way that each group has ‘n’ letters. For instance, with n = 3 and the input ‘albert einstein,’ the text is split into [’alb’, ’lbe’, ’ber’, ’ert’, ...];

- **description**: Can be associated with an entity to provide additional context that can be used at query time or later for better disambiguation;

- **types**: Associated with the mention and can be used at query time to filter out results, as well as for better disambiguation in subsequent stages.

4.3 Information Retrieval System for Dataset with LamAPI

In this section, the proposed solution to the data retrieval problem is presented. Specifically, a tool named LamAPI, which stands for Label Matching API, has been developed.

LamAPI is implemented in Python using ElasticSearch and MongoDB. It comes with a Swagger documentation page for demonstration purposes, as shown in Figures 4.1 and 4.2. The `LamApi` Repository[1] is available, allowing the code to be downloaded and customized if necessary.

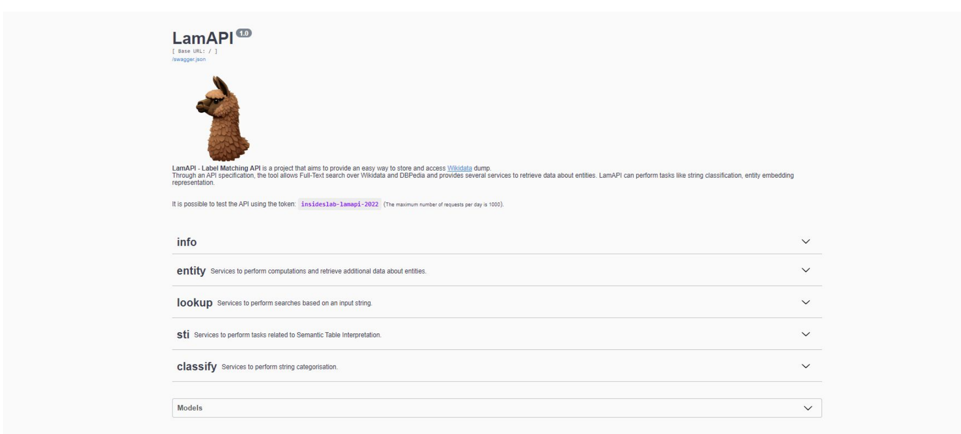


Figure 4.1: LamAPI documentation page

[1] bitbucket.org/discounimib/lamapi

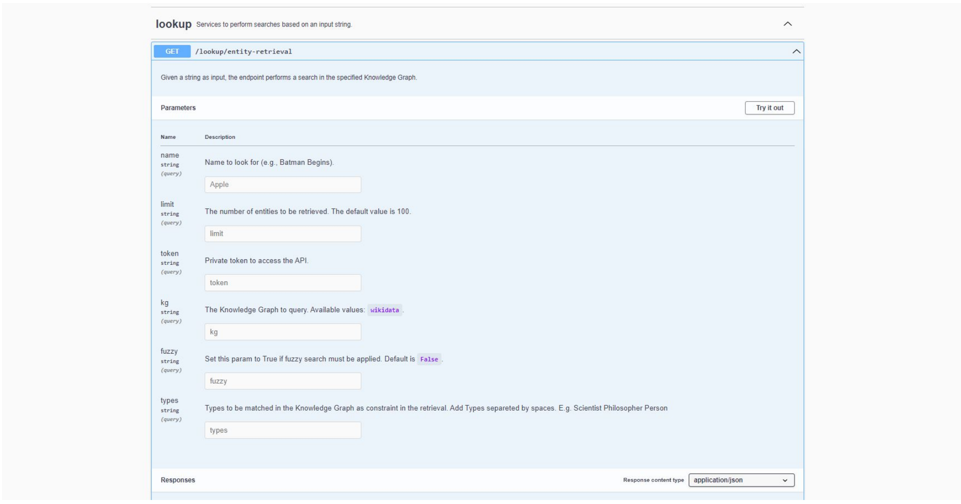


Figure 4.2: LamAPI Lookup service

For completeness, the list with the relative description of the LamAPI services is provided.
Types: given the unique *id* of an entity as input, it retrieves all the types of which the entity is an instance. The service relies on vector similarity measures among the types in KG to compute the answer. For DBpedia entities, the service returns both direct types, transitive types, and Wikidata types of the related entity, while for Wikidata, it returns only the list of concepts/types for the input entity.
Literals: given the unique *id* of an entity as input, it retrieves all relationships (predicates) and literal values (objects) associated with that entity.
Predicates: given the unique *id* of two entities as input, it retrieves all the relationships (predicates) between them.
Objects: given the unique *id* of an entity as input, it retrieves all related objects and predicates.
Labels: given the unique *id* of an entity as input, it retrieves all the related labels and aliases (`rdfs:label`).
Literal Recognizer: Given an input array consisting of a collection of strings, this component employs a series of regular expression rules in conjunction with the spaCy natural language processing library[2] to categorize each literal into its corresponding type. The types of literals recognized include *dates* (e.g., 1997-08-26, 1997.08.26, 1997/08/26), *numbers* (e.g., 2,797,800,564, 25 thousand, +/- 34657, 2 km), *URLs*, *email addresses*, and *times* (e.g., 12:30 pm, 12 pm).
Fasttext: Given an array of strings as input, the endpoint generates and returns the corresponding embedding representations using Fasttext vectors.

In Listing 4.1, the output of the lookup service is shown, displaying the first three candidates for the mention 'Jurassic World', along with the values of their respective features provided by LamAPI. As can be observed, LamAPI provides preliminary text-based features such as the mention's ntoken, the popularity of the entity derived from the KG, two features obtained from

[2] https://spacy.io/

Elastic (pos_score, es_score), the Jaccard score (jaccard_score and jaccardNgram_score), and finally, the cosine_similarity.

Listing 4.1: Returned candidates for the mention *Jurrasic World*.

```
{
    "id": "Q3512046",
    "name": "Jurassic world",
    "description": "2015 sci-fi adventure directed by Colin Trevorrow",
    "types": [...],
    "ambiguity_mention": 0.135,
    "corrects_tokens": 1,
    "ntoken_mention": 2,
    "ntoken_entity": 2,
    "length_mention": 14,
    "length_entity": 14,
    "popularity": 0.07,
    "pos_score": 0.01,
    "es_score": 1,
    "ed_score": 1,
    "jaccard_score": 1,
    "jaccardNgram_score": 1,
    "cosine_similarity": 1
},
{
    "id": "Q21877685",
    "name": "Jurassic World",
    "description": "2018 5th Jurassic Park film directed by Juan Antonio",
    "types": [...],
    "ntoken": 2,
    "ambiguity_mention": 0.135,
    "corrects_tokens": 1,
    "ntoken_mention": 2,
    "ntoken_entity": 2,
    "length_mention": 14,
    "length_entity": 14,
    "popularity": 0.05,
    "pos_score": 0.03,
    "es_score": 0.988,
    "ed_score": 1,
    "jaccard_score": 1,
    "jaccardNgram_score": 1,
    "cosine_similarity": 1
},
{
    "id": "Q2336369",
    "name": "Jurassic World",
    "description": "American media franchise",
    "types": [...],
    "ntoken": 2,
    "ambiguity_mention": 0.135,
    "corrects_tokens": 1,
    "ntoken_mention": 2,
    "ntoken_entity": 2,
    "length_mention": 14,
    "length_entity": 14,
    "popularity": 0.03,
    "pos_score": 0.04,
    "es_score": 0.985,
    "ed_score": 1,
    "jaccard_score": 1,
    "jaccardNgram_score": 1,
    "cosine_similarity": 1
}
```

In Listing 4.2, the output from the objects service is displayed. Specifically, this output illustrates how relationships are structured. Within the 'objects' dictionary, the dictionary's keys represent the objects of the relationships. For each key, the associated values are arrays that enumerate the predicates corresponding to those relationships.

Listing 4.2: Returned relationships between candidate Q3512046 and other entities.

```
{
  "wikidata": {
    "Q3512046": {
      "objects": {
```

```
        "Q229390": [
          "P31"
        ],
        "Q11424": [
          "P31"
        ],
        "Q261636": [
          "P31"
        ],
        "Q5145625": [
          "P57",
          "P161",
          "P58"
        ],
        "Q471839": [
          "P136"
        ],
        "Q319221": [
          "P136"
        ],
        "Q188473": [
          "P136"
        ],
        "Q1077883": [
          "P136"
        ],
        "Q430": [
          "P921"
        ],
        "Q17862144": [
          "P179"
        ],
        "Q1860": [
          "P364"
        ]
        ...
      }
    }
  }
}
```


In Listing 4.3, the output from the literals service is displayed. Specifically, this output illustrates how relationships are structured. Within the 'literals' dictionary, the keys signify the datatypes associated with the relationships. Each key has an array of values, highlighting the literal values pertinent to those relationships.

Listing 4.3: Relationships: returned the relationships that candidate Q3512046 has with the other literals values.


```
{
  "wikidata": {
    "Q3512046": {
      "literals": {
        "GEOSHAPE": {},
        "DATETIME": {
          "P577": [
            "+2015-06-12T00:00:00Z",
            "+2015-06-11T00:00:00Z",
            "+2015-06-10T00:00:00Z",
            "+2015-05-29T00:00:00Z"
          ]
        },
        "MUSICAL_NOTATION": {},
        "TABULAR_DATA": {},
        "MATH": {},
        "NUMBER": {
          "P2047": [
            "+124"
          ],
          "P2142": [
            "+1670400637",
            "+652270625"
          ],
          "P2130": [
            "+150000000"
          ]
        },
```

```
        "STRING": {
          "P345": [
            "tt0369610"
          ],
          "P646": [
            "/m/051179"
          ],
          "P856": [
            "http://www.jurassicworld.com"
          ],
          "P1258": [
            "m/jurassic_world"
          ],
          "P373": [
            "Jurassic World"
          ],
          "P1712": [
            "movie/jurassic-world"
          ],
          "P1562": [
            "v576633"
          ],
          "P1237": [
            "jurassicpark4"
          ],
          ...
        }
        ...
      }
    }
  }
}
```

Now, let's delve into the workings of the `Literals-recognizer`. In Listing 4.4, the data fed into the Literal Recognizer Service is illustrated. Subsequently, in Listing 4.5, the result obtained when the service processes the data from Listing 4.4 is displayed.

Listing 4.4: Input to Literal Recognizer Service

```
{
  "json": [
    "50",
    "12/11/1997",
    "https://www.unimib.it/",
    "mario.rossi@gmail.it",
    "Mount Blanc is located in Aosta Valley"
  ]
}
```

Listing 4.5: Results from Literal Recognizer Service

```
{
  "50": {
    "datatype": "INTEGER",
    "classification": "NUMBER",
    "tag": "LIT",
    "xml_datatype": "xsd:integer"
  },
  "12/11/1997": {
    "datatype": "DATE",
    "classification": "DATETIME",
    "tag": "LIT",
    "xml_datatype": "xsd:date"
  },
  "https://www.unimib.it/": {
    "datatype": "URL",
    "classification": "STRING",
    "tag": "LIT",
    "xml_datatype": "xs:anyURI"
  },
  "mario.rossi@gmail.it": {
    "datatype": "EMAIL",
    "classification": "STRING",
    "tag": "LIT",
    "xml_datatype": "xsd:string"
  },
  "Mount Blanc is located in Aosta Valley": {
```

```
    "datatype": "STRING",
    "classification": "STRING",
    "tag": "NE",
    "xml_datatype": "xsd:string"
  }
}
```

Figure 4.3 presents the comprehensive workflow of lamAPI construction. Initially, the process begins with the large Wikidata dump which, when compressed, is approximately 77GB in size. This data can be accessed at `https://dumps.wikimedia.org/wikidatawiki/entities/`. This file is processed in its compressed form and the intermediate results are stored in a MongoDB database. Subsequently, an index is created on the names of the entities in Elasticsearch.

Upon completion of these steps, several important services become available:

- **Lookup**: This service performs searches in the KG, taking a text string as input and returning a set of candidate entities.
- **Objects**: This service reveals the relationships between entities in the KG.
- **Literals**: This service provides the relationships between entities and literal values, such as numbers, dates, or text.

These services facilitate various operations over the knowledge graph, thereby enhancing the utility and accessibility of the stored data.

DBpedia, Wikidata and the like are very large KGs that require an enormous amount of time and resources to perform CR. Additionally, information available in turtle format is excellent for representing relationships among entities, but they are unsuitable for applying CR algorithms. This issue has been tackled by devising a more condensed data representation by utilizing MongoDB collections, which can be indexed for swift retrieval of the intended data.

For each indexed KG, the relative dump has been downloaded and parsed to store all triples in a local copy. For Wikidata, a single file named 'latest-all.json.bz2' of size around 78 GB has been parsed, while for DBpedia multiple turtle file have been parsed to create a complete dump of the KG. Subsequently, an ElasticSearch[3] index has been constructed, leveraging an engine designed to search and analyze extensive data volumes in nearly real-time swiftly. These customized local copies of the KGs are then used to create endpoints to provide EL retrieval services. The advantage is that these services can work on partitions of the original KGs to improve performance by saving time and using fewer resources.

The entire process illustrated in Figure 4.3 takes approximately 36 hours to complete. The bulk of this time (around 31 hours) is spent processing the large Wikidata dump file. This involves processing the compressed file line by line, decompressing a single line at a time, and storing the results in the MongoDB database. The remaining 5 hours are used to create the Elasticsearch index. This time investment ensures efficient and accurate operations over the knowledge graph, thereby furnishing a comprehensive and searchable database.

[3]www.elastic.co

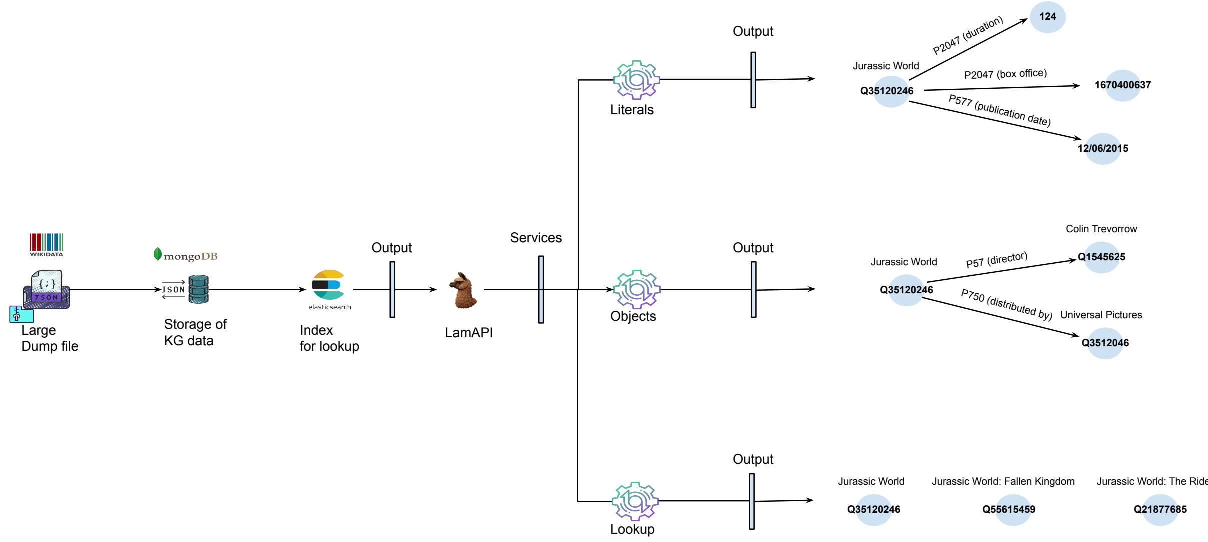


Figure 4.3: The high level workflow to build LamAPI

4.4 Evaluation of LamAPI Performance

In this section, the analysis focused on evaluating the LamAPI output quality is presented. Specifically, it scrutinizes the distribution of correct candidates across a variety of datasets derived from the SemTab challenge. The datasets enlisted for the evaluation include *Round1-T2D*, *Round3*, *Round4*, *2T*, *HardTableR2*, and *HardTableR3*. Below, each dataset is elucidated in greater detail:

- **Round1-T2D** [60]: extracted from the T2Dv2 Gold Standard, this dataset comprises manually annotated correspondences such as row-to-instance, attribute-to-property, and table-to-class, distributed across 779 web tables;

- **Round3** [35]: this dataset stands out for featuring a substantial number of abbreviated individual names, like 'J. F. Kennedy' introducing a nuanced layer of complexity to the dataset;

- **Round4** [37]: constructed during the SemTab 2020, it encompasses 22,207 tables, averaging 21 rows per table;

- **2T (Tough Table)** [17]: recognized for its high-quality tables which incorporate cells with ambiguous identifiers, typos, and misspelled entity names, it houses approximately 70,000 unique cells in 180 tables. It supports the testing of CR with misspelled words and offers the scope to carry out comparative analyses for both Wikidata and DBpedia KG through identical tables;

- **HardTableR2** [16]: a synthetic dataset characterized by an average of 16 rows per table and mentions that predominantly consist of one or two tokens, introducing a considerable degree of ambiguity and making the disambiguation process particularly challenging;

- **HardTableR3** [16]: similar to HardTableR2 but distinguished by the fact that each table contains only one NE-column, coupled with an average of 8 rows per table. This configuration makes the disambiguation process more challenging, primarily because the sole reliance on a single NE-column can limit context and introduce ambiguity.

It is worth noting that the majority of the aforementioned datasets were formulated utilizing an automated data generator operating through a SPARQL endpoint. The initiative behind this approach was to generate tabular data resembling those found on the web while securing a reasonable variation in terms of size and the span of classes and properties from diverse domains.

A comprehensive portrayal of these datasets, encapsulating a range of domains and accompanied by ground truths, is presented. Table 4.1 delineates the statistics concerning the six datasets: *Round1_T2D*, *Round3*, *Round4*, *2T-2020*, *HardTableR2*, and *HardTableR3*.

Table 4.1: Statistics of the Datasets Used in the Experiments

dataset	**table**	**columns**	**rows**	**# entities (CEA)**	**# classes (CTA)**	**# predicates (CPA)**
Round1_T2D	64	323	9089	8078	119	115
Round3	2161	9736	152753	390456	5761	7574
Round4	22207	78750	475897	994920	31921	56475
2T-2020	180	802	194438	667243	539	0
HardTableR2	1750	5589	29280	47439	2190	3835
HardTableR3	7207	17902	58949	58948	7206	10694

The validation process starts with a set of mentions M, and a number k of candidates associated with each mention. The *Lookup* service returns a set of candidates E_m that includes all the candidates found. The returned set is then checked against the 2T to verify which among the correct entities are present and in what position in the ranked results in E_m. The coverage is computed following this formula:

$$coverage = \frac{\#\, candidates\ found}{\#\, total\ candidates\ to\ find} \tag{4.1}$$

Where # represents ‘number of’.
In Table 4.2 the various coverage values are presented for lookup based on label matching on a mention by enabling *fuzzy* and *n-grams* searches. The experiments were conducted using 20 parallel processes on a server with 40 CPU(s) Intel Xeon Silver 4114 CPU @ 2.20GHz and 40GB RAM.

Table 4.3 and 4.4 show the coverage using the constraint on types. To select and expand types, four methods were applied.

1. **Type**: This method considers only the type or set of types (seed types) indicated in the call to the Lookup service, and it does not carry out any expansion of types.

2. **Type Co-occurrency**: For the seed types, it extracts additional types based on the co-occurency of types in the KG. The co-occurency score represents the number of times each type co-occurs with another type in a KG at entity level.

3. **Type Cosine Similarity**: The seed types are extended by the cosine similarity of RDF2Vec[4].

4. **Soft Inference**: The seed types are extended using a Feed Forward Neural Network that takes as input the RDF2Vec vector of an entity, linked to a mention and predicts the possible types for the input entity [15].

In Table 4.3, it is possible to notice that the first method achieves a higher coverage. The best result is obtained by adding two types. Co-occurrencies and Type Cosine Similarity are both idempotent methods. The Soft Inference technique uses the entities obtained by a prelinking. Not all entity vectors are available, so extending the set of types is not always possible. In Table 4.4, the results for Wikidata are presented. Also, in this case, the best results here are achieved using the first method. The achieved coverage is highest because this KG has a comprehensive hierarchy with more detailed types.

Even if lower, the coverage values achieved with type expansion methods are promising. It must be considered that the exact type to use as a filter is often not known a priori in real scenarios. For instance, in order to select a type, a user needs to be familiar with the profile of a KG and understand how it is utilized to describe entities. Owing to the methods described above, the search results will include entities that belong to other types yet remain related to the input.

Table 4.2: Coverage results and response times for different searches in Wikidata and DBpedia v. 2022.03.01.

Methods	DBpedia		Wikidata	
	Coverage	Time	Coverage	Time
N-gram	0,842	228 s	0,787	649 s
Fuzzy	0,806	226 s	0,805	766 s
Token	0,561	227 s	0,530	230 s
N-gram + Fuzzy	0,891	267 s	0,926	1649 s
N-gram + Token	0,883	229 s	0,891	807 s
Fuzzy + Token	0,812	226 s	0,825	773 s
N-gram + Fuzzy + Token	**0,895**	270 s	**0,929**	1577 s

Table 4.3: Coverage results for 2T DBpedia.

Methods	w/o type	1 type	2 types	3	4	5	6	7	8	9	10
Type	0,892	0,904	**0,905**	0,904	0,889	0,884	0,879	0,872	0,870	0,867	0,848
Type Co-occurency	0,892	0,886	**0,896**	0,886	0,856	0,884	0,830	0,834	0,834	0,833	0,823
Type Cosine Similarity	0,892	**0,892**	0,886	0,889	0,885	0,881	0,873	0,869	0,825	0,825	0,830
Soft Inference	**0,892**	0,885	0,872	0,884	0,882	0,879	0,885	0,886	0,878	0,874	0,869

[4] rdf2vec.org

Table 4.4: Coverage results for 2T Wikidata.

Methods	w/o type	1 types	2 types	3	4	5	6	7	8	9	10
Type	0,929	0,941	0,939	0,946	0,946	**0,947**	**0,947**	0,945	0,945	0,943	0,944
Type Co-occurency	0,929	0,854	0,808	0,796	0,793	0,795	0,797	0,797	0,795	0,796	0,795
Type Cosine Similarity	0,929	0,853	0,853	0,852	0,851	0,850	0,849	0,849	0,848	0,847	0,845

The validation previously proposed referred to the results obtained through varying configurations while maintaining the constant use of the 2T-2020 dataset. Currently, a more general validation is being introduced that takes into consideration different datasets to authenticate the performance achieved by LamAPI in terms of coverage across diverse datasets.

Figure 4.4 depicts how the coverage score varies with the number of candidates. The results reveal that for most datasets, the coverage remains consistently around 0.90 when the number of candidates ranges from 20 to 30. Nevertheless, there are exceptions, notably in the case of *Round3* and *HardTableR3*, where the coverage reaches approximately 0.80 with a hundred candidates. The primary factor contributing to this score in *Round3* is the prevalence of name abbreviations, making them challenging for LamAPI to be identified. In contrast, *HardTableR3* faces disambiguation challenges, as most cells consist of only one token, providing a little context for disambiguation.

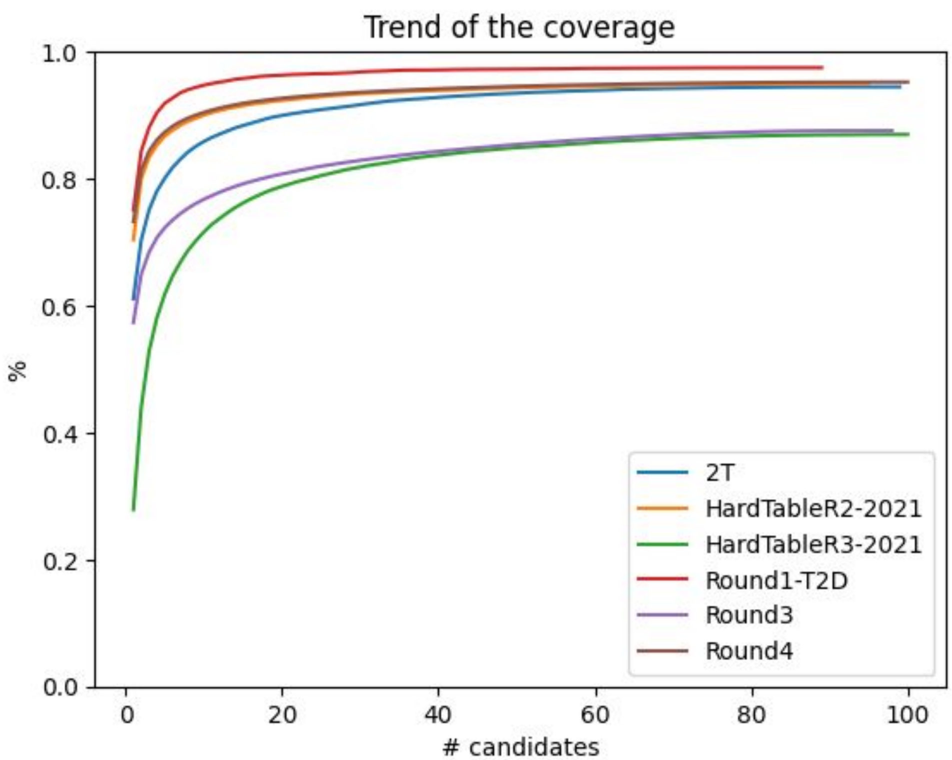


Figure 4.4: Coverage trends across different considered datasets

The other experiment concerns the average position of the correct candidate across the different datasets. The Figure 4.5 shows the average position of the correct entity for each dataset.

HardTableR3 obtain the worst results because one-token mentions have a wide range of possible candidates, so the correct candidates are often in the lower ranking position. *Round3* is critical for abbreviations of people names, but when the correct candidate is found, it is detected around the 4-th position. Also *2T-2020* has an average position of the correct candidate higher (around 3) because this dataset is full of misspelled mentions, so it's more complicated to retrieve the correct candidate in the first position. *HardTableR2*, *Round1_T2D* and *Round4* are easier since they have fewer typos and misspelled mentions.

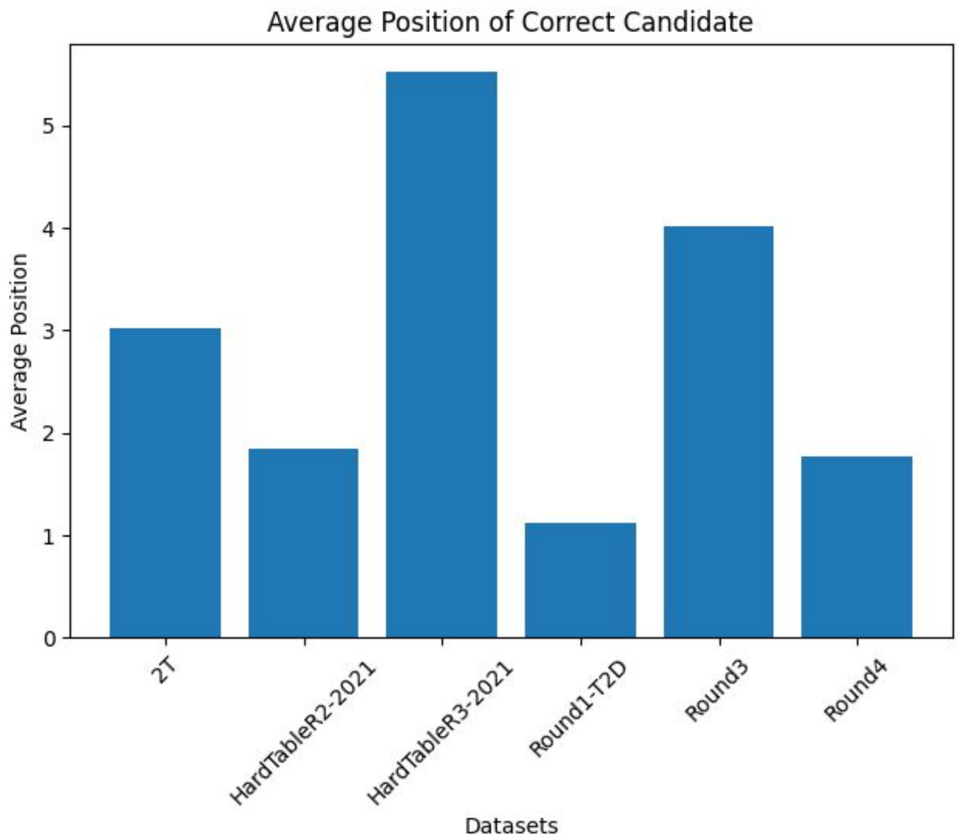


Figure 4.5: Average position of the correct candidate across different considered datasets

5. Semantic Enrichment of Tabular Data

The approach presented in this chapter addresses the task of entity linking by leveraging the inherent structure and context of tabular data. It combines heuristic and machine learning techniques to identify and link mentions in the table to corresponding entities in the knowledge graph (KG). Additionally, the proposed approach incorporates a scoring mechanism that provides confidence scores for the linked entities, facilitating result evaluation and interpretation.

A comprehensive analysis of the proposed algorithm is provided, examining its strengths, limitations, and potential applications. Furthermore, a thorough evaluation of its performance and effectiveness is conducted by comparing it against established benchmarks and state-of-the-art approaches for entity linking. The evaluation process encompasses various metrics and a large number of datasets, enabling a comprehensive assessment of the algorithm's capabilities.

By the end of this chapter, readers will gain a thorough understanding of the proposed entity linking algorithm for tabular data, its driving ideas, and its performance in comparison to existing approaches. This knowledge will serve as a solid foundation for further advancements in entity linking over tabular data, paving the way for improved data integration, knowledge discovery, and information retrieval.

5.1 Overview of the Proposed Approach

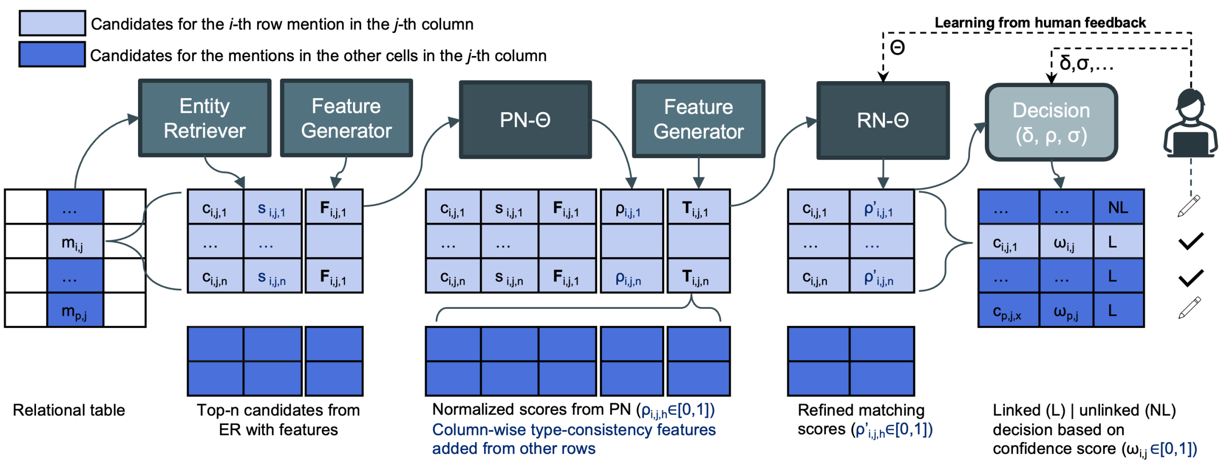


Figure 5.1: Linking workflow with Human-In-The-Loop feedback.

The proposed linking process of a table T follows the workflow illustrated in Figure 5.1. This workflow encompasses the following key steps:

1. **Candidate retrieval**: Here, the reference KG is queried to identify candidate entities related to each mention in the table. The result is a weighted list of candidates, where the (typically unbounded) weight represents the confidence in their quality.

2. **Candidate Ranking**: Employing a pretrained neural network with weights symbolized by Θ, the ranking of candidates is a two-step process that generates normalized confidence scores, initially denoted as ρ and subsequently as ρ'. These scores are instrumental in selecting the most suitable candidate for each mention

3. **Uncertainty estimation and decision**: This task involves the use of a score δ that considers the distance between the scores of the top-two candidates for a mention, and the confidence score ρ of that mention. The weighted sum of δ and ρ produces a global score ω for ranking all candidates; A threshold σ discriminates between *linked* and *unlinked* mentions.

4. **Human revision**: The human actor reviews the most uncertain links (also referred to as *annotations*). The domain knowledge helps validate or correct the results. The review process considers the mentions featuring candidates with progressively increasing values of ω with the objective of improving the overall F1 measure.

By following this structured workflow, the approach ensures the effective linking to entities in the reference KG with their mentions in table T, combining learning-based automated techniques with human expertise.

At the end of the linking process, human intervention is essential to improve the quality of the results and, thus, the algorithm's effectiveness. The essence of HITL (Human-in-the-Loop) is to enhance the performance of a matching algorithm, specifically, an entity linking algorithm, by necessitating a controlled level of user interaction. HITL matching usually considers two aspects of the problem that can possibly be combined: how to identify uncertain links, which can speed up the revision process by pinpointing mismatched mentions first, and how to maximize the algorithm's performance, simultaneously minimizing the user's effort, measured in terms of the percentage of links to review.

As illustrated in Figure 5.1, the user's actions may be propagated back to *Candidate ranking* and *Uncertainty estimation and decision* in the entity linking process. This effect could result in a fine-tuned version of the neural network model for candidate ranking and a change in the rules for uncertainty estimation and decision. For example, it may lead to the definition of revised criteria to combine thresholds δ and scores ρ for selecting linked entities.

5.1.1 Data Preparation & Candidate Retrieval

Data preparation and candidate retrieval are crucial tasks that precede entity linking.

The data preparation task primarily involves identifying the most probable datatype of each column. Special attention is accorded to columns containing Named Entities (NE) and Literals (LIT). To distinguish between these two categories of columns, a synergistic approach involving regular expressions and Natural Language Processing (NLP) techniques is utilized, as elaborated in Section 4.3. In addition, data preparation also includes cleaning operations, such as converting strings to lowercase, removing extra spaces, and eliminating problematic characters.

Candidate retrieval is done following the identification of the column data types. This step encompasses the retrieval of candidates for all mentions in order to obtain a set of candidates. Further details on this process are elaborated in Section 4.2.

5.1.2 The Entity Linking Algorithm

In the realm of entity linking, the primary goal is to address the challenge of linking 'mentions', defined as labels or strings found in cells within relational tables, to entities in a reference Knowledge Graph (KG) like WikiData or DBPedia.

The task of entity linking comprises three main objectives: *Cell Entity Annotation (CEA)*, *Column Type Annotation (CTA)*, and *Column Property Annotation (CPA)*. The CEA objective involves annotating table cells with entities from the reference KG. The CTA aims to detect the semantic type of mentions within the same column. Lastly, the CPA aims to identify the semantic relationship between the mentions in the same row and, subsequently, the relationship between the columns containing those mentions. The CTA and CPA tasks are termed as *'schema-level annotations',* as they concern the analysis of the table's schema.

The CEA task is central in entity linking, as it streamlines both CTA and CPA activities. Upon obtaining mention annotations within the table through CEA, estimations for CTA and CPA can be conducted. The estimation process for both tasks is executed by employing majority voting across occurrences. In the context of the CTA task, type frequencies of the winning candidates are aggregated, followed by majority voting to determine the prevailing type. Similarly, for the CPA task, predicate frequencies of the winning candidates are gathered, and a subsequent majority vote is conducted to ascertain the dominant predicate.

Finally, after obtaining a set of candidates for each mention through the data retrieval phase, the objective of the entity linking algorithm is to provide a ranking of these candidates. That is, each mention will be associated with a ranked list of candidates where the first candidate (top one) is considered the most probable. This top candidate can be viewed as the winning candidate for that mention.

An EL algorithm can employ ML techniques, rely solely on heuristic methods, or use a combination of both. To compute the similarity between a mention and a candidate, the entity linking algorithm needs a set of similarity measures, referred to as a set of features.

Using these features, the EL algorithm can compute a confidence score for each candidate, which is then used to generate the final ranking. This confidence score can either be bounded or unbounded, but a bounded score (for example, ranging from 0 to 1) is preferred because it provides an interpretable measure of confidence.

5.1.3 Decision

In the *Decision* phase, strategies are employed to determine whether to establish a link. One approach involves setting a threshold for the confidence score (ρ) obtained from the entity linking algorithm, below which a mention should be considered unlinked. Another strategy involves considering the difference (δ) in confidence scores (ρ) between the top two candidates for a mention. This measure helps to gauge the degree of uncertainty. Ultimately, various strategies can be devised that incorporate both ρ and δ values to assess the level of uncertainty and make the final decision regarding linking or not.

5.2 Entity Linking over Tabular Data

In this section, the proposed solution to address the challenge of entity linking over tabular data is presented. A tool named *Alligator* has been developed, an acronym for Automated Learning and Linking for Intelligent Graph-Based Association of Tabular Objects and Relationships.

The foundation of this approach is a feed-forward neural network, trained to excel in entity linking tasks within the realm of tabular data.

To ensure the model's effectiveness, the network is trained using a meticulously curated dataset known as the Gold Standards (GS). This dataset serves as the gold standard for entity linking, providing the model with a robust foundation for learning and making precise associations.

In addition to the training data, a set of well-crafted features has been created to feed the model. These features play a pivotal role in enhancing the model's capacity to extract valuable information from the tabular data and the related candidates, enabling it to identify and link entities with a high degree of precision.

The entity linking solution, anchored by this neural network and complemented by feature engineering, represents a powerful tool for tackling the intricacies of entity linking in the context of tabular data. It is this combination of advanced technology and carefully designed features that empower the model to excel in this complex task.

5.2.1 Feature Engineering

As illustrated in Figure 5.2, three distinct vectors are defined: a vector corresponding to the mention (**V1**), a vector corresponding to the combined mention and entity (**V2**), and a vector corresponding to the entity alone (**V3**). Each vector encapsulates a specific set of features: *Mention*, *Mention-Entity*, and *Entity* features, respectively. So, **V1** pertains to the mention 'Jurassic World' and encompasses features such as string length and token count. **V2** is associated with the composite of mention and entity, encapsulating features like Levenshtein distance and Jaccard similarity coefficient. Lastly, **V3** focuses solely on the entity and includes features such as popularity and the token count of the entity's label. These vectors (**V1**, **V2**, and **V3**) serve as input to the machine learning model, which then generates a confidence score. This score assesses the likelihood that the candidate **Q3512046** is the correct entity to be linked to the mention 'Jurassic World.'

In particular, the ML model (also referred to as the Neural Ranker) will identify the weights $w_1, w_2, ..., w_n$ to assign to the individual features, learning them from the correct candidates observed during the training phase.

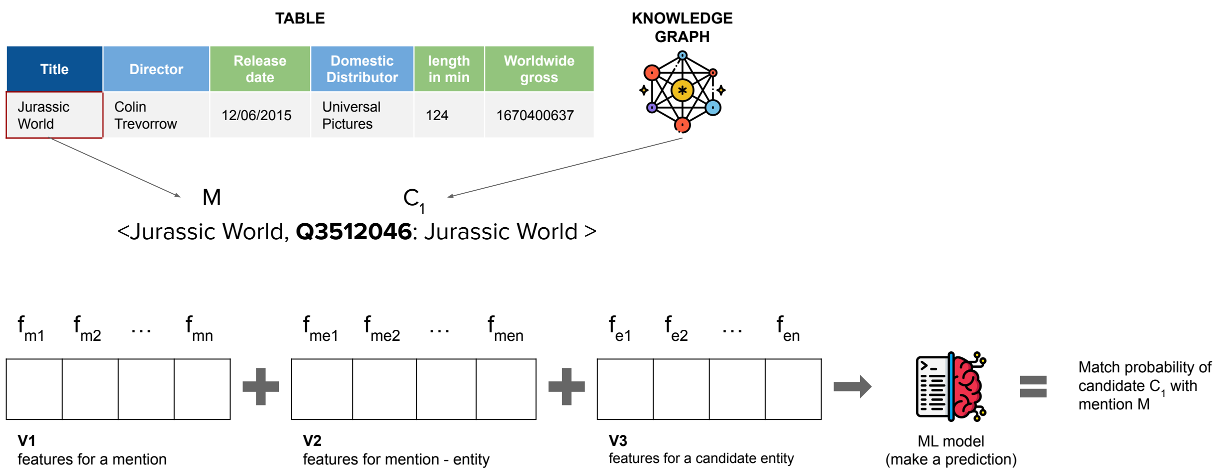


Figure 5.2: Example of feature vector.

Table 5.1 lists all the features considered in this study. For each feature, it is specified whether it is associated only with the mention, only with the candidate entity, or with both. The features can exhibit different natures. Some features are directly tied to the Information Retrieval (IR) system, meaning they depend on its characteristics. For example, if the scoring function of the IR system is modified, these features will consequently change. Other features remain stable, these include text similarity-based features such as edit distance or Jaccard similarity. Additionally, there are features that are contingent on the Knowledge Graph (KG) being used. This implies that as the KG improves or evolves, certain features may also improve, potentially enhancing the system's ability to disambiguate candidates in specific cases.

Below is a concise description of the various features utilized in the entity linking process:

- **ambiguity_mention**: this feature quantifies the level of ambiguity associated with a given mention. For example, a mention such as 'Paris' would yield a high ambiguity score, approaching 1. The score is computed through a query to an Information Retrieval (IR) system;

- **correct_token**: this feature yields a score representing an estimation of the number of accurately spelled tokens in the mention, serving as a proxy for spelling reliability. This score is also computed through a query to an IR system;

- **ntoken_mention**: the number of tokens in the mention *m*. This feature serves as an indication of the ambiguity of the mention;

- **ntoken_entity**: this feature represents the number of tokens present in the entity's name. This feature serves as an indication of the ambiguity of the entity;

- **length_mention**: this feature quantifies the length of the mention in terms of the number of characters;

Table 5.1: Features Utilized in the Feature Vector

Name	Range	Feature Type
ambiguity_mention	[0, 1]	Mention
correct_token	[0, 1]	Mention
ntoken_mention	[0, ∞)	Mention
ntoken_entity	[1, ∞)	Entity
length_mention	[0, ∞)	Mention
length_entity	[0, ∞)	Entity
popularity	[0, 1]	Entity
pos_score	[0, 1]	Mention-Entity
es_score	[0, 1]	Mention-Entity
ed_score	[0, 1]	Mention-Entity
jaccard_score	[0, 1]	Mention-Entity
jaccardNgram_score	[0, 1]	Mention-Entity
cosine_similarity	[0, 1]	Mention-Entity
p_subj_ne	[0, 1]	Mention-Entity
p_subj_lit_datatype	[0, 1]	Mention-Entity
p_subj_lit_all_datatype	[0, 1]	Mention-Entity
p_subj_lit_row	[0, 1]	Mention-Entity
p_obj_ne	[0, 1]	Mention-Entity
desc	[0, 1]	Mention-Entity
descNgram	[0, 1]	Mention-Entity
cta_t1 to cta_t5	[0, 1]	Mention-Entity
cpa_t1 to cpa_t5	[0, 1]	Mention-Entity

- **length_entity**: this feature quantifies the length of the entity's name in terms of the number of characters;

- **popularity**: the value of the number of site-links associated with entity e. This information comes from Wikidata; This feature serves as an indicator of the entity's popularity and prominence in the digital domain;

- **pos_score**: a positional score computed as follows: $1/i$, where i is the position of the candidate entity e in the LamAPI ranking. This feature captures the importance of the candidate entity's position in the IR system's ranking.

- **es_score**: a score computed internally by Elasticsearch. It comes from LamAPI and can be considered as a pure syntactic match. This feature captures the score provided by the Information Retrieval (IR) system;

- **ed**: is a measure of the similarity between two strings, calculated by determining the minimum number of single-character edits required to transform one into the other. It can be used to evaluate the similarity between the mention of an entity and its name in a knowledge graph. This feature captures the similarity between the mention and name of the candidate;

- **jaccard**: is a measure of the similarity between two strings, calculated by dividing the number of matching tokens in the two strings by the total number of unique tokens. It can be used to evaluate the similarity between the mention of an entity and its name in a knowledge graph. This feature captures the similarity between the tokens in the mention and the tokens in the name of the candidate entity;

- **jaccardNgram**: is a measure of the similarity between two strings, calculated by dividing the number of matching n-grams in the two strings by the total number of unique n-grams. It can be used to evaluate the similarity between the mention of an entity and its name in a knowledge graph; This feature feature captures the similarity between the n-grams in the mention and the n-grams in the name of the candidate entity. In this case, the value of n is equal to 3. This feature captures the similarity between the n-grams in the mention and the n-grams in the name of the candidate entity;

- **cosine_similarity**: this score is computed by measuring the cosine similarity between the vectors of the mention m and the name of the entity e. The vectors are computed using the FastText library[1]. The cosine similarity measures the cosine of the angle between the two vectors and ranges between -1 (opposite directions) to 1 (same direction). A higher cosine similarity indicates a stronger semantic similarity between the mention and the entity name. This feature captures the text similarity between the mention and the name of the candidate entity by representing them as vectors and computing the cosine similarity between these vectors;

- **p_subj_ne**: is a score that reflects the relationship between a current candidate for a Name Entity (NE) cell and other candidate NE cells on the same row in a table. This task has been implemented by retrieving relations among entities from LamAPI Objects endpoint. This feature provides insights into the relation between the current candidate and other candidates in the same row;

- **p_subj_lit_datatype** : is a score that reflects the similarity between the literal values associated with a current candidate for a subject cell and the literal values on the same row of the table. This task has been implemented by retrieving relations among entities and Literals from LamAPI Literals endpoint; This feature captures the similarity between the literal values associated with a current candidate for a subject cell and the literal values on the same row of the table;

[1] https://fasttext.cc/

- **p_subj_lit_all_datatype**: this feature generates a score based on the match between the literal values associated with a given candidate entity and the literal values in the corresponding row, irrespective of data type;

- **p_subj_lit_row**: this feature calculates a score representing the degree of match between the literal values associated with the candidate entity and the entire content of the corresponding row;

- **p_obj_ne** : is a score that reflects the relationship between a current candidate for a Name Entity (NE) cell and other candidate NE cells on the same row in a table, where the current candidate is in a relationship as an object with the other candidate NE cells. This feature captures the relationship between a current candidate for a Name Entity (NE) cell and other candidate NE cells on the same row in a table, where the current candidate is in a relationship as an object with the other candidate NE cells;

- **desc**: is a score that reflects the similarity between the content of a row in a table and the description of a current candidate in a knowledge graph, using Jaccard similarity based on tokens to compare the two strings; The feature feature provides insights into the similarity between the tokens in the text of the row and the description of the current candidate. It reflects the degree of resemblance or shared information between the two texts.

- **descNgram**: is a score that reflects the similarity between the content of a row in a table and the description of a current candidate in a knowledge graph, using Jaccard similarity based on 3-grams (also known as trigrams) to compare the two strings. This feature provides insights into the similarity between the n-grams (trigrams) in the text of the row and the description of the current candidate. It reflects the degree of resemblance or shared information between the two texts at a more granular level than token-level comparison;

- **cta_t1 to cta_t5**: These five distinct features are designed to capture information related to types associated with the candidates. Each feature is computed as follows: based on the estimated types for a column (represented as a dictionary of type frequencies), all the types that belong to the candidate under consideration and are present in the types dictionary are considered. These types are then sorted, with the most frequent type assigned to cta_t1, the second-most frequent to cta_t2, and so on, up to cta_t5. If it's not possible to assign values to all the cta features, the unassigned features will default to a value of 0.

- **cpa_t1 to cpa_t5**: Similar to cta_t1 to cta_t5, these five distinct features are designed to capture information related to predicates associated with the candidates. Each feature is computed as follows: based on the estimated predicates for a column (represented as a dictionary of predicate frequencies), all the predicates that belong to the candidate under consideration and are present in the predicates dictionary are taken into account. These predicates are then sorted, with the most frequent predicate assigned to cpa_t1, the second-most frequent to cpa_t2, and so on, up to cpa_t5. If it's not possible to assign values to all the cpa features, the unassigned features will default to a value of 0.

Table 5.3 provides an example of annotations for the mentions in Table 1.1. For each mention, three candidates are presented, along with their respective ρ and ρ' scores. Analyzing these scores in conjunction with the type frequencies reported in Table 5.2, several observations can be made. In general, candidates that align with the most frequent types in Table 5.2 tend to have increased confidence scores (ρ') compared to the initial confidence scores (ρ). This trend confirms the importance of aligning candidate types with dominant types in improving confidence.

An exception is observed for the mention 'Jurassic World,' where the correct candidate **Q3512046** experiences a decrease in score from 0.996 (ρ) to 0.753 (ρ'). Despite this decrease, the correct candidate still maintains its position as the top-ranked candidate in the ranking. Conversely, for the mention 'Avatar,' the candidate **Q83090**, which does not align with the most frequent types in Table 5.2, experiences a score decrease from 0.866 (ρ) to 0.778 (ρ'). In contrast, **Q1462437** shows an increase in score (ρ') and aligns with the type 'video game,' which has a reported frequency of 0.6. These observations illustrate the interplay between candidate types, confidence scores, and their impact on entity linking decisions.

One final observation pertains to the limited amount of context available, as Table 1.1 contains only a few rows, providing minimal aggregation of results across rows, particularly in terms of type consistency. However, it is noteworthy that despite this limited context, the correct candidate consistently appears in the first position of the ranking for all mentions.

Table 5.2: Type Frequencies in Table 1.1 Column 1

Type	**Frequency**
Q11424 (film)	0.6
Q7889 (video game)	0.6
Q229390 (3D film)	0.6
Q482994 (album)	0.4
Q24856 (film series)	0.2
Q14514600 (group of fictional characters)	0.2
Q14623646 (fictional organization)	0.2
Q111241092 (film reboot)	0.2
Q25110269 (live-action/animated film)	0.2
Q23847174 (religious concept)	0.2
Q261636 (sequel)	0.2
Q196600 (media franchise)	0.2

5.2.2 The Machine Learning Model

Once a set of candidates has been associated with each mention, a ranking algorithm is required to prioritize and select the best options. In the proposed approach, a pretrained feed-forward neural network is employed defined by a set of parameters Θ to perform the ranking task. This

Table 5.3: Entity Linking Confidence Scores for Table 1.1

mention	id	name	description	ρ	ρ'
X-Men	Q106182	x-men	2000 american superhero film directed by bryan singer	1	1
X-Men	Q2006869	x-men	american superhero film series	0.981	0.932
X-Men	Q3108064	x-men	1993 video game	0.036	0.554
Batman Begins	Q166262	batman begins	2005 british-american superhero film directed by christopher nolan	0.995	0.996
Batman Begins	Q2891561	batman begins	video game based on the film of the same name	0.328	0.53
Batman Begins	Q2401367	batman begins	soundtrack album to the batman begins film	0.138	0.269
Superman Returns	Q328695	superman returns	2006 superhero film directed by bryan singer	0.984	1
Superman Returns	Q3977963	superman returns	soundtrack album for the 2006 film of the same name	0.245	0.445
Superman Returns	Q655031	superman returns	video game loosely based on the movie of the same name	0.18	0.218
Avatar	Q24871	avatar	2009 american epic science fiction film directed by james cameron	1	1
Avatar	Q1462437	avatar	video game	0.32	0.81
Avatar	Q83090	avatar	material appearance or incarnation of a deity on earth in hinduism	0.866	0.778
Jurassic World	Q3512046	jurassic world	2015 american science fiction adventure film directed by colin trevorrow	0.996	0.753
Jurassic World	Q21877685	jurassic world	2018 5th jurassic park film directed by juan antonio bayona	0.826	0.22
Jurassic World	Q20647533	jurassic world	film score	0.045	0.024

neural network calculates an accurate and normalized confidence score ρ for each candidate entity. The confidence score represents the probability that an entity is the correct match for the associated mention.

The reported experiments involved a neural model that was trained using the datasets from the SemTab challenge that were introduced in Chapter 4 in Section 4.4. This choice was also dictated by the need to obtain a model capable of good-quality results for tables with different levels of data quality and belonging to different domains.

The information from these tables was used to create a specific training dataset for the binary classification model. In this model, each candidate can be classified as either correct or incorrect for a certain mention. For every mention, a query was made to the KG, and 11 candidates were considered with different similarity scores, ensuring that the correct candidate was always included. Consequently, each training example consists of one positive instance, representing the correct entity to assign for a given mention, and ten negative instances, representing incorrect candidates for the mention. The deliberate decision to include ten negative examples was aimed at providing the model with a comprehensive understanding of what incorrect candidates entail. By exposing the model to a diverse range of incorrect options, it is believed that a better grasp of the variations and patterns that distinguish them from the correct entity would be achieved.

For each training example, a set of features is calculated. The considered features can be classified into three groups: features for the mention, features for candidates, and features that relate mentions to candidates. For mention-related features, only the tokens are considered. This feature represents the number of tokens in the mention and aims to capture the ambiguity associated with it.

For candidate features, the focus is on the popularity of the candidate in the KG reference. This feature serves as an indication of the entity's importance.

Regarding the features that relate the mention to candidates, there are three groups: text-based features, semantic-based features, and context-based features. Text-based features capture the similarity between mentions using different criteria, such as Levenshtein distance and token-based features like Jaccard similarity with n-grams (n=3). Semantic-based features consider the semantic relations among candidates in different cells within the same row, leveraging the semantic information of the table. Context-based features utilize the entire table context and aggregate information such as types, predicates, and additional features related to the last prediction. These features may include the prediction score and the difference between the top two candidates. By incorporating these feature groups, The objective is to comprehensively capture various aspects of the problem, including textual, semantic, and contextual information, to enhance the entity resolution process.

Table 5.4 provides details about the architecture of the neural network employed in this study. It is a plain feed-forward neural network, whose hyper-parameters were determined through preliminary experiments. In recent times, deep networks have demonstrated remarkable potential in handling increasingly difficult and complex tasks, often rivaling or even surpassing human capabilities. These networks are typically built using highly intricate architectures. However, for this particular work, an approach prioritizing simplicity and speed was chosen, while still maintaining strong learning capability and generalization. It is acknowledged that there is potential for enhancing the network's classification capability, devising an architecture optimized for the candidate ranking task is beyond the scope and objectives of this work.

Table 5.4: Model Architecture

Layer (type)	Output Shape	Param #	Connected to
dense	(64,)	1344	(20,)
batch_norm	(64,)	256	(64,)
dense_1	(128,)	8320	(64,)
batch_norm_1	(128,)	512	(128,)
dense_2	(256,)	33024	(128,)
batch_norm_2	(256,)	1024	(256,)
dense_3	(128,)	32896	(256,)
batch_norm_3	(128,)	512	(128,)
dense_4	(64,)	8256	(128,)
batch_norm_4	(64,)	256	(64,)
dense_5	(2,)	130	(64,)

5.2.3 Decision: Uncertainty Estimation and Metrics

Uncertainty estimation in the decision-making process is comprised of two primary steps: firstly, the calculation of an uncertainty measure, denoted as ω, for each candidate to facilitate the ranking of all mentions; secondly, the setting of a threshold parameter σ (defaulted at 0.5) to categorize each mention as either *linked* or *unlinked*. The parameter ω is obtained through a linear combination of a confidence score, represented as ρ, and another parameter δ, defined as the difference between the ρ scores of the top two candidates.

The formula for calculating the set Ω, which consists of the ω values for each *i-th* mention, is expressed in Equation 5.1. Here, M represents the total number of mentions and k is a learnable weight, which can be fine-tuned through Human-in-the-Loop (HITL) support, if necessary.

$$\Omega = \{\omega_i\}_{i=1}^{M}, \quad \omega_i = (1-k)\rho_i + k\delta_i \tag{5.1}$$

After sorting the elements of Ω, a global ranking of candidates corresponding to mentions is obtained. This ranking can then be employed to partition the set of mentions into subsets denoted as *linked* and *unlinked*. Candidates with higher values of ω are presumed to be correct, while those with lower values are deemed uncertain and may require human review for disambiguation.

To evaluate the effectiveness of the algorithm in correctly ranking candidates, standard metrics such as F1 score, Precision, and Recall are employed. These metrics are calculated based on the first candidate in the ranking being considered as the correct candidate. The formulas for calculating the F1 score are consistent with the definitions provided by the Semantic Tabular Data Challenge (SemTab)[2] and are specified in Equations 5.2, 5.3, and 5.4 as follows:

$$Precision = \frac{\text{Number of Correct Annotations}}{\text{Number of Submitted Annotations}} \tag{5.2}$$

$$Recall = \frac{\text{Number of Correct Annotations}}{\text{Number of Gold Standard Annotations}} \tag{5.3}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.4}$$

These metrics provide a structured methodology for assessing algorithmic performance by taking into account the number of correct annotations, submitted annotations, and ground truth annotations present in the dataset.

Table 5.5 provides an example according to what was introduced previously. In detail, it presents ρ and ρ' values provided by the ML models, δ which represents the distance in terms of confidence between the top two candidates, and lastly, ω (with the value of $k = 0.1$) which combines the confidence ρ and δ values in order to define the final ranking function.

The ω scoring function is designed to provide a measure of confidence for the corresponding correct candidate. The motivation for choosing $k = 0.1$ is due to the low level of context available in the considered table, so it is better to give more weight to the confidence (ρ) provided by the ML model (In this case prefer more ρ over δ). The table has only a few rows, which limits the effectiveness of δ as a disambiguation factor, resulting in a significant distance between the confidence scores of the top two candidates.

The values in the table illustrate how the ω scoring function assigns high confidence to the correct candidate in each case, despite the limited context. This demonstrates the effectiveness

[2]https://sem-tab-challenge.github.io/2023/

of the proposed approach in ranking candidates with confidence, even in scenarios with minimal contextual information.

Table 5.5: Example with δ and ω values

mention	id	name	description	ρ	ρ'	δ	ω
X-Men	Q106182	x-men	2000 american superhero film directed by bryan singer	1	1	0.068	0.9068
X-Men	Q2006869	x-men	american superhero film series	0.981	0.932	0.068	0.8456
X-Men	Q3108064	x-men	1993 video game	0.036	0.554	0.068	0.5054
Batman Begins	Q166262	batman begins	2005 british-american superhero film directed by christopher nolan	0.995	0.996	0.466	0.943
Batman Begins	Q2891561	batman begins	video game based on the film of the same name	0.328	0.53	0.466	0.5236
Batman Begins	Q2401367	batman begins	soundtrack album to the batman begins film	0.138	0.269	0.466	0.2887
Superman Returns	Q328695	superman returns	2006 superhero film directed by bryan singer	0.984	1	0.555	0.9555
Superman Returns	Q3977963	superman returns	soundtrack album for the 2006 film of the same name	0.245	0.445	0.555	0.456
Superman Returns	Q655031	superman returns	video game loosely based on the movie of the same name	0.18	0.218	0.555	0.2517
Avatar	Q24871	avatar	2009 american epic science fiction film directed by james cameron	1	1	0.19	0.919
Avatar	Q1462437	avatar	video game	0.32	0.81	0.19	0.748
Avatar	Q83090	avatar	material appearance or incarnation of a deity on earth in hinduism	0.866	0.778	0.19	0.7192
Jurassic World	Q3512046	jurassic world	2015 american science fiction adventure film directed by colin trevorrow	0.996	0.753	0.533	0.731
Jurassic World	Q21877685	jurassic world	2018 5th jurassic park film directed by juan antonio bayona	0.826	0.22	0.533	0.2513
Jurassic World	Q20647533	jurassic world	film score	0.045	0.024	0.533	0.0749

5.3 Implementation and Evaluation of the Algorithms

In this section, the implementation of the proposed approach in a tool is outlined, providing a description of the conducted experiments, and discuss the obtained results.

The proposed approach was implemented in Alligator[3] a tool that is implemented in Python and utilizes MongoDB. It conducts entity linking tasks over tabular data, supported by a machine learning (ML) algorithm.

A public demonstration environment is available for testing purposes and is detailed on a Swagger documentation page (see Figures 5.3 and 5.4). The Alligator's repository is also publicly accessible, making the code available for download and customization as needed.

[3]https://github.com/roby-avo/alligator

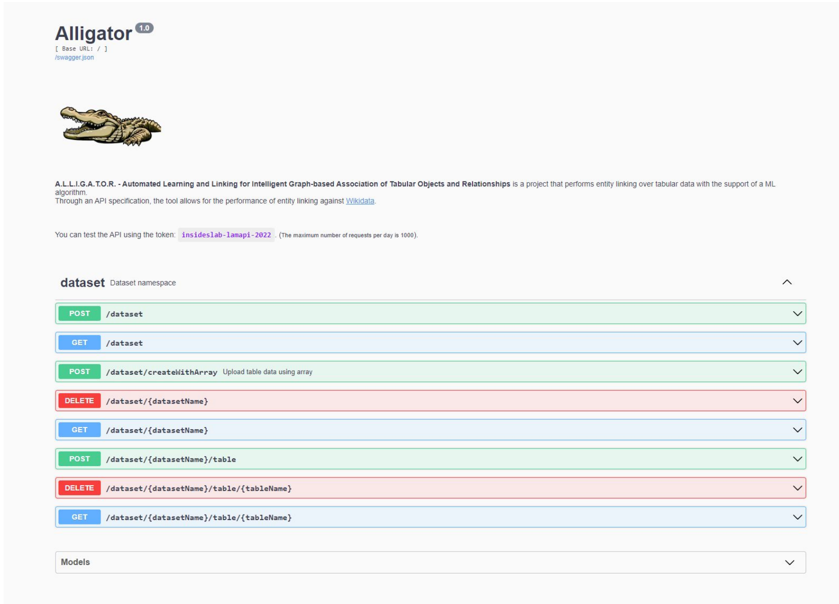


Figure 5.3: Overview of the Alligator Swagger Page

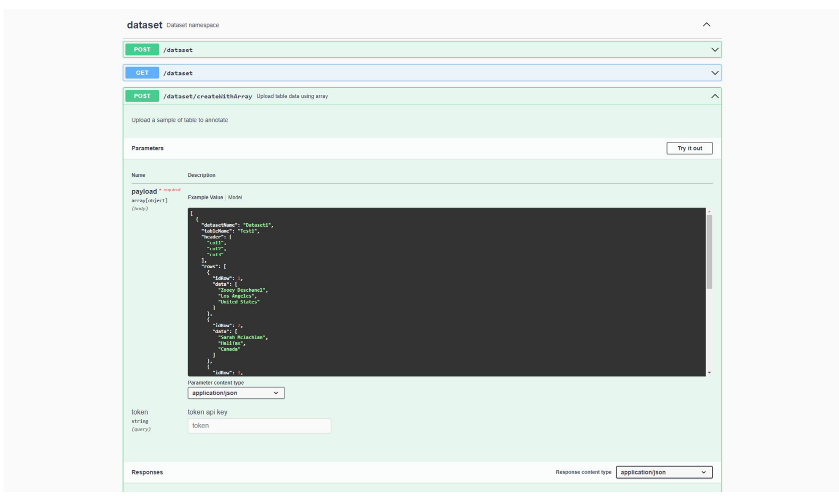


Figure 5.4: Focus on the POST Method for Performing EL on Tabular Data in Alligator

Listing 5.1: Example of input to Alligator API

```
[
    {
        "datasetName": "Dataset1",
        "tableName": "TEST1",
        "header": ["col1","col2","col3"],
        "rows": [
            {"idRow": 1, "data": ["Jurassic World", "Colin Trevorrow",
                                  "12/06/2015", "Universal Pictures",
                                  "124", "1670400637"]},
            {"idRow": 2, "data": ["Superman Returns", "Bryan Singer",
                                  "21/06/2006", "Warner Bros.",
                                  "154", "391081192"]},
            {"idRow": 3, "data": ["Batman Begins", "Christopher Nolan",
                                  "15/06/2005", "Warner Bros.",
```

```
                                        "140", "371853783"]}
          ],
        "semanticAnnotations": { "cea": [], "cpa": [], "cta": []},
        "metadata": {
            "column": [
                {"idColumn": 0, "tag": "NE"},
                {"idColumn": 1, "tag": "NE"},
                {"idColumn": 2,"tag": "LIT", "datatype": "NUMBER"},
                {"idColumn": 3,"tag": "NE"}
                {"idColumn": 4,"tag": "LIT", "datatype": "NUMBER"},
                {"idColumn": 5,"tag": "LIT", "datatype": "NUMBER"}
            ]
         },
       "kgReference": "wikidata"
    },
    {
       "datasetName": "Dataset1",
       "tableName": "TEST2",
       "rows": [...],
        ...
       "kgReference": "wikidata"
    }
]
```

Listing 5.2: Example of output to Alligator API

```
[
{
  "datasetName": "Dataset1",
  "tableName": "TEST1",
  "rows": [...],
  "semanticAnnotations": {
    "cea": [
      {
        "idColumn": 0,
        "idRow": 1,
        "entity": [
          {
            "id": "Q3512046",
            "name": "jurassic world",
            "type": [ {"id": "Q229390","name": "3D film"},
                      {"id": "Q11424","name": "film"},
                      {"id": "Q261636", "name": "sequel"}],
            "description": "2015 american science fiction adventure
                                film directed by colin trevorrow",
            "match": true,
            "score": 1.0
          }
        ]
      }
    ],
    "cpa": [{"idSourceColumn": 0, "idTargetColumn": 0,"predicate": "P138"},
            {"idSourceColumn": 0, "idTargetColumn": 1, "predicate": "P57"},
            ...
    ],
    "cta": [{"idColumn": 0, "types": ["Q229390"]},
            {"idColumn": 1, "types": ["Q5"]}
    ],
    "metadata": {
            "column": [ {"idColumn": 0, "tag": "NE"},
                        {"idColumn": 1, "tag": "NE"},
                        {"idColumn": 2,"tag": "LIT", "datatype": "NUMBER"}
            ]
    },
    "kgReference": "wikidata"
    "status": "DONE"
},
...
]
```

5.3.1 The Experimental Campaign

The experiments are designed following the *K-fold dataset validation* process, with $K = 6$ [59]. In each experiment, a dataset is alternately used as the test set, while the remaining datasets are utilized for training the neural network and learning the parameter k to compute Ω.

Table 5.6 presents the results of the six experiments, showcasing the progressive increase of the F1 score throughout the linking process (columns 1 to 3). It is worth noting that the reported figures represent the results of automatic annotation prior to any decision or human revision while the last column provides the highest-scoring entry for each table from the SemTab challenge [39, 36, 16], serving as a benchmark for comparison.

The evidence is that F1 increases at every step for every experiment reaching for the full approach (third column in boldface) results in line with the state-of-the-art.

In particular, **PN ranking** refers to the first prediction made by the model without considering any contextual information such as types and predicates. On the other hand, **PN + RN ranking with types** corresponds to the second prediction made by the model, where information about types (mostly) and predicates is taken into account.

Regarding **candidate retrieval with indexing**, this represents the ranking generated by the Elasticsearch-based indexing technique using a scoring function. It's important to note that this ranking only relies on textual matches.

A more detailed analysis reveals a significant increase in F1 scores in the first two steps (columns) for the 2T-2020 and HardTableR3 datasets. This improvement can be attributed to the deliberate inclusion of numerous typos in the 2T-2020 dataset and the high ambiguity of mentions in the HardTableR3 dataset. These deliberate challenges were introduced to test the robustness of the annotation tools. Consequently, the Elasticsearch-based indexing technique struggles to address the complexity of these specific test cases, while a more sophisticated machine learning model performs significantly better.

Two critical datasets, namely *Round3* and *HardTableR3*, were specifically designed to present challenges. The former contains a large number of person names written in an abbreviated form, such as 'JFK' instead of 'John Fitzgerald Kennedy', as well as nicknames like 'Doctor J' instead of 'Julius Irving'. Differently, the latter dataset consists mostly of tables with a single entity column and several numeric columns, lacking the contextual information that could support the linking process. As the neural network has primarily been trained on the remaining, more regular datasets, the obtained results are not comparable with the SemTab Top Scorer. Nevertheless, it should be noted that these cases are outliers and significantly differ from real-world datasets, limiting their practical applicability. In particular, it can observed for *Round3* the best result (**PN + RN ranking with types**) is still a bit far away from the **SemTab Top Scorer** because specific rules to address the abbreviation in person name are needed. While, regarding *HardTableR3* is even reached out the **SemTab Top Scorer** but it is possible to notice, how the result **Retrieval with indexing** is very low, which means how much ambiguous are the mentions inside that dataset.

In Table 5.6, an interesting observation concerning the *2T-2020* dataset is the decline in performance when types are considered, evident from the difference between **PN Ranking** and **PN + RN Ranking with Types**. Upon investigation, it was found that the Neural Ranker struggles to correctly annotate certain mentions such as 'Halifax' (Column 1) and 'Canada' (Column 2), as illustrated in Table 5.7. Given that *2T-2020* contains several tables with numerous rows, these inaccuracies contribute significantly to the final F1 score. A potential solution for this issue is

Table 5.6: F1 for each Step in the Linking Workflow

Test Dataset	**Retrieval with indexing**	**PN ranking**	**PN + RN ranking with types**	**SemTab Top Scorer**
	F1	**F1**	**F1**	**F1**
Round_T2D	0.72	0.85	0.89	0.90
Round3	0.60	0.80	0.81	0.97
Round4	0.70	0.93	0.94	0.99
2T-2020	0.33	**0.92**	0.88	0.90
HardTableR2	0.68	0.94	0.97	0.98
HardTableR3	0.26	0.95	**0.97**	0.97

discussed in Section 6.1.

Table 5.7: Example Table from *2T-2020* Dataset Illustrating Name Variations

col0	**col1**	**col2**
Zooey Deschanel	Los Angeles	United States
Zooey Dechanel	Los Angeles	United States
Zooey Deschannel	Los Angeles	United States
Sarah Mclaughlinn	Halifax	Canada
Sarah Mclouglin	Halifax	Canada
Sarah Maclean	Halifax	Canada
Alanis Maurissette	Ottawa	Canada
Alanis Morrisetti	Ottawa	Canada
Alanis Morisa	Ottawa	Canada

5.3.2 Examining Variations in Scores

In this section, the nuances of how scores change with varying values of k are explored. Specifically, the focus rests on the extreme values of k, namely when $k = 0$ and $k = 1$. In addition, the analysis includes evaluating the scores using the learned values of k for each dataset, as highlighted in the Appendix A.

The primary objective is to discern trends associated with both 'WRONG' and 'CORRECT' cases as the value of k increases. An expected outcome is a decrease in scores for the 'WRONG' cases. Simultaneously, it is imperative to maintain high scores for the 'CORRECT' cases, aiming for minimal reduction in their scores. With this objective in focus, further investigation into these score variations across different datasets is warranted, taking into account the various k values outlined earlier.

To avoid overburdening this section, results for only the *HardTableR2* dataset are presented here. Comprehensive results across all datasets can be found in Appendix A.2.

In the *HardTableR2* dataset, represented in Figure5.5, there's a clear downward trend in scores for the 'WRONG' cases at $k = 0.4$. At the extreme of $k = 1$, while many 'WRONG' cases descend in scores, a significant number of 'CORRECT' cases remain in their original scoring bands.

In conclusion, after examining the score distributions for the 'WRONG' and 'CORRECT' cases across all datasets, the initial expectations have been largely confirmed. As the value of k increases, the 'WRONG' cases consistently gravitate towards lower score values. However, vigilance must be maintained concerning the 'CORRECT' cases to prevent excessive shifting toward these lower scores. Therefore, a value of $k = 1$ is not recommended, as it results in an undesirable migration of too many 'CORRECT' cases to lower score levels.

For the most part, this observation holds true across all datasets. An exception, however, is observed with the *2T-2020* dataset. Here, an unusually large number of 'CORRECT' cases shift to lower scores. As discussed in subsection 5.3.1, this anomaly can be attributed to the inherent noise in the *2T-2020* dataset.

5.3.3 Further Discussion and Analyses of Score Distributions

In this context, additional analyses are conducted, leveraging the rankings generated by the various models mentioned in Section 5.3.1.

In detail, an analysis was conducted for each dataset to examine the distribution of scores (ρ) for the top one candidate and the difference in terms of score between the top one candidate and the second candidate in the ranking (δ). Also here, like in the previous Section 5.3.2, in order to avoid overburdening this section, results for only the *HardTableR2* dataset are presented here. Comprehensive results across all datasets can be found in Appendix A.4.

Figure 5.6 illustrates the results obtained for the HardTableR2 dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.8 to 1. Additionally, the values of δ are primarily concentrated within the 0.8 to 1 range, indicating effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.8 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.3 range.

By examining the bar plots for each dataset, A potential correlation between the distributions of scores and delta values for both 'CORRECT' and 'WRONG' cases is observed. For 'CORRECT' cases, higher scores are typically associated with candidates that exhibit a significant difference in delta, indicating a successful disambiguation. Conversely, for 'WRONG' cases, higher scores tend to be linked to candidates with a smaller difference in delta, suggesting a more challenging disambiguation scenario.

To illustrate these observations, thresholds for the ρ and $delta$ values are established for each dataset. A reasonable choice is a ρ threshold of 0.7 and a δ threshold of 0.5. Consequently, the following rules are defined:

- **Rule1 for 'CORRECT' cases:** If $\rho > 0.7$ and $\delta > 0.1$;
- **Rule2 for 'WRONG' cases:** If $\rho <= 0.7$ and $\delta <= 0.1$.

In accordance with the pre-established criteria, two metrics, denoted as **#CORRECT** and **#WRONG**, are calculated using the ground truth of each dataset as a reference. The variable **NO RULE** designates instances that neither comply with Rule 1 nor Rule 2 and have not been compared against the ground truth, representing misclassified examples.

The metric **#CORRECT**is formally calculated as follows:

$$\frac{\text{\#CORRECT}}{\text{\#annotations conforming to Rule 1}}$$

Similarly, **#WRONG** is defined as:

$$\frac{\text{\#WRONG}}{\text{\#annotations conforming to Rule 2}}$$

By these definitions, **#MISCLASSIFIED** refers to instances that adhere to either Rule 1 or Rule 2 but are inconsistent with the ground truth. Consequently, the sum of **#CORRECT** and **#MISCLASSIFIED** is equal to 1.

In the case of **NO RULE**, only **#MISCLASSIFIED** exists, indicating that the annotation does not comply with either Rule 1 or Rule 2. The computation for **#MISCLASSIFIED** under **NO RULE** is formulated as:

$$\frac{\text{\#ann not in Rule 1 or 2}}{\text{\#Total ann}}$$

where $\text{\#Total ann} = \text{\#ann not in Rule 1 or 2} + \text{\#ann in Rule 1} + \text{\#ann in Rule 2}$.

The results are reported in Table 5.8.

Table 5.8: Statistic analysis considering thresholds reported above for ρ and δ over the different datasets

Dataset	**Rule1**		**Rule2**		**NO RULE**
	#CORRECT	**#MISSCLASSIFIED**	**#WRONG**	**#MISSCLASSIFIED**	**#MISSCLASSIFIED**
Round1_T2D	91.62%	8.38%	66.21%	33.79%	12.43%
Round3	88.34%	11.66%	74.61%	25.39%	18.77%
Round4	96.74%	3.26%	66.72%	33.28%	16.55%
2T-2020	95.55%	4.45%	83.81%	16.19%	43.71%
HardTableR2-2021	98.07%	1.93%	75.03%	24.97%	10.47%
HardTableR3-2021	72.12%	27.88%	87.10%	12.90%	19.74%

Furthermore, for each dataset utilized during the experiment, a version with NIL injections was created. This involved removing approximately 10% of the correct candidates from the ground truth and converting them into NIL annotations. This was achieved by eliminating the correct candidates during the candidate retrieval phase. To determine which candidates to remove, the entities within each dataset were ordered based on their frequency of appearance, from the most popular to the least. Subsequently, the bottom 10% of these entities were selected for removal and treated as NIL entities. Upon the introduction of NIL cases, the expectation was that the level of uncertainty within the dataset would increase. Specifically, in the context of Rule 2, an increase in the **#WRONG** percentage was anticipated, aligning with the notion of heightened uncertainty. Concurrently, for instances categorized under **NO RULE**, the **#MISCLASSIFIED** percentage was expected to rise. In contrast, for Rule 1, a decrease in the **#CORRECT** percentage was projected (Table 5.9).

Table 5.9: Statistical analysis considering thresholds reported above for ρ and δ over different datasets with NIL injections

Dataset	Rule1		Rule2		NO RULE
	#CORRECT	#MISCLASSIFIED	#WRONG	#MISCLASSIFIED	#MISCLASSIFIED
Round1_T2D	87.47%	12.53%	83.30%	16.70%	14.92%
Round3	86.13%	13.87%	77.94%	22.06%	20.55%
Round4	95.14%	4.86%	78.74%	21.26%	19.26%
2T-2020	93.76%	6.24%	89.55%	10.45%	43.63%
HardTableR2	95.16%	4.84%	90.91%	9.09%	14.05%
HardTableR3	64.90%	35.10%	89.69%	10.31%	22.27%

In conclusion, Table 5.9 illustrates how the introduction of NIL entities resulted in increased uncertainty, aligning with the initial expectations. The analysis confirmed that the proposed approach is sensitive to NIL cases.

5.3.4 Benchmarking Tools Across Tables

In this section, a benchmark comparison of available online solutions is conducted. The objective was to evaluate the performance of these tools and subsequently discuss their respective strengths and weaknesses.

In Figure 5.7, the benchmarking results are presented, comparing several tools that were evaluated. Specifically, consideration was given to ChatGPT, notable for its significant popularity. Additionally, an examination was conducted on Mtab, a solution that emerged from the SemTab challenge. OpenRefine, a renowned tool that handles various operations on structured data, including tabular data, was also explored. Finally, an evaluation was performed on Alligator, the approach proposed in this thesis.

As illustrated in Figure 5.7, the performance of the proposed solution, Alligator, is commendable. In particular, Alligator exhibits robust performance across various metrics. In the following sections, a comparative analysis of the various approaches will be conducted, with a focus on highlighting the strengths and weaknesses of each.

- **chatGPT3.5** The results from chatGPT3.5 are impressive. However, the confidence scores assigned by the model appear arbitrary and it's susceptible to hallucination;

- **MTab** exhibits strong performance, even with noisy data. Its main drawbacks are the occasional misidentification of column data types and the absence of confidence scores for its annotations;

- **OpenRefine** is a versatile tool offering a plethora of operations. Nonetheless, its efficiency wanes when dealing with unclean data;
- **Alligator** provides commendable performance across the board. It manages dirty data effectively and additionally offers confidence scores for annotations.

Table 5.10 pertains to movies. It includes some misspelled text, and notably, the 'Main Actor' column often lists two names. Regarding this specific column, ChaGPT successfully identified both names, whereas other approaches typically recognized only one. There are also disparities in the results generated by different algorithms. For instance, in the second row of the 'Main Actor' column, one algorithm identified 'Tim Robbins' while another detected 'Morgan Freeman'.

Table 5.10: Table movies with dirty data

Film Title	**Year**	**Director**	**Main Actor/Actress**	**Description**
Forset Gump	1994	Roberd Zemeckis	Tom Hanks	A heartfelt stroy abuot a man with low IQ navigating thourgh lige
The Shawshenk Redemtion	1994	Franik Darabont	Tim RObbins, Morgna Freeman	Two inmets befriending each other in prison, with a stunning twist at the end
Pulp Fictino	1994	Quetin Taratino	John Travota, Samule L. JAckson	A non-chnorologically ordened movide with volence and drma
The Dark Knght	2008	Christother Nolan	Christian Bale, Heath Legder	A fantasic acion-movide with a blakc-clad hero and iconic villian
Fightr Club	1999	Davied Fincher	Brad Pitt, Ed Nortan	An existential and pscgholigcal thriller about a secret underground fighting club

The table 'movies with clean data' refers to Table 1.1, and that table is straightforward and provides ample information, aiding algorithms in their disambiguation efforts. Consequently, algorithms generally perform exceptionally well on this table.

Table 5.11 presents data related to public bodies. This table comprises authentic data, and the aim is to evaluate the proficiency of the algorithms in handling real-world data.

Table 5.12 pertains to places. Considering the limited information available within the table, it is anticipated that the algorithms may face challenges in handling it.

Table 5.11: Table with SN data

buyer	aug_buyer_name	url	postal_town	administrative_area_level_2	administrative_area_level_1	country
Allerdale Borough Council	Allerdale Borough Council	http://www.allerdale.gov.uk/	Workington	Cumbria	England	United Kingdom
Be First (Regeneration) Limited	Be First	http://www.befirst.london/	Barking	Greater London	England	United Kingdom
Cleeve School & Sixth Form Centre of Excellence	Cleeve School and Sixth Form Centre of Excellence	http://www.cleeveschool.net/	Cheltenham	Gloucestershire	England	United Kingdom
CPD - Construction Procurement Delivery (CPD)	Construction	https://www.finance-ni.gov.uk/construction-procurement-delivery	Belfast	Belfast	Northern Ireland	United Kingdom
Defra Network & eTendering Portal	Nobel House		London	Greater London	England	United Kingdom
Driver and Vehicle Licensing Agency	DVLA	https://www.gov.uk/government/organisations/driver-and-vehicle-licensing-agency	Swansea	Swansea	Wales	United Kingdom

Table 5.12: Table two columns with ambiguous data

col0	col1
Park Palace	81
Park Avenue	100
Millennium Tower	103
The Parade	110
Bahia Center Tower C	111
Rita Tower	117
Standard Bank Building	139
El Gezira Tower Movenpick Hotel	142

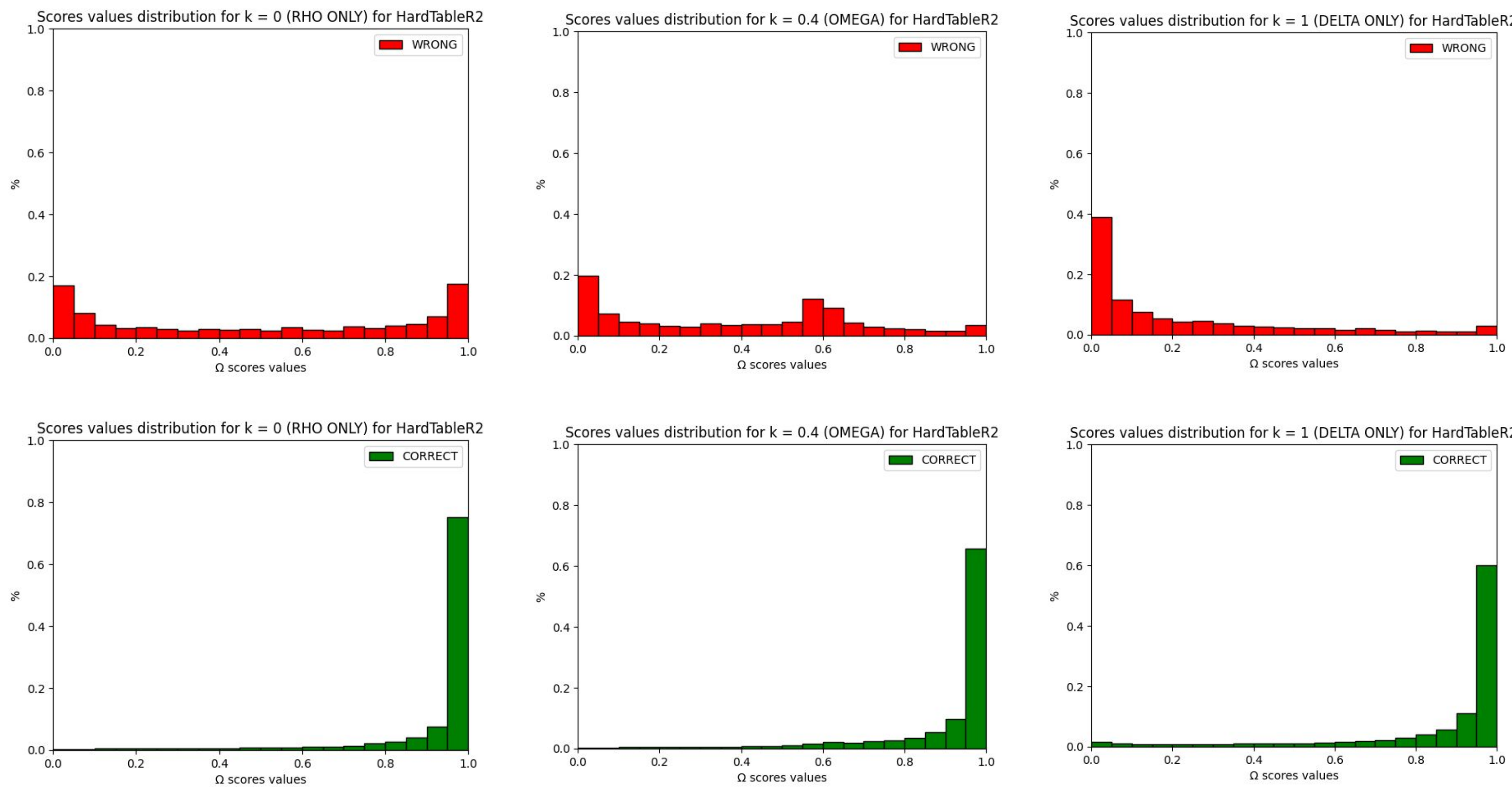


Figure 5.5: Scores variations for *HardTableR2* dataset

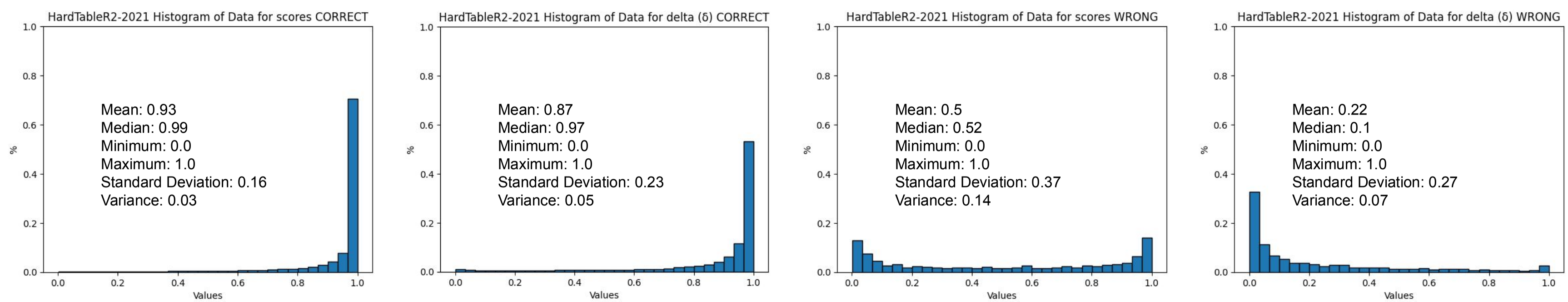


Figure 5.6: Distribution of ρ and δ over *HardTableR2*

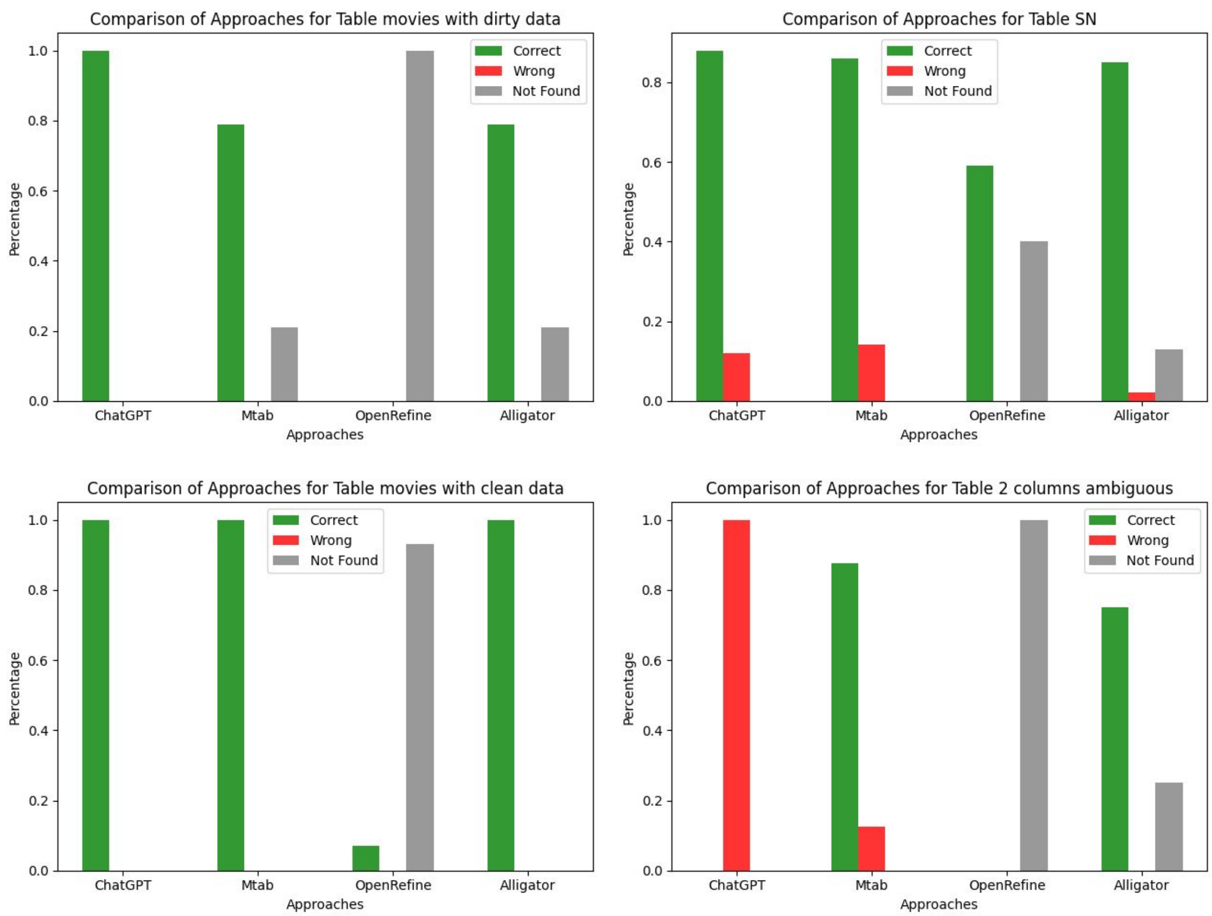


Figure 5.7: Benchmark comparison of various solutions across four distinct tables

6. Human In The Loop (HITL) over Tabular Data

Automated linking processes often face challenges in generating precise results, especially when operating on diverse or complex datasets. While computational models can substantially automate these tasks, human expertise remains indispensable for resolving ambiguities and refining the quality of the outputs.

In our experimental setups, two types of human intervention have been considered. Initially, an oracle has been employed for model definition and fine-tuning, ensuring that the linking process meets a predefined standard of accuracy. However, to deploy the model in real-world scenarios, real user involvement is essential.

To facilitate this human-in-the-loop (HITL) paradigm, tools must be developed to streamline the user's task. These tools can range from specialized user interfaces (UI) or customized notebooks, aimed at enabling efficient and effective human intervention.

The incorporation of human feedback can lead to various types of refinement in the linking process:

1. **Human's rules:** Iterative updates based on human inputs can be stored for future use, thereby improving the system's adaptability.

2. **Decision Rule Revision:** Adjustments can be made to the decision-making rules, such as modifying the threshold values or refining type-specific rules.

3. **Model Revision:** The parameters of the underlying model can be updated to better tune its performance for specific tasks.

The remainder of this chapter will explore these aspects in greater detail, discussing frameworks, workflows, and case studies relevant to the HITL approach over tabular data. A preliminary work was done in [3].

6.1 Interactive Human Revision

This section explores the integration of the Alligator algorithm within interactive environments to elevate the data enrichment process. It highlights the value of interactive data analysis and algorithmic fine-tuning, emphasizing the advantages offered by the integration of Alligator into both Jupyter Notebooks and a dedicated user interface.

The utilization of Jupyter Notebooks provides data scientists and researchers with a dynamic means to interact with the Alligator algorithm, enabling real-time adjustments and immediate visual feedback. A compelling case study in entity reconciliation serves as an illustrative example of the power of this integration.

Additionally, the section delves into the user interface, known as semTUI (Semantic Enrichment of Tabular Data User Interface), designed to make the semantic enrichment process accessible to both experts and non-experts. It addresses the crucial role of human intervention in refining automated results, and semTUI bridges this gap effectively by allowing users to review, interpret, and improve algorithmic outputs.

Explore the subsections below to understand the benefits of these interactive environments and how they empower users in the data enrichment journey.

6.1.1 User Review through a Jupyter Notebook

The integration of the Alligator algorithm in a Jupyter Notebook environment offers significant advantages in terms of interactive data analysis and algorithmic fine-tuning. This subsection delineates the relevance of such integration by presenting a case study focused on entity reconciliation in Table 1.1.

Jupyter Notebooks facilitate a more dynamic interaction with the algorithm, allowing for real-time adjustments and immediate visual feedback. This interaction model is particularly beneficial for data scientists and researchers who require a more hands-on approach to understanding the behavior and output of Alligator.

Figures 6.1 and 6.2 illustrate the outcome of applying the Alligator algorithm for entity reconciliation on the aforementioned table. In Figure 6.1, the reconciled table displays solely the Wikidata QIDs, presenting a streamlined overview. Conversely, Figure 6.2 offers an enriched output by including additional metadata associated with each QID, namely the entity name, a short description, and the confidence score generated by Alligator.

This modality of interaction allows users to customize their view based on specific requirements, thereby enhancing the usability and interpretability of the algorithm's output. The incorporation of additional metadata, as seen in Figure 6.2, provides a more comprehensive context, which can be instrumental in nuanced data analysis tasks.

6.1.2 User Review through a User Interface

So here it described how it is possible to call Alligator from an UI that is called semTUI[1] which is a fully modular framework for the Semantic Enrichment of Tabular Data, adoptable by both experts and non-experts in the context of semantics. Nowadays, the enrichment task is at the core of almost every data analytics pipeline, and at the same time, it proves to be also costly in both time and money. For this reason, it is important to provide data scientists with tools that guide them through the steps of the enrichment process, supporting them with interactive choices and visualizations. Semantics can bridge the gap in finding links across datasets and find solutions for the extension step, but it is important to include users in the annotation process. Indeed, the extension step is strongly related to the semantic annotation, but automatic algorithms that provide it can fail. However, if results are made available, interpretable, and editable, they can be reviewed and improved by the human's knowledge. A study and overview of state-of-the-art tools for both the Semantic Interpretation and Enrichment task have highlighted some of their

[1] https://i2tunimib.github.io/I2T-docs/

problems. They are limited by not supporting humans in the loop of the annotation process, while also not fully providing extension steps necessary to complete the enrichment of tabular data. Most of the related works also present an entry barrier for less experienced users.

So, in Figure 6.3 is possible to see the splash screen of semTUI with a list of tables inside a dataset.

In Figure 6.4, it is possible to see the outcome result came from Alligator that has taken as input that table showed.

	Title	Title (Wikidata ID)	Director	Director (Wikidata ID)	Release date	Distributor	Distributor (Wikidata ID)	Length in min	Worldwide gross (USD)
0	X-Men	Q106182	Bryan Singer	Q220751	14/07/2000	Twentieth Century Fox	Q434841	104	296300000
1	Batman Begins	Q166262	Christopher Nolan	Q25191	17/06/2005	Warner Bros.	Q126399	140	371853783
2	Superman Returns	Q328695	Bryan Singer	Q220751	28/06/2006	Warner Bros.	Q126399	154	391081192
3	Avatar	Q24871	James Cameron	Q42574	15/01/2009	Twentieth Century Fox	Q434841	162	2744336793
4	Jurassic World	Q3512046	Colin Trevorrow	Q5145625	11/06/2015	Universal Pictures	Q168383	124	1670400637

Figure 6.1: Reconciled table displaying only Wikidata QIDs.

	Title	Title (Wikidata ID)	name	description	confidence score	Director	Director (Wikidata ID)	name	description	confidence score	Release date	Distributor	Distributor (Wikidata ID)	name	description	confidence score	Length in min	Worldwide gross (USD)
0	X-Men	Q106182	x-men	2000 film directed by bryan singer	0.658	Bryan Singer	Q220751	bryan singer	american film director, writer and producer	0.995	14/07/2000	Twentieth Century Fox	Q434841	twentieth century fox	american film studio; subsidiary of walt disney studios	0.668	104	296300000
1	Batman Begins	Q166262	batman begins	2005 film by christopher nolan	0.671	Christopher Nolan	Q25191	christopher nolan	british–american filmmaker	0.407	17/06/2005	Warner Bros.	Q126399	warner bros.	american entertainment company	0.933	140	371853783
2	Superman Returns	Q328695	superman returns	2006 film directed by bryan singer	0.437	Bryan Singer	Q220751	bryan singer	american film director, writer and producer	0.993	28/06/2006	Warner Bros.	Q126399	warner bros.	american entertainment company	0.936	154	391081192
3	Avatar	Q24871	avatar	2009 film by james cameron	0.641	James Cameron	Q42574	james cameron	canadian filmmaker	0.419	15/01/2009	Twentieth Century Fox	Q434841	twentieth century fox	american film studio; subsidiary of walt disney studios	0.668	162	2744336793
4	Jurassic World	Q3512046	jurassic world	2015 film directed by colin trevorrow	0.850	Colin Trevorrow	Q5145625	colin trevorrow	american film director	0.995	11/06/2015	Universal Pictures	Q168383	universal pictures	american film studio	0.657	124	1670400637

Figure 6.2: Reconciled table augmented with additional metadata: name, description, and confidence score.

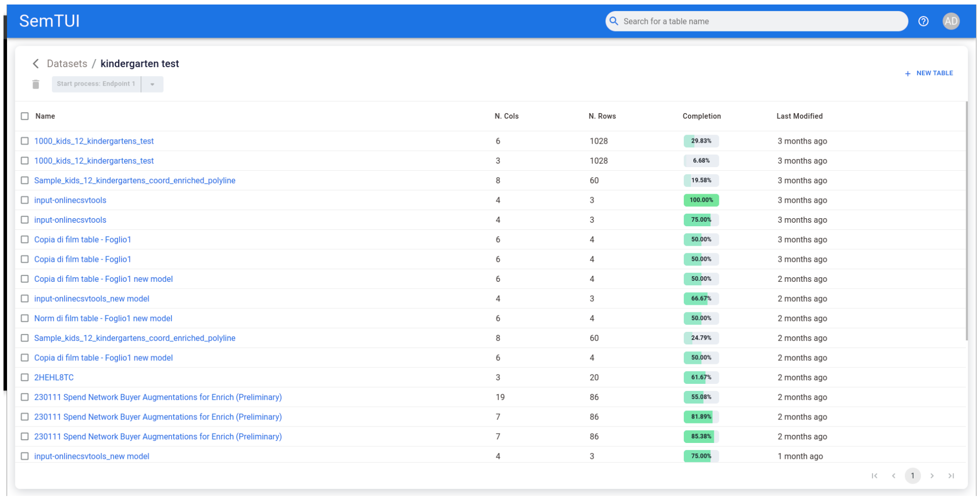


Figure 6.3: Display list of the tables within a dataset. For each table, it is possible to see some statistics like the number of columns and rows.

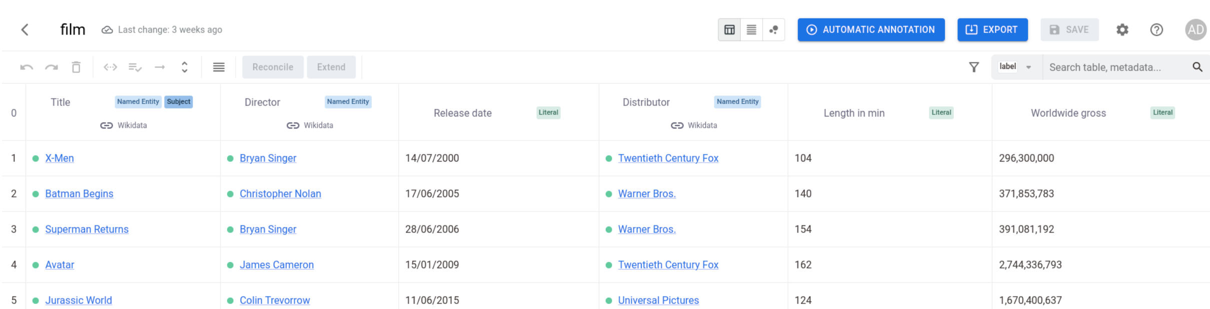


Figure 6.4: Display page with content of a table.

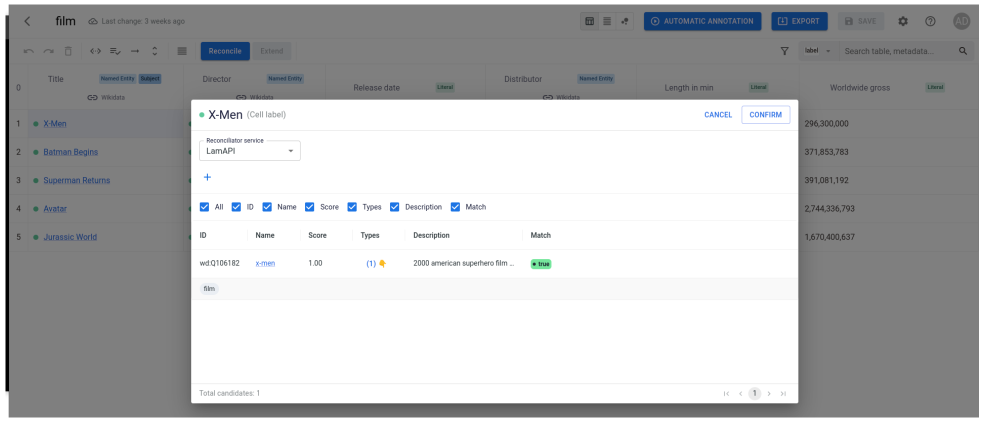


Figure 6.5: Detailed page for the analysis result of reconciling a mention.

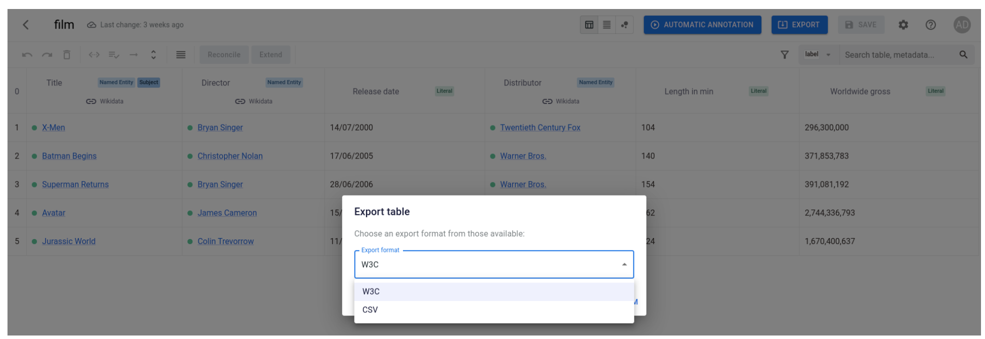


Figure 6.6: Page for downloading annotations in different formats.

semTUI is an open-source tool that provides ways to both annotate a raw table and easily view data of semantically annotated tables, developed using React and Typescript[2]

6.2 Types of Human Feedback

This section elucidates the various types of human feedback and furnishes examples for each. As introduced in the opening of this chapter, it is possible to consider three various types of refinement in the linking process: **Human's rules**, **Decision Rule Revision**, and **Model Revision**. These types of feedback are designed specifically to enhance performance in distinct domains.

6.2.1 Human's Rules

Human's Rules pertain to the dynamic extension of a knowledge base based on human-generated feedback. Consider, for example, Table 1.1, where human feedback specifies that the entry 'Bryan Singer' in the second column refers to a director born in the United States, known for his work on 'X-men' and 'Jurassic World' and associated with the Wikidata ID **Q220751**. This feedback allows a formulation of an association, or rule, denoted as 'Bryan Singer' → **Q220751**. Subsequently, the system can unambiguously identify the corresponding entity for any mention of 'Bryan Singer' as it now contains this rule in its Human's Rules.

The rule's applicability is not limited to a particular domain but is also linked to specific columns in a table, contingent upon their header labels. For example, Table 5.11 includes columns labeled *postal_town* and *administrative_area_level_2*. The term 'Belfast' appears in both columns but holds divergent semantic meanings either as a town or an administrative area. Consequently, two distinct Wikidata entities are involved: **Q10686**[3] and **Q1140130**[4].

As another illustration, consider a dataset focusing on European cities that frequently mentions the entity 'Paris.' In broader contexts, the term 'Paris' could correspond to various candidates such as 'Paris, France' or 'Paris, Texas.' A user, after scrutinizing the dataset, decides that within this European-centric dataset, 'Paris' should exclusively map to 'Paris, France.' This decision culminates in the creation of a domain-specific rule, denoted as 'Paris' → **Q90**. This rule is subsequently incorporated into the system's Human's Rules, ensuring that future mentions of 'Paris' in similar European contexts are automatically mapped to 'Paris, France.'

In summary, Human's Rules capitalize on the likelihood of repeated values within columns. This feature is particularly advantageous when dealing with extensive datasets, as it facilitates accurate entity linking via human-validated rules.

[2] https://github.com/I2Tunimib
[3] https://www.wikidata.org/wiki/Q10686
[4] https://www.wikidata.org/wiki/Q1140130

6.2.2 Decision Rule Revision

This subsection delineates two primary strategies for revising decision rules: *(i)* adjusting a threshold applied to confidence scores, and *(ii)* manipulating the most probable column types to filter candidates.

For the first strategy, *(i)*, Figures 6.7, 6.8, and 6.9 demonstrate a sequence of steps to identify entities correctly. Specifically, using 'Birmingham City Council' as an example, altering the threshold for confidence scores can expand the set of entities considered accurate. The optimal threshold is typically determined by human expertise, which reviews annotations to provide a score that maximizes the identification of accurate entities, even when they possess lower confidence scores. The selection of this threshold may vary across different domains and is highly dependent on the characteristics of the data.

Regarding the second strategy, *(ii)*, a parallel can be drawn to the example described in Section 6.2.1 concerning 'Belfast'. If it is predetermined (like for instance a human can provide it) that the column type pertains to cities or towns, one can apply a filtering mechanism to the candidates, thus favoring the entity 'Belfast' (city) over 'Belfast' (area).

6.2.3 Model Revision

Model Revision pertains to the direct modification and refinement of the underlying model in response to feedback targeting domain-specific performance. This feedback mechanism is vital when dealing with specific domains, where a generalized model might not offer optimal results. By revising the model according to the nuances of the domain, it ensures that the knowledge learned is both relevant and precise. So, the objective is to specialize the model to specific data, which means overfitting the data at the end.

However, the act of revising a model comes with inherent challenges. Two of the most salient issues are 'fine-tuning of the model' and 'catastrophic forgetting'.

Those two challenges are related to each others, the problem of fine-tunig the model is how can we effectively tackling the problem of re-train the model on the specific data but at the same time how to avoid the model loses too much generality then it means the model will under perform even on the specific data where it was trained.

6.3 Feedback from Human Revision

An *Oracle* was employed to evaluate the candidate links, following the order proposed by the uncertainty estimation module, as outlined in the ordered set Ω. This procedure is performed for each test dataset, based on Formula 5.1. The weight value k, is learned on the training set (fold) aiming to optimize the F1 score, while simultaneously minimizing effort by reducing the percentage of candidates requiring review.

To assess performance, the AUC (Area Under the Curve) metric was utilized, which offers a comprehensive measure of the model's predictive quality, irrespective of the chosen classification threshold. Fig. 6.10 shows the values of F1 calculated for different percentages of links to be reviewed and different values of k. The embedded table reports the performance measures AUC.

The figure refers to the experiment with the fold that excludes the *HartTable-R2* dataset.

The evidence suggests that reviewing at most 20% of the training set to reach 0.98 for F1 and that almost any value of k produces similar results. The best value for k is 0.9 with AUC=0.9840.

The learned value of k is finally applied to the test dataset to compute the F1 score and confirm the effectiveness of the method. The result for the *HartTable-R2* test dataset is reported in Fig. 6.11 where the results obtained with $k = 0.0$ (i.e., considering only the ρ scores given by the model), $k = 1.0$ (i.e., considering only the δ values), and a random selection of candidates for review are also displayed.

The results provide evidence that the learned value of k demonstrates good performance on the test dataset, achieving a remarkable AUC value of 0.984 and an F1 score above 0.98 after examining 20% of the mentions.

Table 6.1 presents the results obtained from the experiments conducted on all datasets. The outcomes are consistent with the aforementioned discussion. Specifically, it is evident that in the case of outlier datasets, such as *Round3*, even with less than 20% of reviews, the F1 score surpasses 0.90, whereas the performance of the highest-scoring participant in the Challenge (refer to Table 5.6) is achieved with 40% of reviews. Moreover, for datasets with fewer typos, the threshold of $F1 > 0.95$ is attained much earlier. As an illustration, the maximum F1 score of 0.99 is accomplished after reviewing only 10% of the uncertain cases for the *HardTableR2* dataset.

Table 6.1: F1 with HITL Incremental Percentage of Reviews

Test Dataset	**k**	**10%**	**20%**	**30%**	**40%**	**50%**
		F1	**F1**	**F1**	**F1**	**F1**
Round_T2D	0.4	**0.94**	0.96	0.98	0.98	1.0
Round3	0.5	0.88	0.92	0.95	**0.97**	0.98
Round4	0.1	0.96	0.97	**0.99**	1.0	1.0
2T-2020	0.9	**0.94**	0.98	0.99	1.0	1.0
HardTableR2	0.9	**0.99**	1.0	1.0	1.0	1.0
HardTableR3	0.4	**1.0**	1.0	1.0	1.0	1.0

6.4 Validation of Human Feedback

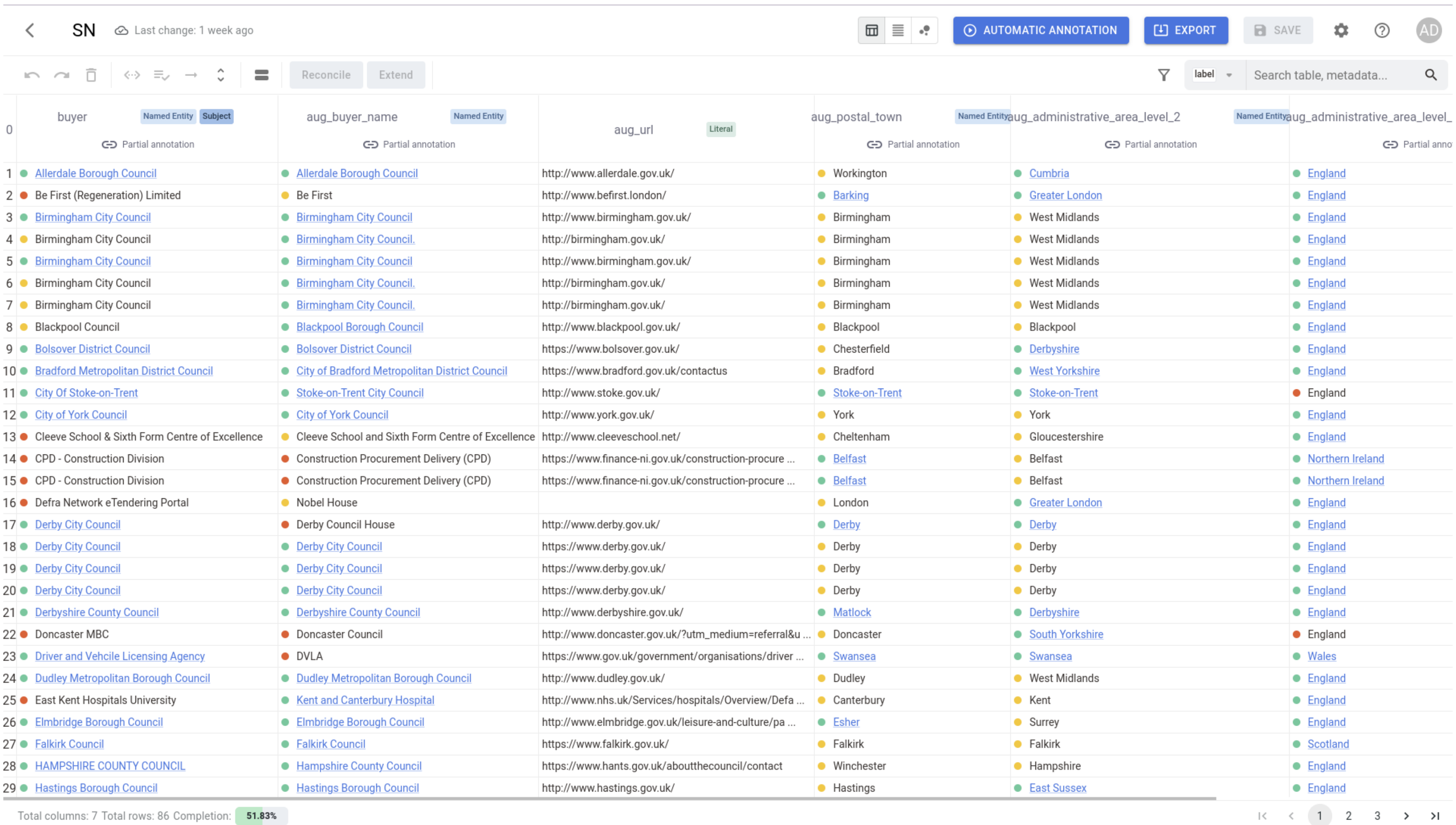
In this section, the methodology employed for validating the Model Revision aspect of the research is delineated. Specifically, it focuses on the simulation of human feedback using an oracle, as discussed in Section 3.5. The challenge of 'catastrophic forgetting' (as outlined in Section 3.5.4) is also addressed in the context of fine-tuning the model.

To comprehensively validate the Human-in-the-Loop (HITL) component and explore its potential, experiments were conducted across all the datasets introduced earlier. These datasets include *Round1_T2D*, *Round3*, *Round4*, *2T-2020*, *HardTableR2*, and *HardTableR3*. The primary objective of these experiments was to assess the impact of model fine-tuning via (simulated) human feedback.

In detail, different configurations were examined, and the results are presented in Tables 6.2, 6.3, 6.4, 6.5, 6.6, and 6.7. Each table displays the outcomes for a specific dataset. The two headers in these tables serve distinct purposes: the first header (expressed as a percentage) over the rows indicates the percentage of data that the Neural Ranker has been exposed to for fine-tuning, a dataset that the model has never encountered before. The second header (referred to as 'factor') over the columns represents a multiplier used to determine how many instances to extract from the original training dataset to mitigate the *catastrophic forgetting* challenge.

To clarify, the number of instances to extract from the original training dataset is determined by calculating a percentage of the total training instances, which is then multiplied by the factor. For instance, if 5% of the training instances are considered, and there are 1000 instances in total, the number of instances to use from the original training dataset is calculated as 1000 multiplied by the factor. For example, with a factor of 2, 2000 instances from the previous training set are considered.

The results from these experiments clearly indicate the successful adaptation of the model. Furthermore, they highlight the importance of including a substantial number of instances from the previous training dataset, with a factor of 2 being the optimal choice.



0	buyer (Named Entity, Subject; Partial annotation)	aug_buyer_name (Named Entity; Partial annotation)	aug_url (Literal)	aug_postal_town (Named Entity; Partial annotation)	aug_administrative_area_level_2 (Named Entity; Partial annotation)	aug_administrative_area_level_ (Partial anno)
1	Allerdale Borough Council	Allerdale Borough Council	http://www.allerdale.gov.uk/	Workington	Cumbria	England
2	Be First (Regeneration) Limited	Be First	http://www.befirst.london/	Barking	Greater London	England
3	Birmingham City Council	Birmingham City Council	http://www.birmingham.gov.uk/	Birmingham	West Midlands	England
4	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
5	Birmingham City Council	Birmingham City Council	http://www.birmingham.gov.uk/	Birmingham	West Midlands	England
6	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
7	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
8	Blackpool Council	Blackpool Borough Council	http://www.blackpool.gov.uk/	Blackpool	Blackpool	England
9	Bolsover District Council	Bolsover District Council	https://www.bolsover.gov.uk/	Chesterfield	Derbyshire	England
10	Bradford Metropolitan District Council	City of Bradford Metropolitan District Council	https://www.bradford.gov.uk/contactus	Bradford	West Yorkshire	England
11	City Of Stoke-on-Trent	Stoke-on-Trent City Council	http://www.stoke.gov.uk/	Stoke-on-Trent	Stoke-on-Trent	England
12	City of York Council	City of York Council	http://www.york.gov.uk/	York	York	England
13	Cleeve School & Sixth Form Centre of Excellence	Cleeve School and Sixth Form Centre of Excellence	http://www.cleeveschool.net/	Cheltenham	Gloucestershire	England
14	CPD - Construction Division	Construction Procurement Delivery (CPD)	https://www.finance-ni.gov.uk/construction-procure ...	Belfast	Belfast	Northern Ireland
15	CPD - Construction Division	Construction Procurement Delivery (CPD)	https://www.finance-ni.gov.uk/construction-procure ...	Belfast	Belfast	Northern Ireland
16	Defra Network eTendering Portal	Nobel House		London	Greater London	England
17	Derby City Council	Derby Council House	http://www.derby.gov.uk/	Derby	Derby	England
18	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
19	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
20	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
21	Derbyshire County Council	Derbyshire County Council	http://www.derbyshire.gov.uk/	Matlock	Derbyshire	England
22	Doncaster MBC	Doncaster Council	http://www.doncaster.gov.uk/?utm_medium=referral&u ...	Doncaster	South Yorkshire	England
23	Driver and Vehcile Licensing Agency	DVLA	https://www.gov.uk/government/organisations/driver ...	Swansea	Swansea	Wales
24	Dudley Metropolitan Borough Council	Dudley Metropolitan Borough Council	http://www.dudley.gov.uk/	Dudley	West Midlands	England
25	East Kent Hospitals University	Kent and Canterbury Hospital	http://www.nhs.uk/Services/hospitals/Overview/Defa ...	Canterbury	Kent	England
26	Elmbridge Borough Council	Elmbridge Borough Council	http://www.elmbridge.gov.uk/leisure-and-culture/pa ...	Esher	Surrey	England
27	Falkirk Council	Falkirk Council	https://www.falkirk.gov.uk/	Falkirk	Falkirk	Scotland
28	HAMPSHIRE COUNTY COUNCIL	Hampshire County Council	https://www.hants.gov.uk/aboutthecouncil/contact	Winchester	Hampshire	England
29	Hastings Borough Council	Hastings Borough Council	http://www.hastings.gov.uk/	Hastings	East Sussex	England

Total columns: 7 Total rows: 86 Completion: 51.83%

Figure 6.7: Annotated table without threshold adjustment.

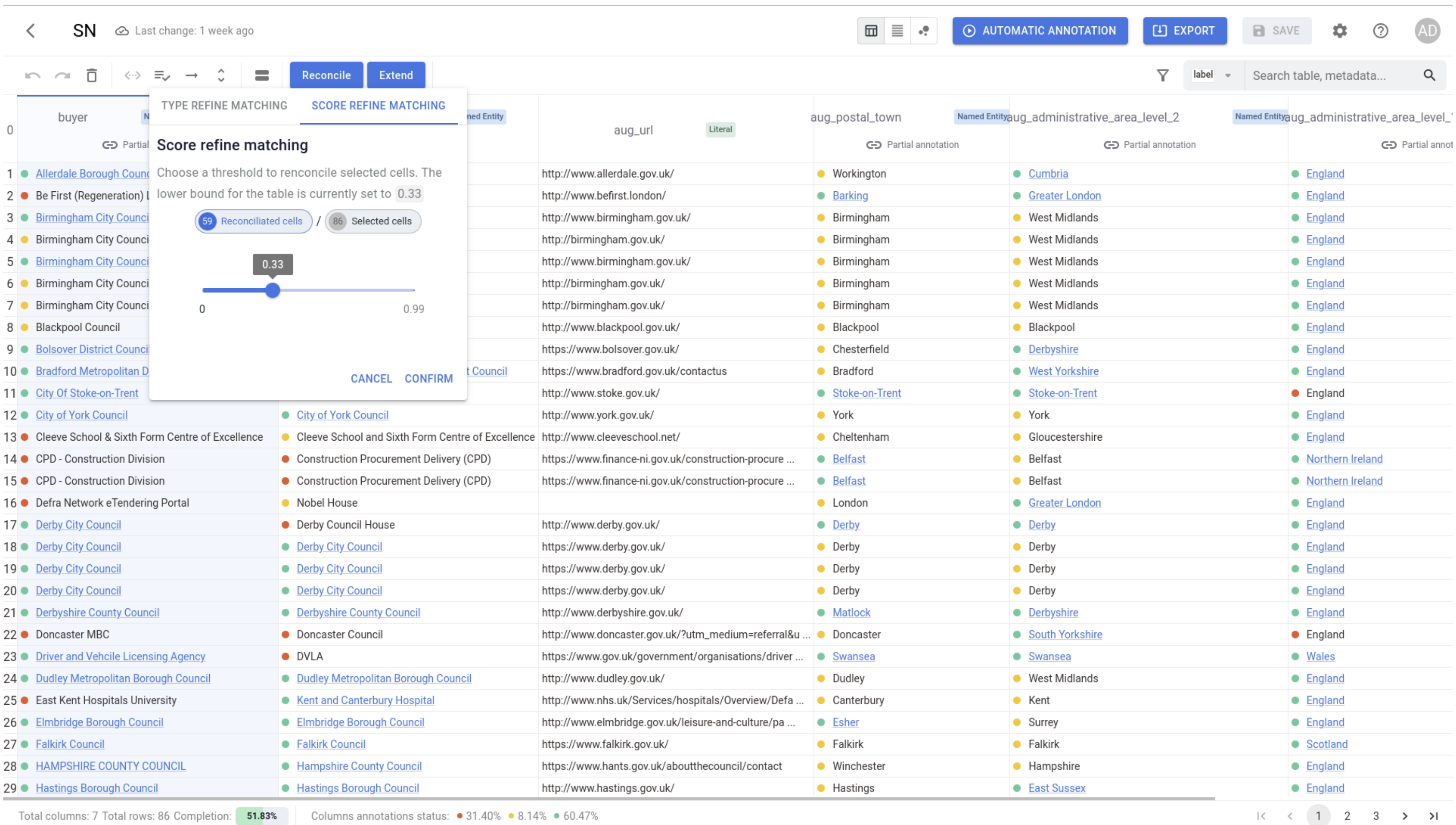
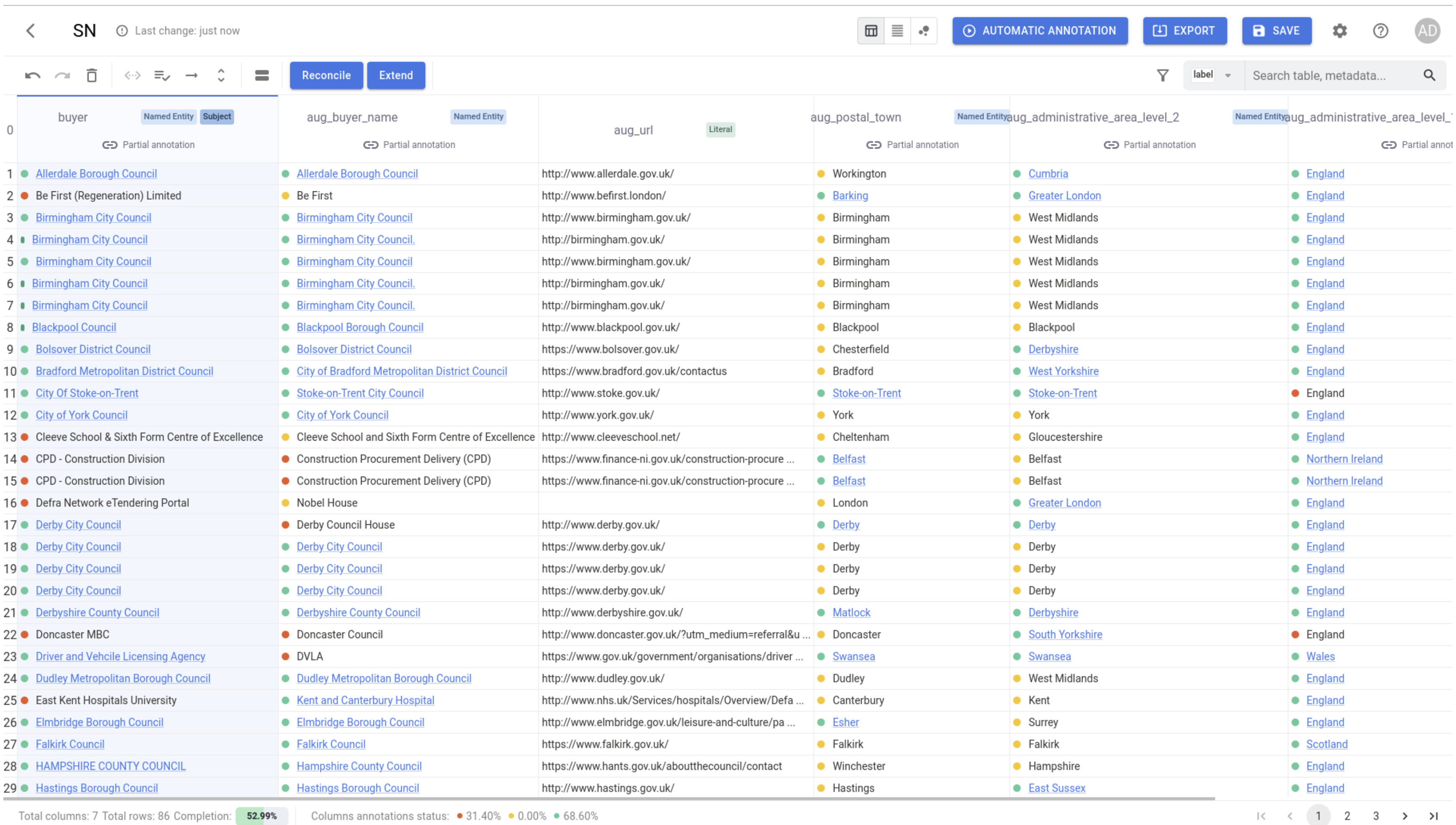


Figure 6.8: Annotated table with threshold value highlighted.



0	buyer (Named Entity, Subject; Partial annotation)	aug_buyer_name (Named Entity; Partial annotation)	aug_url (Literal)	aug_postal_town (Named Entity; Partial annotation)	aug_administrative_area_level_2 (Named Entity; Partial annotation)	aug_administrative_area_level_' (Partial annot)
1	Allerdale Borough Council	Allerdale Borough Council	http://www.allerdale.gov.uk/	Workington	Cumbria	England
2	Be First (Regeneration) Limited	Be First	http://www.befirst.london/	Barking	Greater London	England
3	Birmingham City Council	Birmingham City Council	http://www.birmingham.gov.uk/	Birmingham	West Midlands	England
4	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
5	Birmingham City Council	Birmingham City Council	http://www.birmingham.gov.uk/	Birmingham	West Midlands	England
6	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
7	Birmingham City Council	Birmingham City Council.	http://birmingham.gov.uk/	Birmingham	West Midlands	England
8	Blackpool Council	Blackpool Borough Council	http://www.blackpool.gov.uk/	Blackpool	Blackpool	England
9	Bolsover District Council	Bolsover District Council	https://www.bolsover.gov.uk/	Chesterfield	Derbyshire	England
10	Bradford Metropolitan District Council	City of Bradford Metropolitan District Council	https://www.bradford.gov.uk/contactus	Bradford	West Yorkshire	England
11	City Of Stoke-on-Trent	Stoke-on-Trent City Council	http://www.stoke.gov.uk/	Stoke-on-Trent	Stoke-on-Trent	England
12	City of York Council	City of York Council	http://www.york.gov.uk/	York	York	England
13	Cleeve School & Sixth Form Centre of Excellence	Cleeve School and Sixth Form Centre of Excellence	http://www.cleeveschool.net/	Cheltenham	Gloucestershire	England
14	CPD - Construction Division	Construction Procurement Delivery (CPD)	https://www.finance-ni.gov.uk/construction-procure ...	Belfast	Belfast	Northern Ireland
15	CPD - Construction Division	Construction Procurement Delivery (CPD)	https://www.finance-ni.gov.uk/construction-procure ...	Belfast	Belfast	Northern Ireland
16	Defra Network eTendering Portal	Nobel House		London	Greater London	England
17	Derby City Council	Derby Council House	http://www.derby.gov.uk/	Derby	Derby	England
18	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
19	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
20	Derby City Council	Derby City Council	https://www.derby.gov.uk/	Derby	Derby	England
21	Derbyshire County Council	Derbyshire County Council	http://www.derbyshire.gov.uk/	Matlock	Derbyshire	England
22	Doncaster MBC	Doncaster Council	http://www.doncaster.gov.uk/?utm_medium=referral&u ...	Doncaster	South Yorkshire	England
23	Driver and Vehcile Licensing Agency	DVLA	https://www.gov.uk/government/organisations/driver ...	Swansea	Swansea	Wales
24	Dudley Metropolitan Borough Council	Dudley Metropolitan Borough Council	http://www.dudley.gov.uk/	Dudley	West Midlands	England
25	East Kent Hospitals University	Kent and Canterbury Hospital	http://www.nhs.uk/Services/hospitals/Overview/Defa ...	Canterbury	Kent	England
26	Elmbridge Borough Council	Elmbridge Borough Council	http://www.elmbridge.gov.uk/leisure-and-culture/pa ...	Esher	Surrey	England
27	Falkirk Council	Falkirk Council	https://www.falkirk.gov.uk/	Falkirk	Falkirk	Scotland
28	HAMPSHIRE COUNTY COUNCIL	Hampshire County Council	https://www.hants.gov.uk/aboutthecouncil/contact	Winchester	Hampshire	England
29	Hastings Borough Council	Hastings Borough Council	http://www.hastings.gov.uk/	Hastings	East Sussex	England

Total columns: 7 Total rows: 86 Completion: 52.99% | Columns annotations status: 31.40% 0.00% 68.60% | 1 2 3

Figure 6.9: Annotated table after the application of the adjusted threshold.

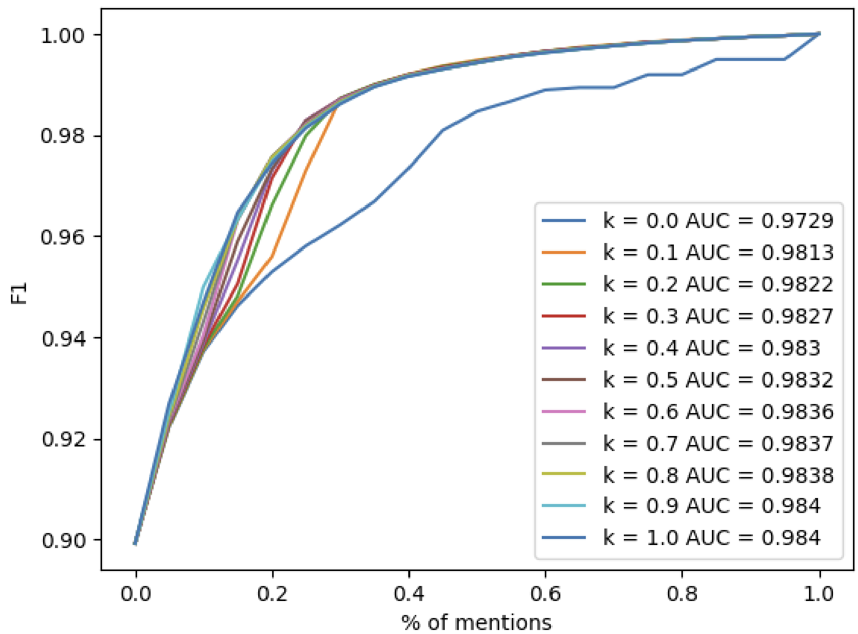


Figure 6.10: F1 and AUC computed for the training dataset.

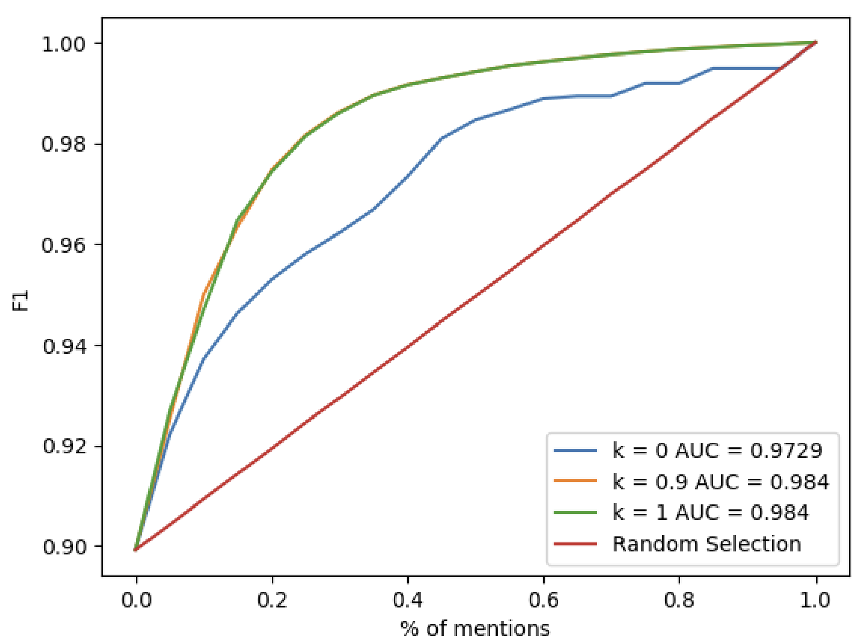


Figure 6.11: F1 and AUC computed for the test dataset.

Table 6.2: Result of Tuning on *Round1_T2D*

% vs factor	**0.5**	**1**	**2**
1%	0.880	0.850	0.886
3%	0.777	0.860	0.876
5%	0.815	0.858	0.880
10%	0.855	0.857	0.877
20%	0.882	0.894	0.892
30%	0.887	**0.899**	0.896

Table 6.3: Results of Tuning on *Round3*

% vs factor	**0.5**	**1**	**2**
1%	0.815	0.765	0.754
3%	0.762	0.742	0.790
5%	0.816	0.827	0.846
10%	0.825	0.865	0.871
20%	0.868	0.881	0.884
30%	0.887	0.890	**0.896**

Table 6.4: Results of Tuning on *Round4*

% vs factor	**0.5**	**1**	**2**
1%	0.905	0.929	0.913
3%	0.889	0.887	0.948
5%	0.941	0.926	0.958
10%	0.945	0.946	0.964
20%	0.958	0.966	0.960
30%	0.957	**0.967**	0.964

Table 6.5: Results of Tuning on *2T-2020*

% vs factor	**0.5**	**1**	**2**
1%	0.785	0.900	0.929
3%	0.877	0.928	0.966
5%	0.917	0.970	0.971
10%	0.949	0.972	0.976
20%	0.974	0.978	0.980
30%	0.976	0.979	**0.981**

Table 6.6: Results of Tuning on *HardTableR2*

% vs factor	**0.5**	**1**	**2**
1%	0.938	0.932	0.935
3%	0.938	0.924	0.925
5%	0.947	0.942	0.942
10%	0.945	0.943	0.961
20%	0.948	0.963	**0.970**
30%	0.953	0.967	0.969

Table 6.7: Results of Tuning on *HardTableR3*

% vs factor	**0.5**	**1**	**2**
1%	0.807	0.836	0.928
3%	0.942	0.954	0.962
5%	0.964	0.969	0.949
10%	0.967	0.961	0.973
20%	0.970	0.970	0.976
30%	0.968	0.973	**0.979**

7. Entity Linking in Large-Scale Data Environments

In the context of STI, handling the substantial amounts of data inherent to the field represents a significant challenge, often involving the reconciliation of numerous tables that can range in the thousands or even millions. Developing an effective data pipeline strategy stands as a promising solution.

This chapter outlines a prospective architecture of a data pipeline designed to aid the process of large-scale data handling, consequently facilitating scalable and efficient entity linking. It navigates through essential components and stages such as data ingestion, pre-processing, entity recognition, candidate generation, and linking.

Furthermore, the chapter highlights the importance of implementing techniques to optimize pipeline functionality, which includes the adaptation of strategies like parallelization, distributed computing, and the utilization of cloud-based solutions. The text also underscores the necessity for a consistent monitoring approach that encompasses aspects such as memory consumption, CPU usage, and execution time, to garner useful insights into the operational processes within the pipeline.

Concluding the chapter, readers will find themselves equipped with a foundational understanding of data pipeline architectures and their role in enhancing entity linking over large data sets. It is intended to prepare individuals to navigate the hurdles associated with processing large-scale data and to introduce strategies for scalable cloud solutions, thus setting the stage for effective knowledge extraction and discovery in large-scale environments.

7.1 Data Pipeline: An Overview

In the realm of entity linking, the pathway data travels from ingestion to final processing determines the efficiency and scalability of the whole operation, particularly over massive datasets. This overview provides a glimpse into what constitutes a data pipeline and its importance.

Why a Data Pipeline?

- **Volume**: Tabular data, especially from diverse sources, can accumulate rapidly. A structured pipeline helps manage and process this data without causing system inefficiencies.
- **Flow**: Data isn't just static; it moves. From its raw state, through various stages of cleaning, processing, and linking, its transition needs management.
- **Scalability**: With increasing data, the system should be able to handle the growth without compromising on performance.

Core Components

- **Data Ingestion**: The entry point, where data is sourced from diverse origins, be it databases, files, or even real-time streams.

- **Data Storage**: Following ingestion, it's imperative to store data efficiently. Storage isn't merely about saving data, but also ensuring its rapid retrieval.
- **Data Preprocessing**: Often, data isn't in its most usable state. This phase ensures data cleaning, normalization, and transformation.
- **Entity Linking Stages**: Specific to this context, these involve candidate retrieval and resolution processes.
- **Data Output**: Post entity linking, the enriched data needs to be channeled either to other systems or stored for subsequent usage.

Challenges

Every pipeline, especially when catering to vast datasets, encounters challenges:

- **Scalability** remains atop, alongside managing **latency**, maintaining **data integrity**, and ensuring **fault tolerance**.

7.2 Data pipelines

In order to test the proposed solution, some data pipelines with multiple steps were needed. Keeping this in mind, the data pipelines developed as part of the European project enRichMyData become useful. The enRichMyData project aims to provide a novel paradigm and a toolbox for building rich, high-quality, and valuable datasets to feed Big Data Analytics and AI applications[1]. It aims at facilitating the specification and scalable execution of data enrichment pipelines, with a focus on supporting various data enrichment operations such as discovery, understanding, selection, cleaning, transformation, integration of Big Data from a variety of sources. enRichMyData makes this paradigm easily accessible to a wide range of large and small organizations that encounter difficulties in delivering suitable data to feed their data analytics solutions, due to the lack of specific tools and expertise to support cost-effective and energy-efficient management of data enrichment pipelines.

In the context of the enRichMyData project, the application of data pipelines is limited in the context of data enrichment procedures. The tools used in this toolbox must satisfy to provide data enrichment solutions useful in several application domains, but also innovative in the context of the vast and competitive landscape of data management solutions.

The project will deliver its capabilities in a set of interoperable tools and services that will form the enRichMyData Toolbox, which will handle complex data enrichment scenarios. The Toolbox's conceptual architecture is centered around the data enrichment pipeline that receives

[1] https://enrichmydata.eu/

input data to be enriched and data to enrich with, generating enriched data. The enrichment process is supported by a set of functional capabilities related to supporting the design of pipelines and non-functional capabilities related to supporting the effective and efficient deployment and execution of pipelines. The main goals of the project are the following ones[2]:

- *Discovery of potentially valuable data for data enrichment:* improve data discovery and profiling featuring search on data, ontologies, and semantic data profiles to identify potentially valuable data for data enrichment;
- *Wrapping data sources in different formats:* improve wrapping of data sources in different formats so they can be securely accessed as virtual semantic graphs and used more easily for data enrichment;
- *Simplified cleaning, linking and extension of data:* simplify cleaning, linking (to reference resources), and extension of structured and semi-structured data, featuring approaches that enable users to specify such operations visually;
- *Simplified annotation and classification of data:* simplify annotation and classification of textual data, featuring entity and concept extraction, feature extraction (via embeddings), and classification with predefined and custom classifiers;
- *Support the management of data enrichment pipelines:* creation and operation of data linking and extension services, a framework for deployment and execution of pipelines at a large scale, and reuse and extension of existing pipelines to deliver a hub of data and services for data enrichment;
- *Support data streaming in data enrichment pipelines:* featuring support for setting up appropriate endpoints and ensuring high throughput during pipeline execution;
- *Energy consumption reduction for data enrichment pipelines:* monitor and reduce energy consumption for executing data enrichment pipelines using models to estimate and track their carbon footprint.

A better understanding of the mentioned process in described in the Figure 7.1 below.

The presented Toolbox[4] consists of the joint work from 13 partners from 11 countries. Each of them contributes to some extents at the development of the tools. The tools could be either tools that provide functional capabilities needed to support the design of pipelines, or infrastructure services that provide non-functional capabilities needed to support the effective and efficient deployment and execution of pipelines. In short, that is their main functionalities:

[2] https://enrichmydata.eu/about-2

[4] https://enrichmydata.github.io/toolbox

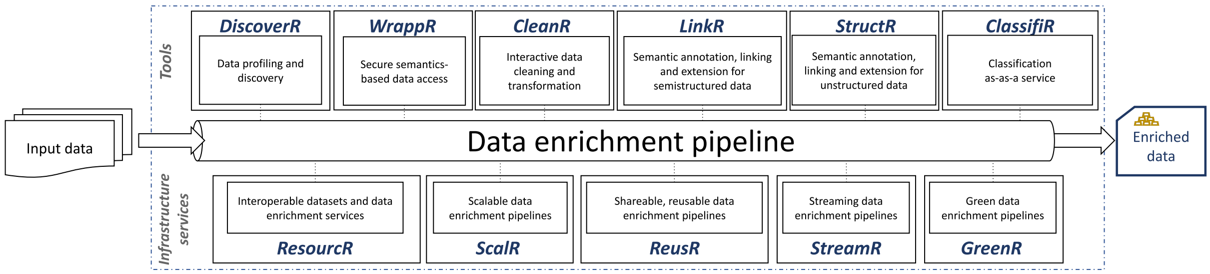


Figure 7.1: Architecture of enRichMyData Toolbox[3]

- *DiscoverR* enhances user access to datasets, ontologies, and enrichment services for pipelines. Users can keyword search cataloged items, including metadata, formats, ontology terms, and quality features, which encompass well-known knowledge bases and pipeline data. DiscoverR employs semantic data profiling to enrich descriptions with metadata and ontology patterns, supporting FAIR principles;

- *WrappR* offers secure data access through a virtual semantic layer. It's a semantic graph database with efficient reasoning, cluster support, and external index synchronization. WrappR provides diverse APIs, access methods, data federation, and virtualization options. It's a practical, robust, and versatile tool for enhanced data accessibility through semantic integration.;

- *CleanR* enables interactive specification of data cleaning and knowledge graph generation from diverse formats. Specifications are stored for reuse. It offers AI-driven transformations, integrated with ResourcR. CleanR supports sharing, management, and integration of operations in ScalR pipelines;

- *LinkR* annotates data using knowledge graphs (e.g., WikiData, DBpedia) and user-defined graphs from ResourcR. ML algorithms recommend annotations, and a human-in-the-loop approach ensures high-quality results with minimal user effort. Annotations become data transformations for enrichment pipelines;

- *StructR* converts unstructured text into structured data through semantic annotation and embedding. It identifies concepts, offers various pre-computed embeddings, and supports custom annotation services through a user-friendly interface for building new annotation models;

- *ClassifiR* complements StructR by providing document-level classification using standard or custom labels. It offers an interactive interface for creating custom classifications and automates the process through a unified endpoint, regardless of the classification method;

- *ResourcR* facilitates the creation of linking services and data access mechanisms for datasets. It streamlines linking and search operations, offering search and linking APIs. When combined with LinkR, it transforms toolbox-produced semantic data into readily reusable resources;

- *ScalR* enables large-scale data operations via containerized pipelines, flexible procedure management, and reusable templates. It promotes the deployable reuse of data enrichment processes, avoiding non-reusable code fragments;

- *ReusR* facilitates asset search, login, access control, editing, and sharing for data enrichment pipelines. It encourages pipeline editing and reuse across various use cases within the enRichMyData toolkit;

- *StreamR* supports streaming in data enrichment pipelines by managing data streams between endpoints with high throughput. It offers configurable tools for creating custom streams for different applications;

- *GreenR* monitors data enrichment pipelines for their environmental impact, tracking carbon footprints of pipeline components. It presents the results through a dashboard, allowing users to manage and mitigate the environmental impact of resource-intensive computations.

Given the described tools, it was possible to define relevant data pipelines for both merging technical functionalities and building a real-world application to satisfy the business cases of the project. In this context, a data pipeline is an adhesive code between the data sources and data enrichment algorithms used in the Toolbox. The data needs to be reconstructed, enriched and conveyed before training of algorithms or analysis commences. A workflow is a feasible solution for the definition of such pipelines, gathering and standardizing the required data. Since the entire process is divided into numerous levels, parallelized computing procedures are used. The pipeline workflow design is not an uncomplicated task; realizing the whole data processing requires some boilerplate codes. Workflow builders standardize the orchestration of data pipelines by using workflow engines and frameworks. Another best practice used for improving the standardization of data pipelines is the usage of container-based technologies.

7.3 Entity Linking at scale using Alligator Scalable Version

In Figure 7.2, the complete and detailed pipeline employed in this study is presented. This pipeline involves several distinct tools, namely Alligator, LamAPI, and the classifier. These tools are categorized as follows, as introduced earlier in Section 7.2: Alligator belongs to the category of *LinkR*, LamAPI is categorized as *ResourceR*, and the classifier belongs to the *classifiR* category.

The Alligator pipeline consists of the following steps:

- *Data Analysis & Pre-processing:* in this initial phase, the columns are divided into NE-columns, which contain potential entity mentions, and LIT-columns, which contain literal values. All text data is transformed to lower case, and special characters (i.e. underscores and extra spaces) are eliminated for cleaner, more uniform data;
- *Entity lookup:* this step involves entity retrieval. For each entity mention, a set of candidates is derived via the LamAPI;
- *Features extraction:* for each candidate, a set of features is computed. These features are used to identify the correct candidate and can be categorized into three types: mention features, entity features, and mention-entity features. Some features focus only on text similarity, while others consider the broader context;
- *Initial predictions:* the initial prediction is made using a Machine Learning model, resulting in a preliminary ranking of candidates;
- *Features extraction revision:* the initial predictions from the previous stage are used to provide some context to the enrichment procedure. At this stage, the features that consider the congruence of types and predicates collected from the candidates are refined;
- *Final predictions:* new predictions are made using the refined types and predicates data from the previous step;
- *API ClassifiR:* application of the classifier tool from Expert.AI[5] to enrich with tags the descriptions of the identified entities;
- *Export:* this is the final step, where a decision is made for each mention about whether to annotate it or not. This is based on the confidence score and the difference between the top two candidates.

Since Alligator is a pipeline from the enRichMyData project, it uses some tools from the project's toolbox.

In particular, **LamAPI (Label Matching API)** the tool was described in Chapter 4. The other external tool mentioned is the **Expert.AI Platform Document Classification,** capable of analyzing text to label and identify media topics, emotional traits, geographical references, and more. The document classification provided determines the subject of a text in terms of categories within a taxonomy. Currently, there are five different available taxonomies, and it supports five different languages.

The entire pipeline has been tested using Argo Workflows[6], an open-source container-native workflow engine designed for orchestrating parallel jobs on Kubernetes. Argo Workflows are implemented as Kubernetes Custom Resource Definitions (CRDs).

[5] https://www.expert.ai/

[6] https://argoproj.github.io/argo-workflows/

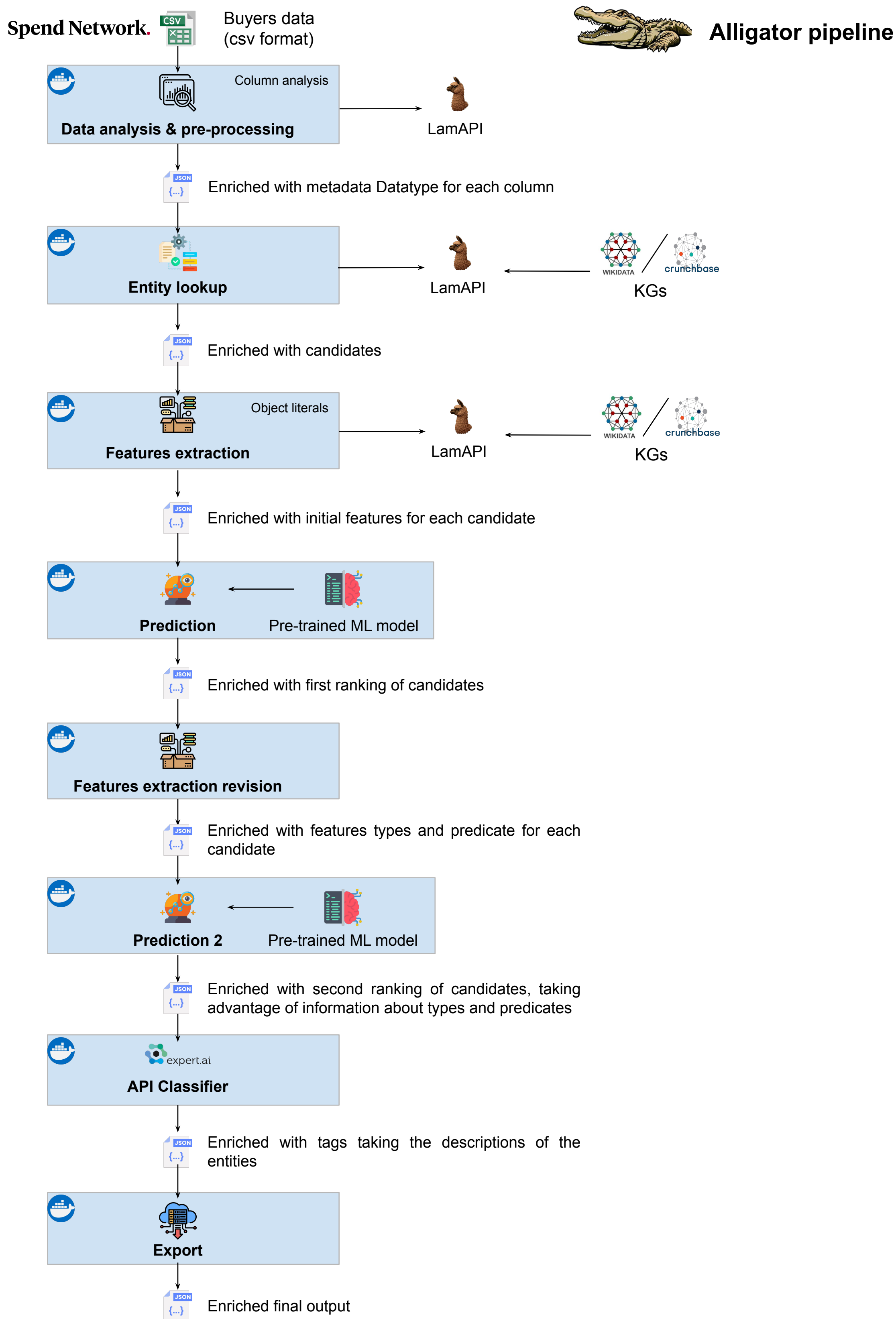


Figure 7.2: Example of data pipeline combining different tools

7.4 Evaluation of Alligator Scalable Version

The results analyzed in this section refer to the run of the Alligator pipeline described above using 75 rows of the dataset provided by **Spend Network**[7]. Spend Network is an open data company that provides insight and analytics for public sector spending. It works with public procurement and finance data to help suppliers and governments forge lasting partnerships. In the context of the enRichMyData project, Spend Network represents one of the use cases of the toolbox. In particular, as shown in Figure 7.2 the Alligator pipeline is applied to test the capabilities of LinkR (implemented by Alligator) and ClassifiR in finding additional information about companies using the Wikidata or Crunchbase[8] knowledge graph. Although, in this chapter only experiments with Wikidata KG will be presented, it is worth noting that the choice of the KG has marginal impact on resource consumption.

[7] https://spendnetwork.com/

[8] https://www.crunchbase.com/

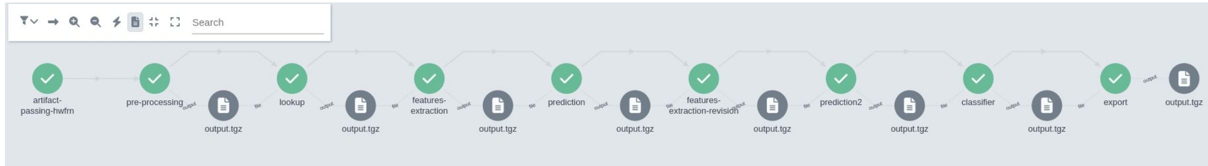


Figure 7.3: Representation of the Alligator pipeline's steps as visualized in the Argo Workflows interface.

```
   MESSAGE                          TEMPLATE                        PODNAME
✔ artifact-passing-rxrm7           alligator-pipeline
├──✔ pre-processing                 pre-processing                  artifact-passing-rxrm7-pre-processing-1760652163
├──✔ lookup                         lookup                          artifact-passing-rxrm7-lookup-4416589
├──✔ features-extraction            features-extraction             artifact-passing-rxrm7-features-extraction-1345747141
├──✔ prediction                     prediction                      artifact-passing-rxrm7-prediction-2333001188
├──✔ features-extraction-revision   features-extraction-revision    artifact-passing-rxrm7-features-extraction-revision-44551223
├──✔ prediction2                    prediction2                     artifact-passing-rxrm7-prediction2-1213565316
├──✔ classifier                     classifier                      artifact-passing-rxrm7-classifier-3591047579
└──✔ export                         export                          artifact-passing-rxrm7-export-1500004743
```

Figure 7.4: Representation of the Alligator pipeline's steps as observed from the command-line interface (CLI).

As delineated in Figures 7.3 and 7.4, the Alligator pipeline is structured linearly, comprising eight sequential steps. The SIM-PIPE framework affords flexibility in the deployment of the pipeline, allowing for execution either through a graphical front-end interface, as evidenced by Argo Workflows, or via a command-line interface (CLI). Regardless of the chosen mode of execution, the pipeline results remain accessible through the Argo Workflows interface. It should be noted that the partitioning of the pipeline into eight discrete steps was not driven by architectural necessities, but rather implemented to enhance the interpretability of the pipeline's operation.

7.4.1 Evaluating Pipeline Stepwise Time Performance

An empirical evaluation was conducted to assess the time performance of each component within the Alligator pipeline.

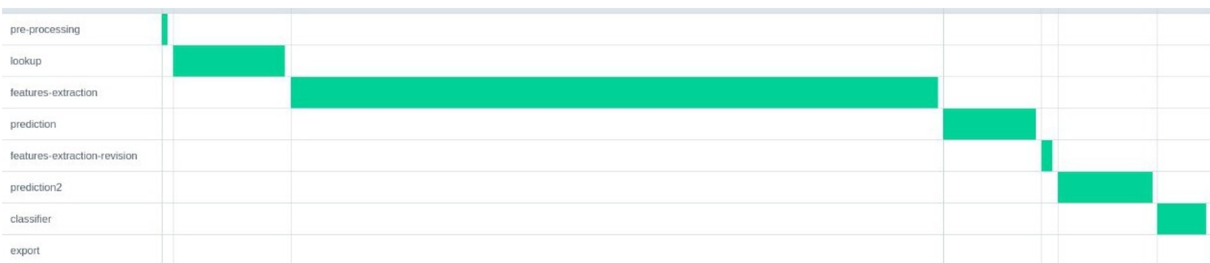


Figure 7.5: Temporal distribution of each step in the Alligator pipeline, as visualized in the Argo Workflows user interface.

Table 7.1: Duration of each step in the Alligator pipeline.

Step Name	Duration
Pre-processing	10s
Lookup	3m 11s
Features Extraction	19m 47s
Prediction	2m 15s
Features Extraction Revision	20s
Second Prediction	2m 31s
Classification	1m 4s
Export	14s

As delineated in Table 7.1 and visually represented in Figure 7.5, each pipeline step exhibits a unique time footprint. The overall duration for executing the pipeline on the sample data exceeds 31 minutes. A noteworthy observation is the disproportionate time allocation to the 'Features Extraction' step, which accounts for more than half of the overall time.

Upon closer inspection, a discrepancy emerges between the aggregate time of individual steps, which sums up to 29 minutes and 32 seconds, and the reported total execution time of

31 minutes and 13 seconds. This temporal gap can be attributed to the time elapsed between the completion of one step and the initiation of the subsequent step. Each of these steps is containerized within a distinct Docker container, contributing to this time differential.

7.4.2 Collection of Metrics for Pipeline Performance Evaluation

The performance of the Alligator pipeline was empirically evaluated using SIM-PIPE[9], a specialized tool designed for the testing and assessment of data pipelines [73]. SIM-PIPE provides detailed metrics on resource utilization, including CPU usage, memory allocation, and network bandwidth consumption.

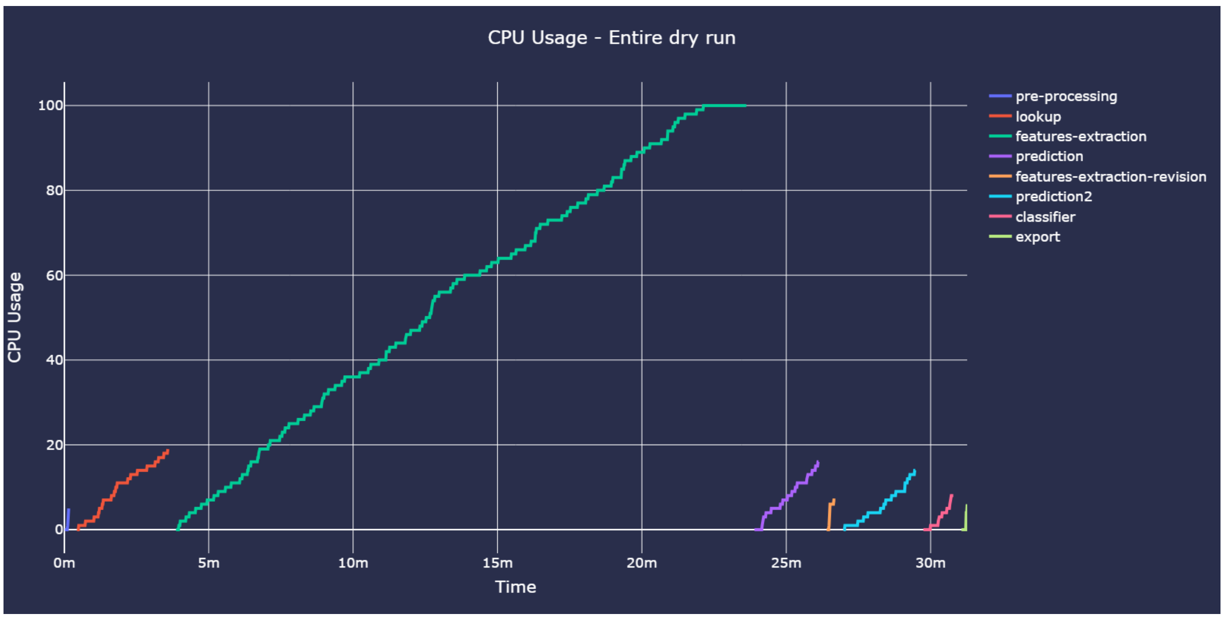


Figure 7.6: Central Processing Unit (CPU) utilization during the complete dry run of the Alligator pipeline.

The data, represented in Figures 7.6, 7.7, and 7.8, elucidate the resource consumption patterns across different stages of the Alligator pipeline. Most notably, the 'Features Extraction' stage manifests as the most resource-intensive, particularly during its terminal phase where CPU utilization reaches its maximum capacity of 100%. In contrast, the other stages did not demonstrate any significant impact on CPU, memory, or network bandwidth.

7.4.3 Assessing Pipeline Scalability

A critical consideration in the deployment of any data processing system is its scalability. To empirically assess the scalability of our pipeline for entity linking, we conducted an experiment to examine how execution time is affected when varying the number of processing nodes.

[9] https://github.com/DataCloud-project/SIM-PIPE

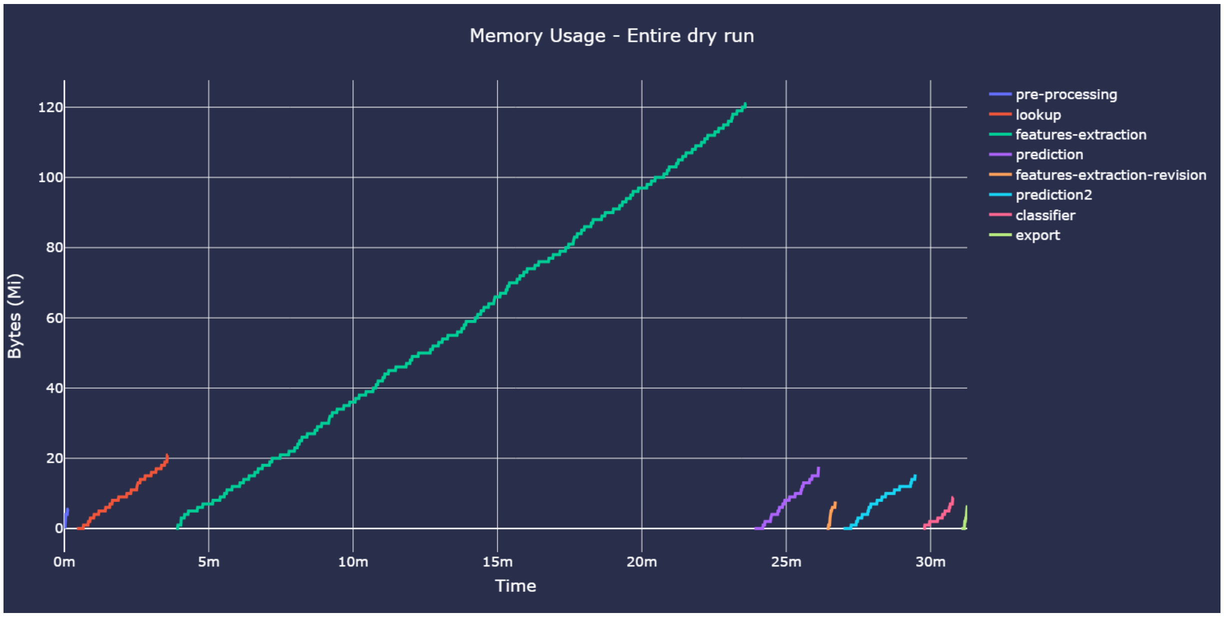


Figure 7.7: Memory utilization during the complete dry run of the Alligator pipeline.

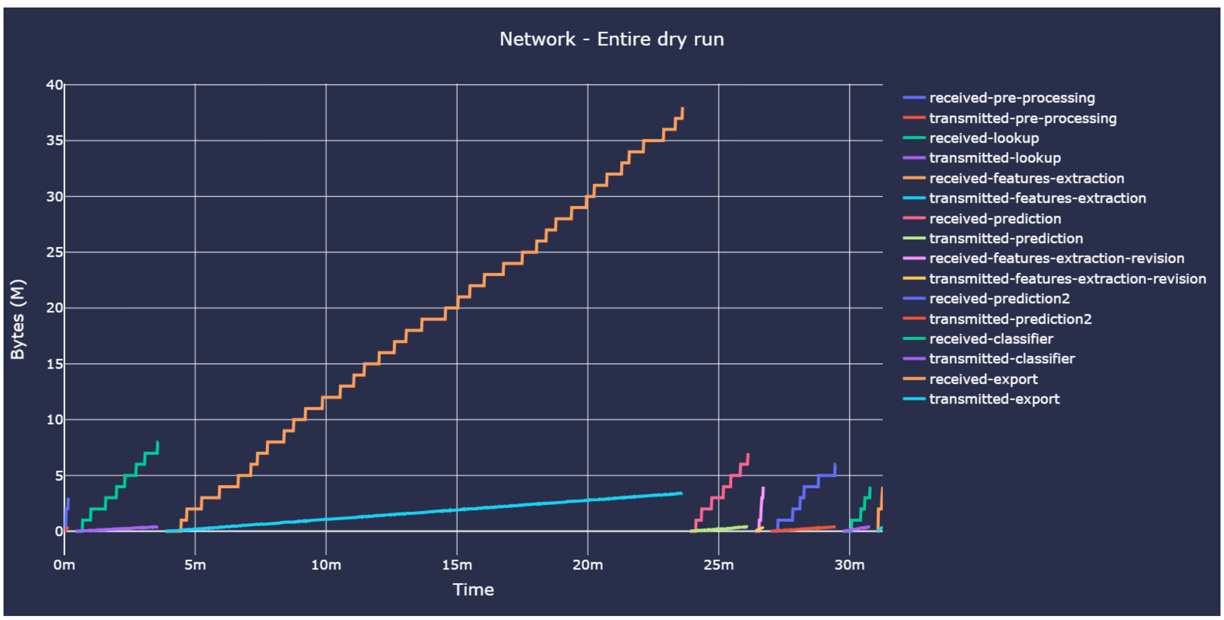


Figure 7.8: Network bandwidth utilization during the complete dry run of the Alligator pipeline.

In this experiment, a reference sample size of 120 rows was considered. This sample was then partitioned among the available processing nodes. For example, in a configuration with two nodes, each node processed 60 rows. With four nodes, each node processed 30 rows, and so on. The objective was to ascertain how well the system scales as the workload is distributed across an increasing number of nodes.

As depicted in Table 7.2, the execution time shows a decreasing trend as the number of processing nodes increases. Specifically, with 12 nodes each processing a partitioned sample

Table 7.2: Execution time as a function of the number of processing nodes and partitioned sample size.

Partitioned Sample Size	Number of Nodes	Time (sec)
120	1	1113
60	2	614
30	4	373
20	6	304
15	8	271
12	10	261
10	12	253

size of 10 rows, the execution time reduces to 253 seconds. This represents an approximate improvement by a factor of 4.4 compared to the time required for a single-node configuration that processes the entire sample of 120 rows, which is 1113 seconds. These findings affirm that the pipeline for entity linking is scalable and thus well-suited for large-scale data processing tasks.

8. Conclusion and Future Directions

The theme of data preparation for downstream analysis and exploitation is difficult to carry on manually, in particular when dealing with huge amount of, possibly unknown, data. In this thesis the topic of automatic annotation of tabular data has been discussed to augment the mentions in a table with links, types and properties found in a reference Knowledge Graph.

The adopted approach foresees a combination of heuristic-based and machine-learning-based algorithms to provide a tool that can annotate relational tables. The proposed algorithms have demonstrated to be effective with empirical validations over a set of general purpose datasets.

The Human-In-The-Loop paradigm allows for enhancing the results by both correcting the uncertain annotations and providing useful information to improve and specialize the algorithms for a specific domain. The user feedback can have immediate effect by expanding the corrections and changing the decision rules, and long-term effects by improving the model with the discovered annotations.

The encouraging results suggests to continue the investigation of new features that can improve the model to better support tailoring the solutions for specific domains. In such specialization activity, the role of the user need to be further investigated, in particular to fully exploit feedback collected during the review process.

Another critical issue that should be further investigated deals with the candidate retrieval activities that need to guarantee effectiveness, i.e., that the actual candidate is really present in the candidate set for a mention, and sustainability, i.e., that the candidate set for a mention is retrieved in a limited amount of time with minimal resource consumption.

Appendices

A. First Appendix

A.1 F1 score and AUC curve for the all Datasets

As depicted in Figures A.1, A.2, A.3, the results obtained for each dataset are displayed, following the procedure delineated in the Chapter 5 in the Subsection For each dataset, the k values are estimated using the other datasets. Subsequently, these deduced k values are applied to the test dataset, as shown in Figures A.4, A.5, A.6.

In particular, Figures A.1, A.2, A.3 show that the values of AUC are very close to each other then is means k is not so relevant. Nevertheless, another observation that can be made is that a k value of 0 is not advisable, as illustrated by the graphs. Consequently, it would be reasonable to select a k value greater than 0.5.

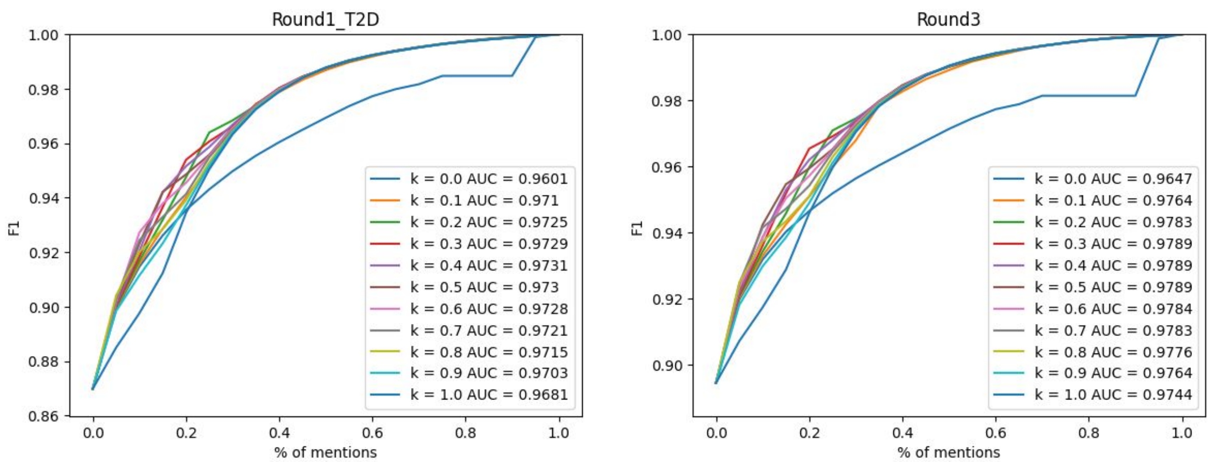


Figure A.1: F1 and AUC computed for train *Round1_T2D* and *Round3*

So in summary, how it is possible to see in the test results in Figures A.4, A.5, A.6 that is observable that, as expected, opting for a random solution consistently yields the least favorable outcome. Moreover, it is affirmed that selecting a k value of 0, as previously mentioned, is not a prudent choice.

A.2 Examining Variations in Scores over all Datasets

Figure A.7 presents the score trends for the 'WRONG' and 'CORRECT' cases in the *Round1_T2D* dataset. At $k = 0.4$, a significant downward shift in scores for the 'WRONG' cases is evident. By $k = 1$, a majority of the 'WRONG' cases have transitioned to lower score values, with a notable portion of 'CORRECT' cases also migrating towards these lower ranges.

In Figure A.8, the score trends for the *Round3* dataset are displayed. At $k = 0.5$, there is a prominent decrease in scores for the 'WRONG' cases. Conversely, when $k = 1$, while

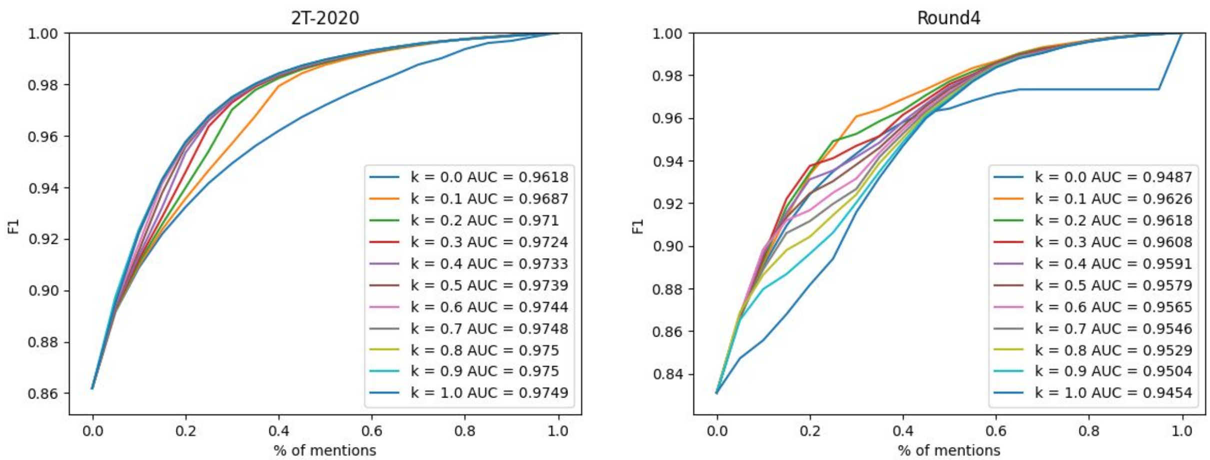


Figure A.2: F1 and AUC computed for train *2T-2020* and *Round4*

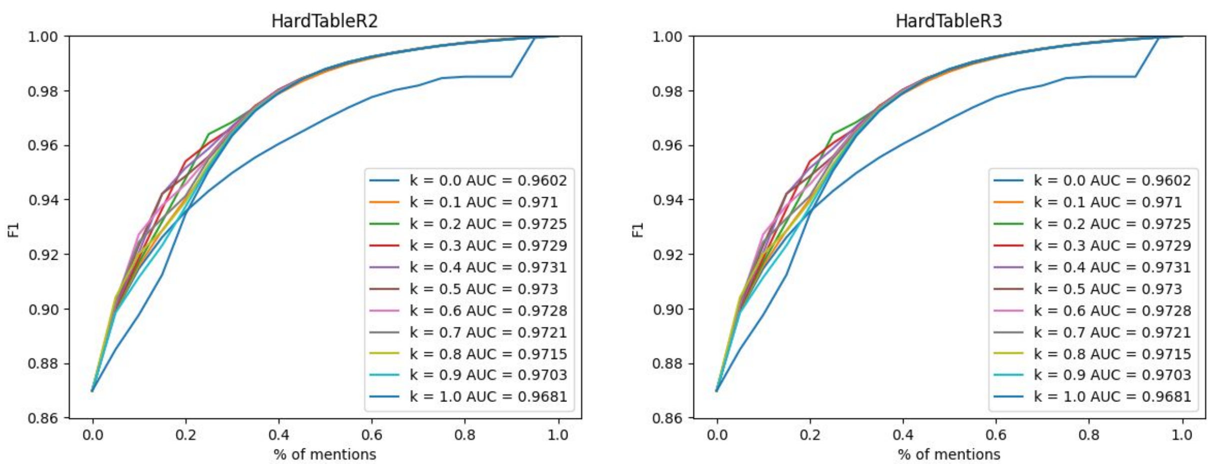


Figure A.3: F1 and AUC computed for train *HardTableR2* and *HardTableR3*

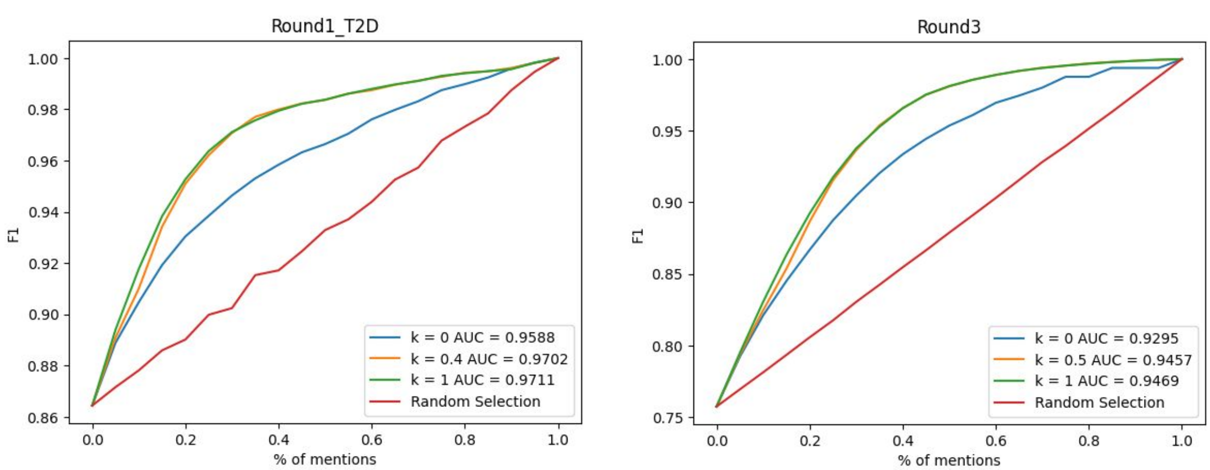


Figure A.4: F1 and AUC computed for test *Round1_T2D* and *Round3*

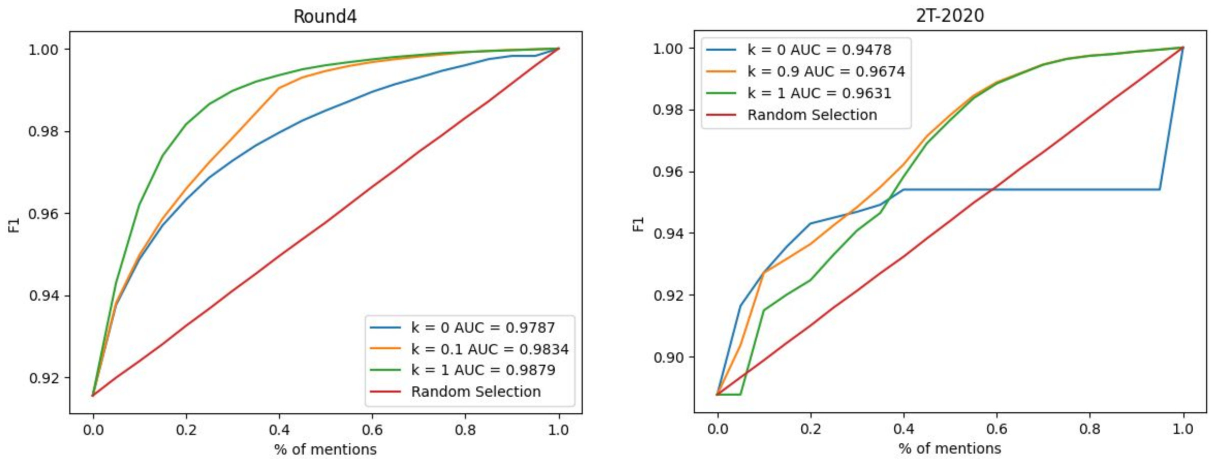


Figure A.5: F1 and AUC computed for test *2T-2020* and *Round4*

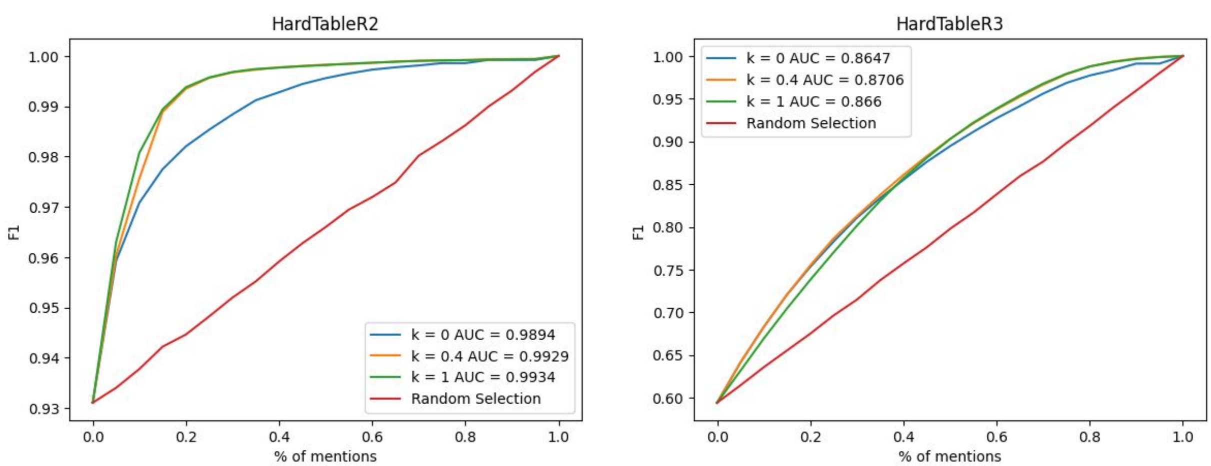


Figure A.6: F1 and AUC computed for test *HardTableR2* and *HardTableR3* the majority of 'WRONG' cases fall to lower values, many 'CORRECT' cases maintain their original score positions.

Figure A.9 displays the trends in the *Round4* dataset. At $k = 0.1$, scores for 'WRONG' cases considerably shift downwards. However, by $k = 1$, along with a significant portion of 'WRONG' cases transitioning to lower scores, a substantial number of 'CORRECT' cases also move to these reduced score brackets.

For the *2T-2020* dataset, as shown in Figure A.10, at $k = 0.9$, both 'WRONG' and a subset of 'CORRECT' cases experience a significant decline in scores. By the time k reaches 1, most 'WRONG' cases have decreased scores, but a sizeable fraction of 'CORRECT' cases also shifts to these lower values.

Lastly, Figure A.11 showcases the trends for the *HardTableR3* dataset. At $k = 0.4$, scores for the 'WRONG' cases exhibit a notable decrease. When k is set to 1, along with most 'WRONG' cases shifting to lower values, a distinct subset of 'CORRECT' cases also adopts these diminished score levels.

A.3 Examining Variations in Scores using Mammotab10K

In Chapter 5 in Section 5.3.2, score variations among different datasets and the various models employed were discussed. In this section, variations observed in the *Mammotab10K* dataset, a subset of the larger *Mammotab* dataset [48], are explored. Models previously trained on other datasets are utilized for this examination. This experiment serves to demonstrate the viability of the approach even when applied to an unfamiliar dataset, such as *Mammotab*, not used during the training phase. For the analysis, a k value of 0.6 is chosen. As established in Subsection 5.3.2 of Chapter 5, this choice is rational and also provides a consistent metric for cross-dataset comparison.

Figure A.12 displays the results obtained using the model tested on the *Round1_T2D* dataset. The graphs indicate that with $k = 0.6$, a majority of the 'WRONG' labels shift towards lower scores, while a significant portion of 'CORRECT' labels remain at higher score values. This pattern is similarly observed even when k is set to 1.

Figures A.13, A.14, A.15, A.16, and A.17 all illustrate a similar trend. The 'WRONG' labels predominantly shift towards the lower scores, and a considerable amount of 'CORRECT' labels maintain their high score values. However, it's worth noting that for the models tested on the *Round4* and *2T-2020* datasets, approximately 30% of the 'CORRECT' labels shift towards lower scores when k is set to 1.

In conclusion, based on the graphs presented in this section, the models developed demonstrate a commendable level of generality. Furthermore, there's a noticeable agreement across different datasets, with the k value of 0.6 proving to be an appropriate choice for all datasets considered.

A.4 Further Discussion and Analyses of Score Distributions

Figure A.18 illustrates the results obtained for the Round1_T2D dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.7 to 1. Additionally, the values of δ are primarily concentrated within the 0.6 to 1 range, indicating effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.7 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.3 range.

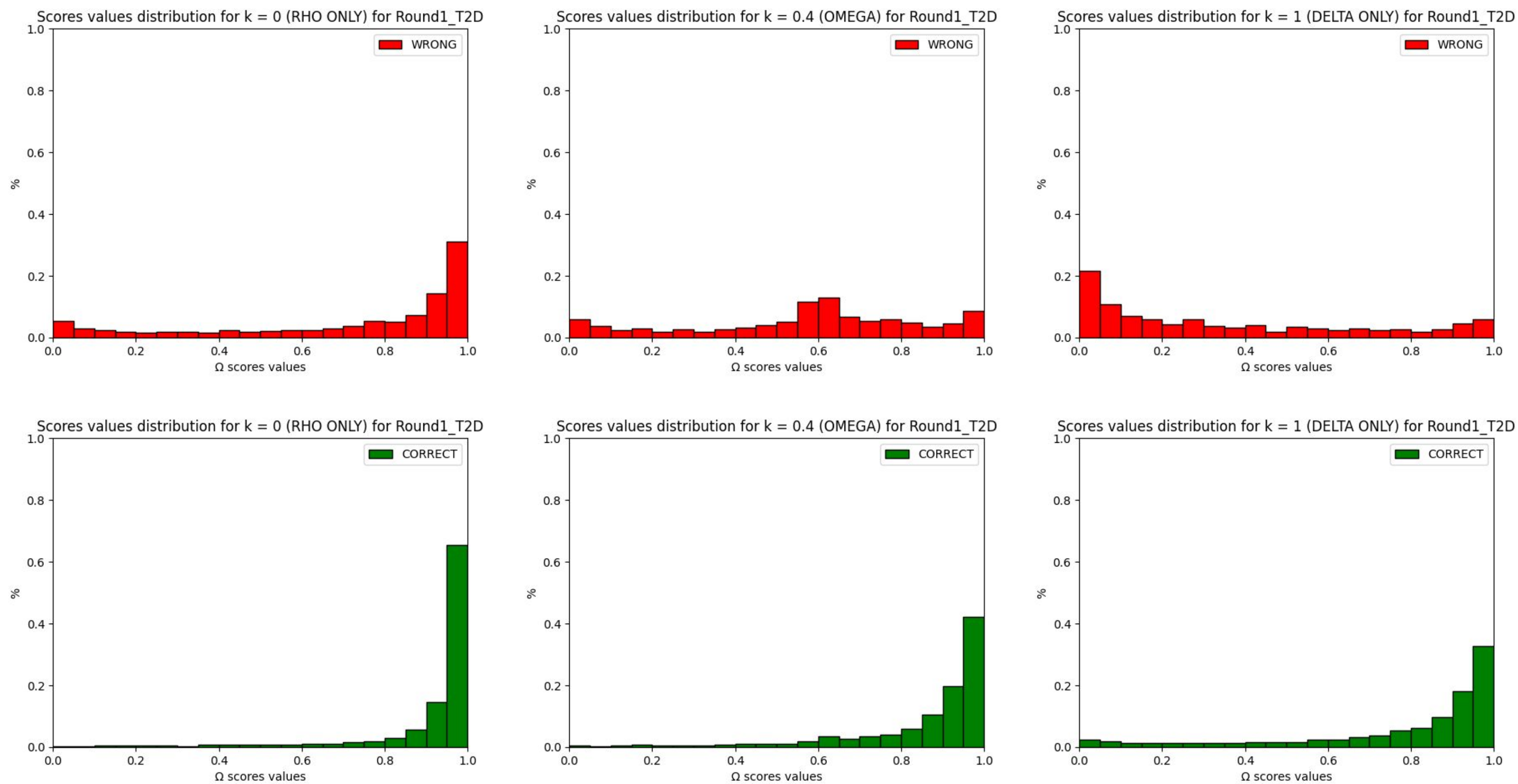


Figure A.7: Scores variations for Round1_T2D dataset

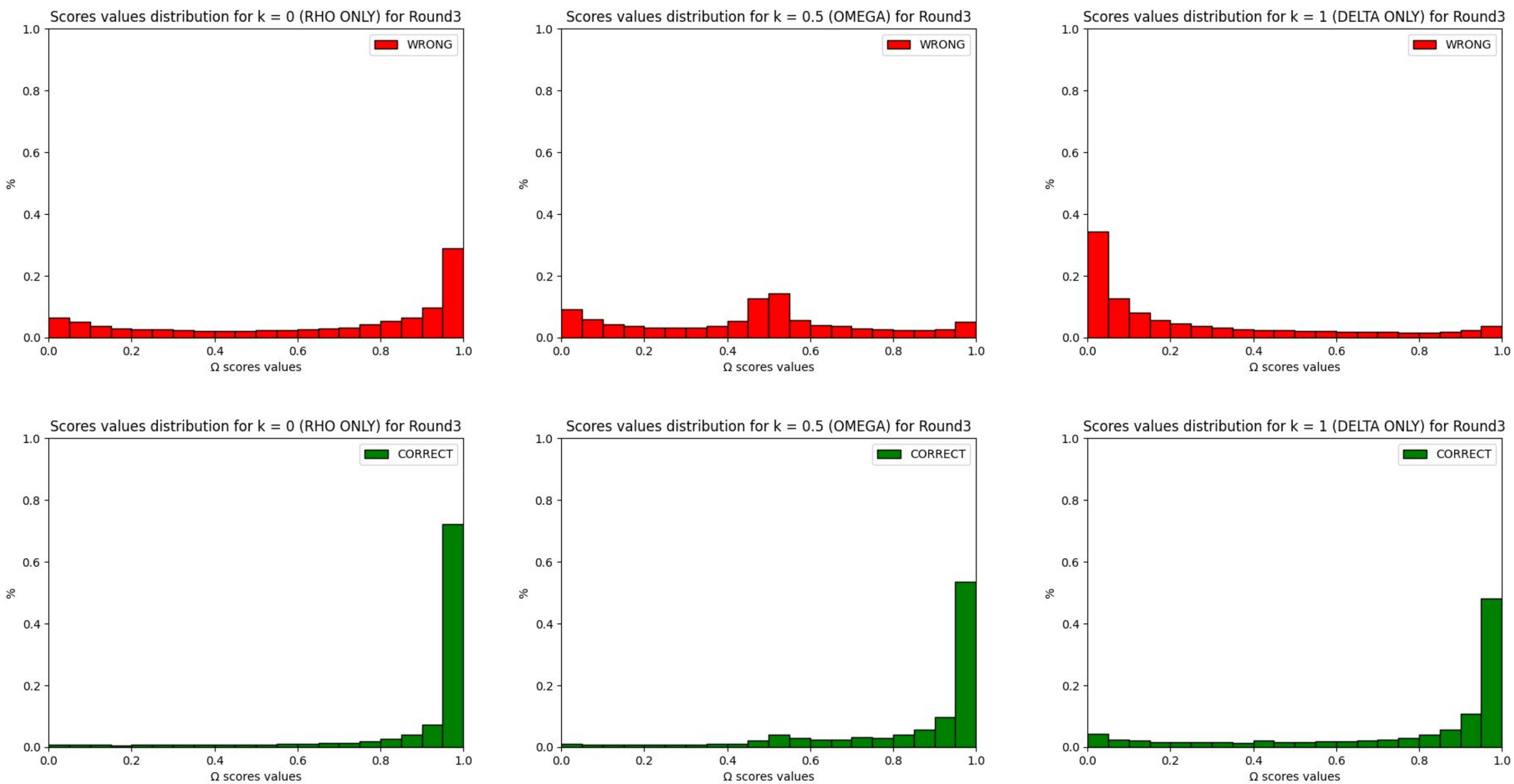


Figure A.8: Scores variations for Round3 dataset

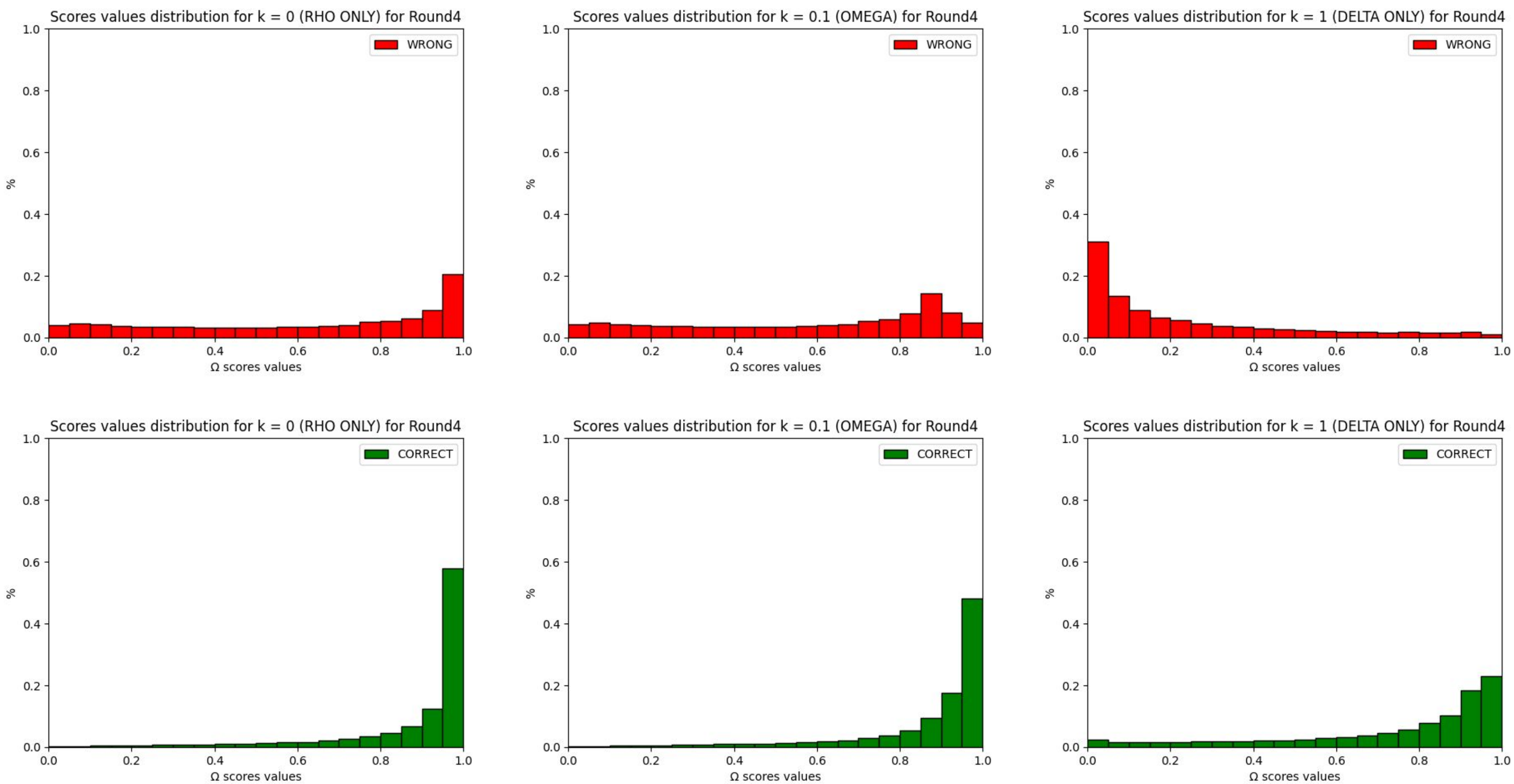


Figure A.9: Scores variations for Round4 dataset

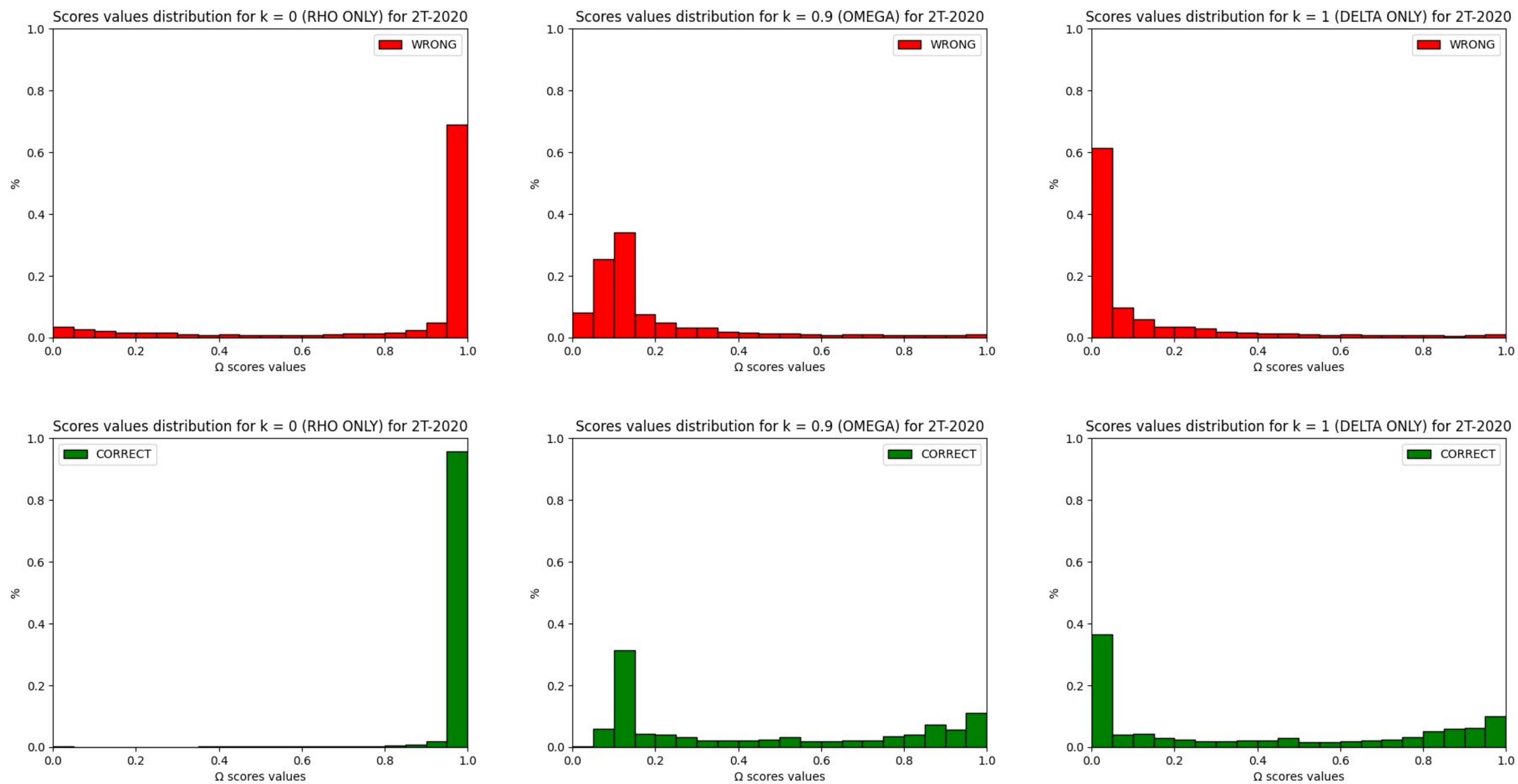


Figure A.10: Scores variations for 2T-2020 dataset

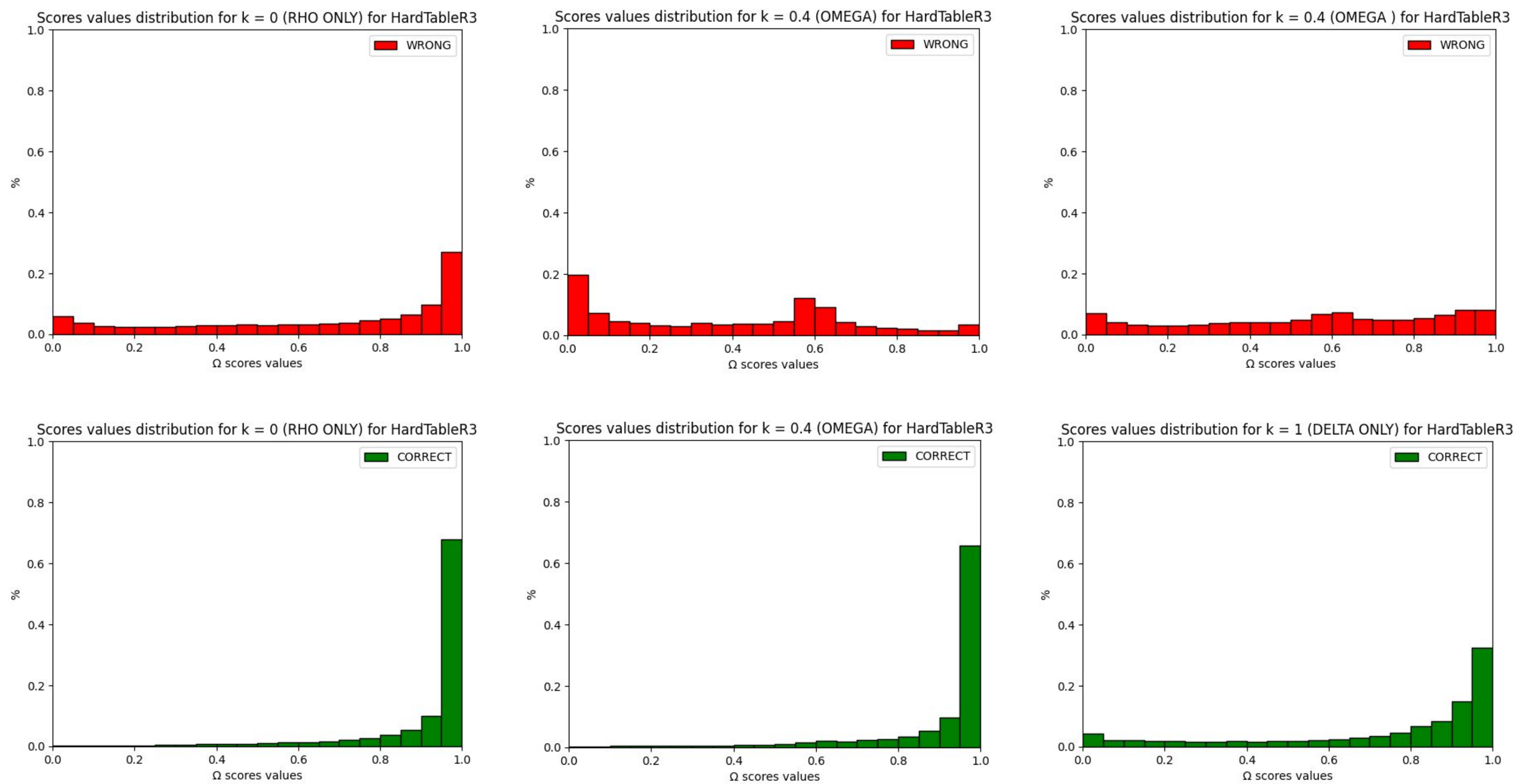


Figure A.11: Scores variations for HardTableR3 dataset

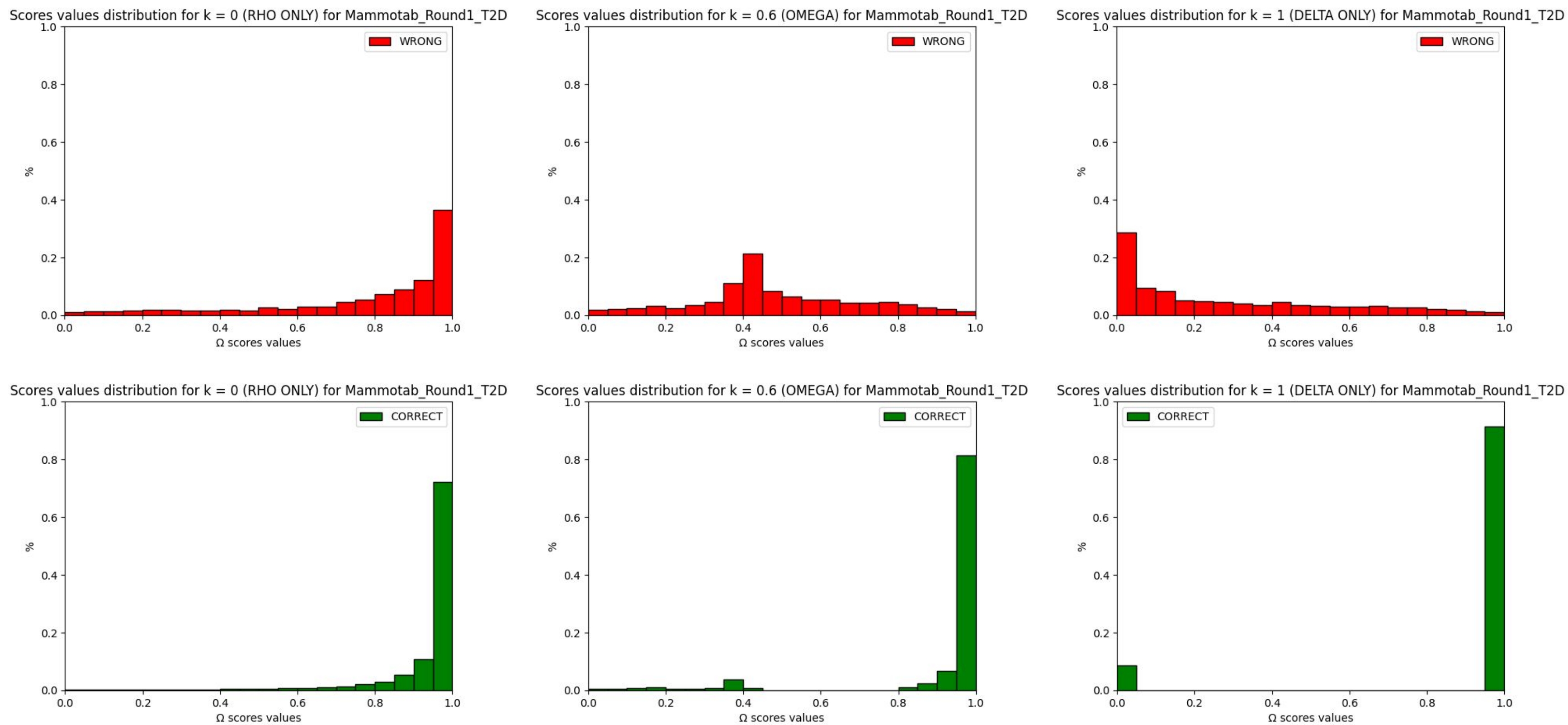


Figure A.12: Scores variations for Mammotab10K dataset using the model from Round1_T2D dataset

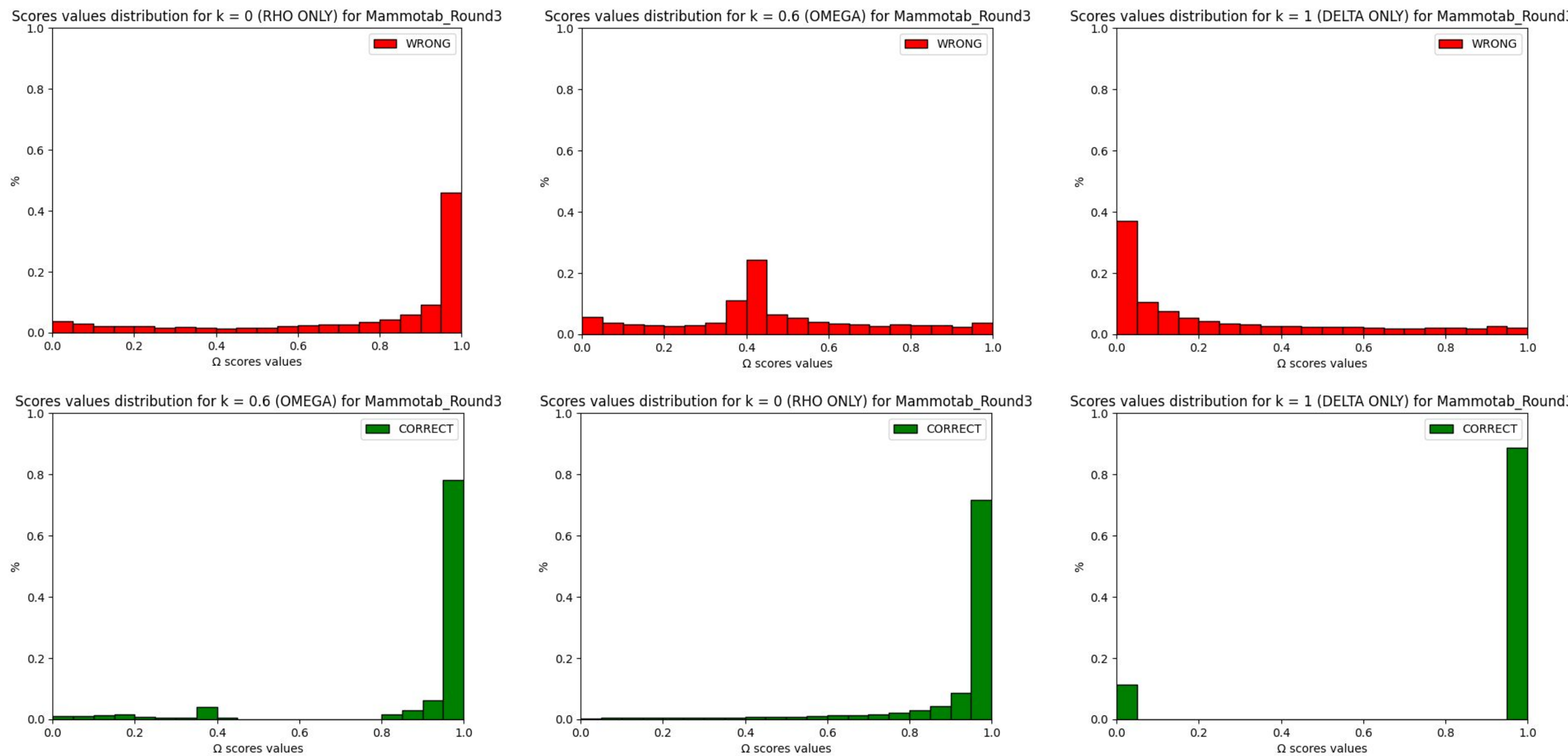


Figure A.13: Scores variations for Mammotab10K dataset using model from Round3 dataset

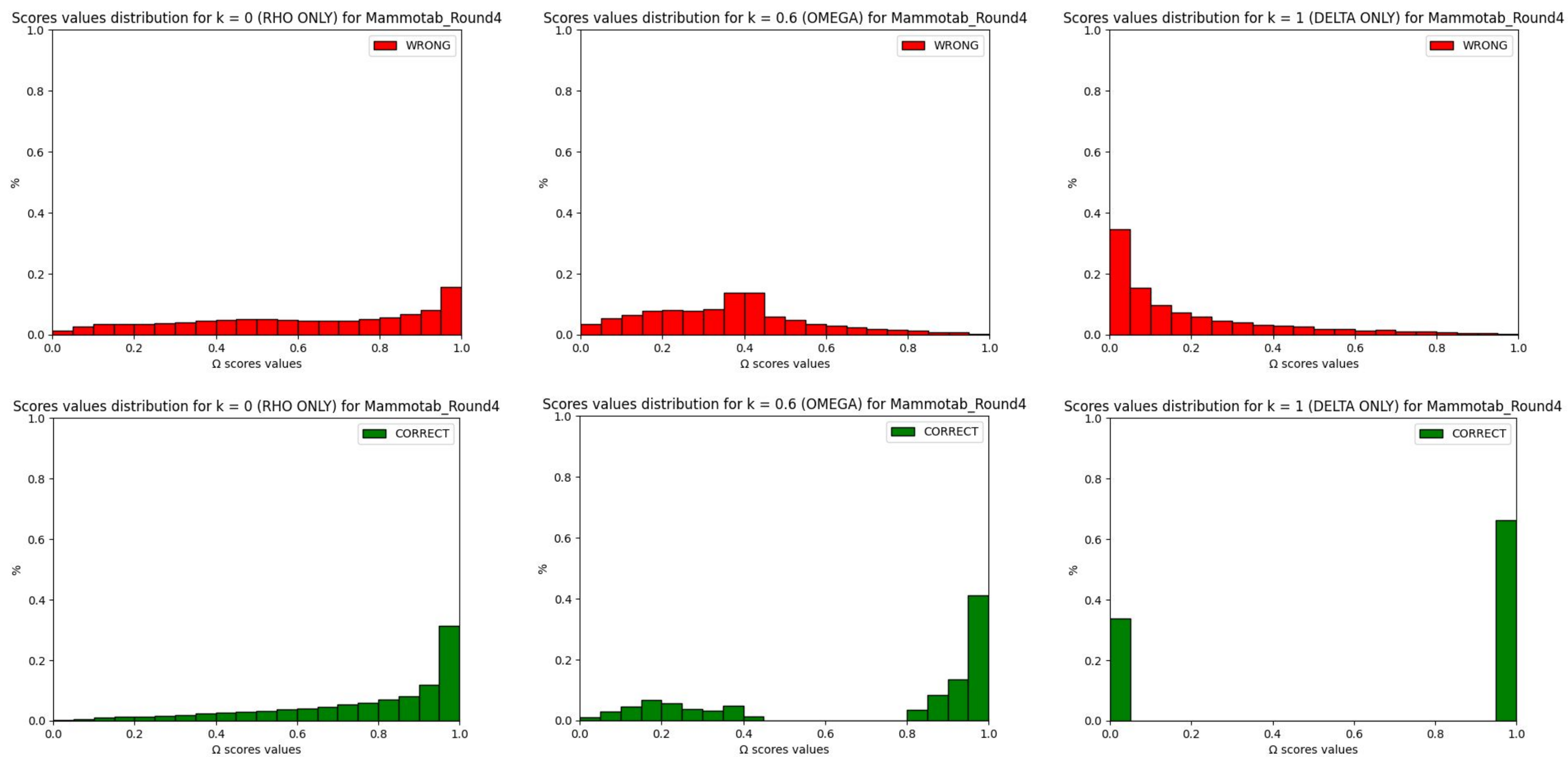


Figure A.14: Scores variations for Mammotab10K dataset using model from Round4 dataset

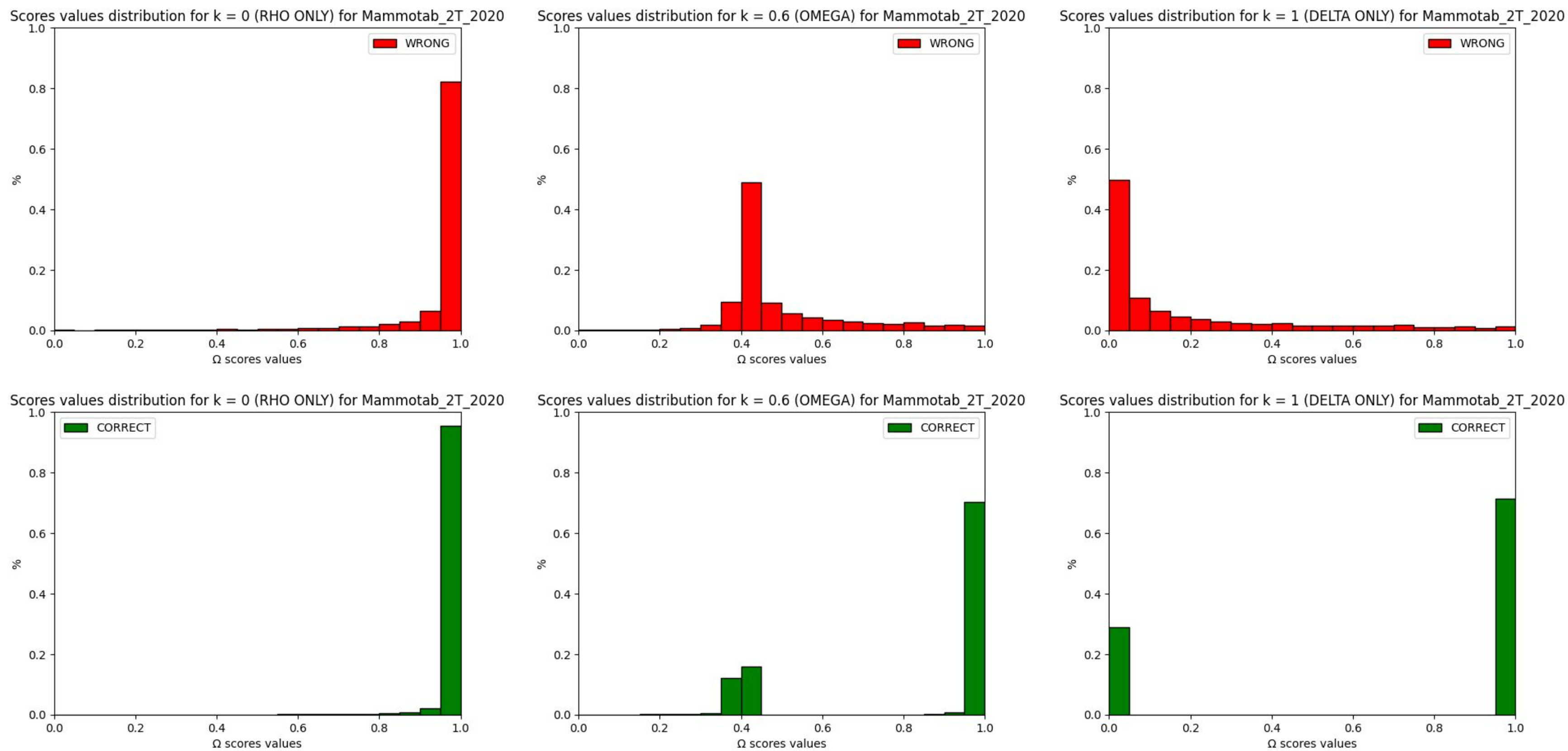


Figure A.15: Scores variations for Mammotab10K dataset using model from 2T-2020 dataset

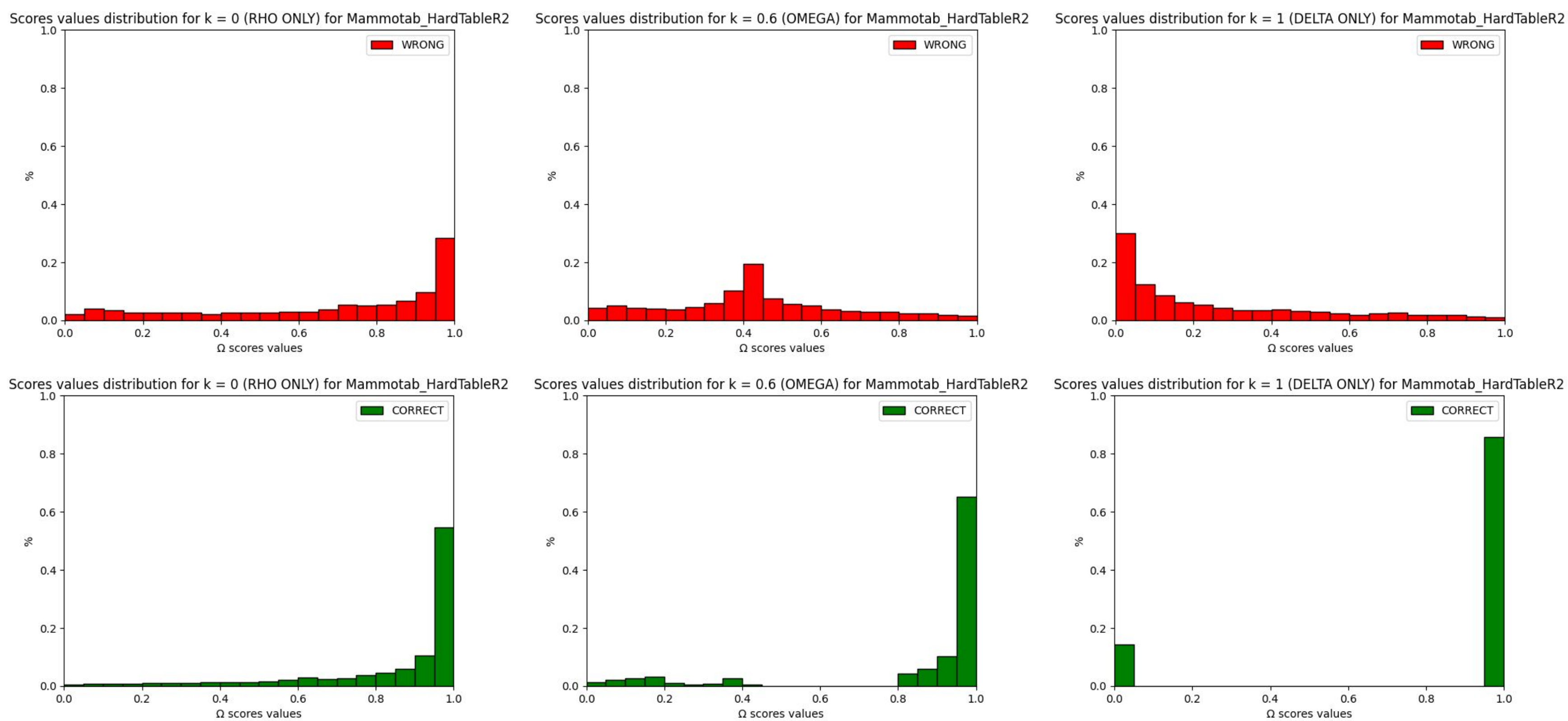


Figure A.16: Scores variations for Mammotab10K dataset using model from HardTableR2 dataset

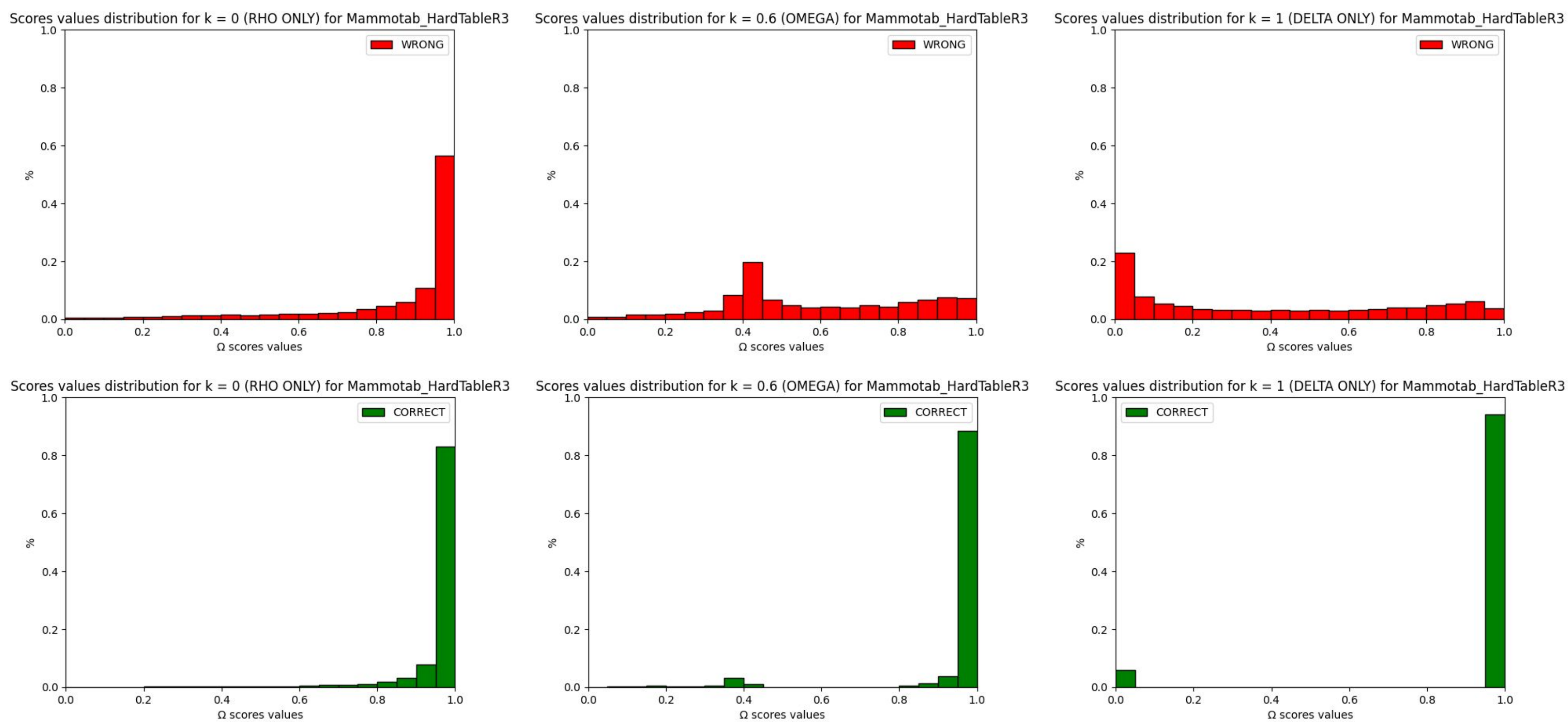


Figure A.17: Scores variations for Mammotab10K dataset using model from HardTableR3 dataset

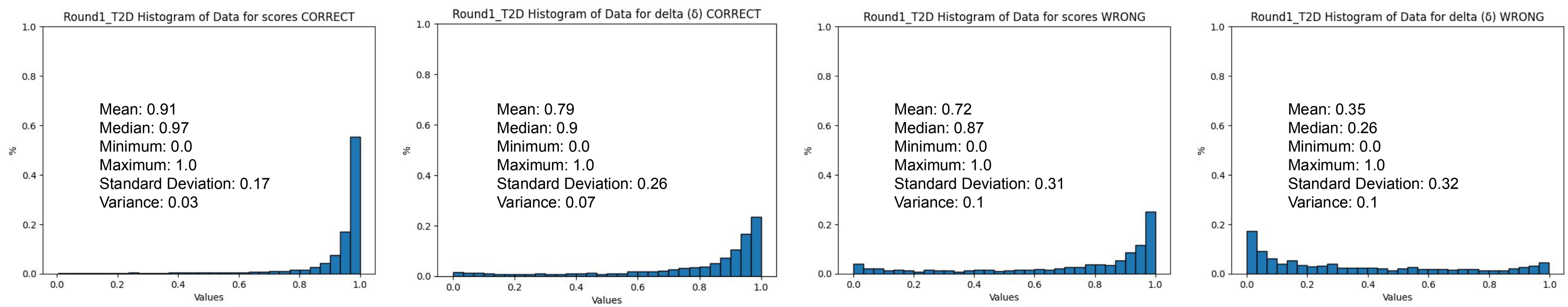


Figure A.18: Distribution of ρ and δ over *Round1_T2D dataset*

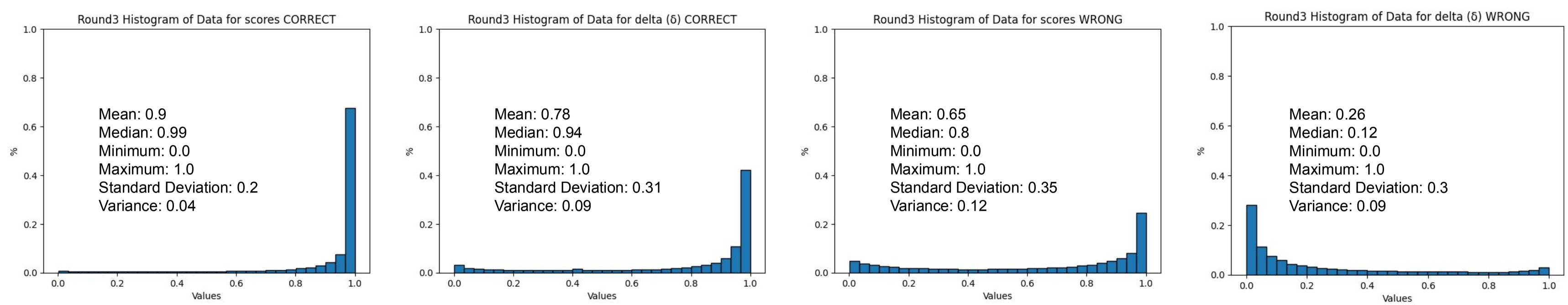


Figure A.19: Distribution of ρ and δ over *Round3*

Figure A.19 illustrates the results obtained for the Round3 dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.8 to 1. Additionally, the values of δ are primarily concentrated within the 0.8 to 1 range, indicating effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.8 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.3 range.

Figure A.20 illustrates the results obtained for the Round4 dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.8 to 1. Additionally, the values of δ are primarily concentrated within the 0.7 to 1 range, indicating effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.7 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.4 range.

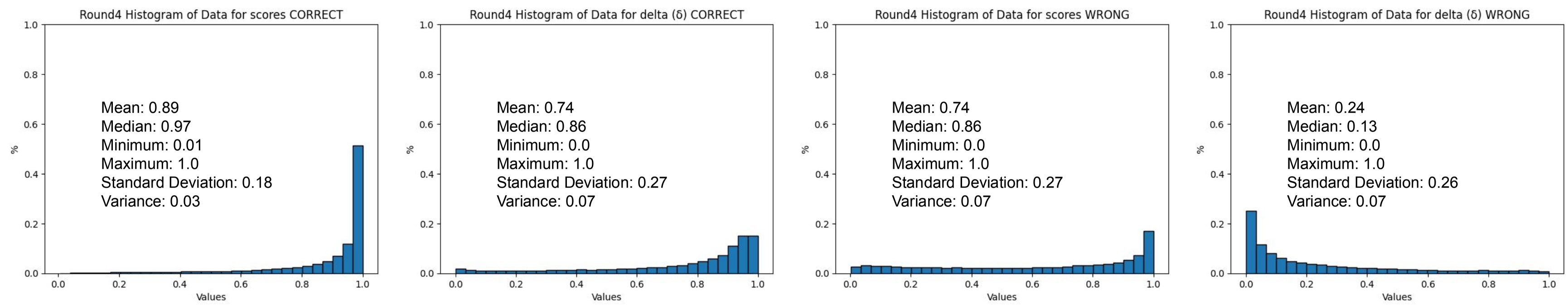


Figure A.20: Distribution of ρ and δ over *Round4*

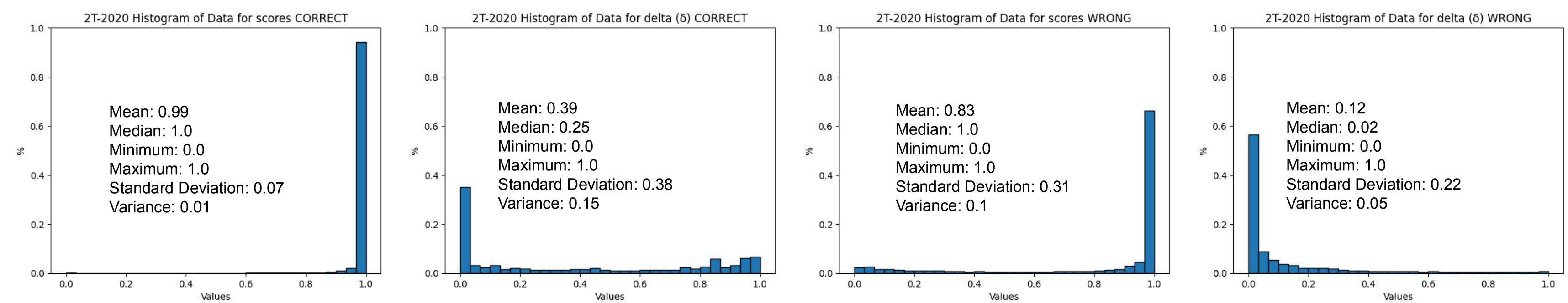


Figure A.21: Distribution of ρ and δ over *2T-2020*

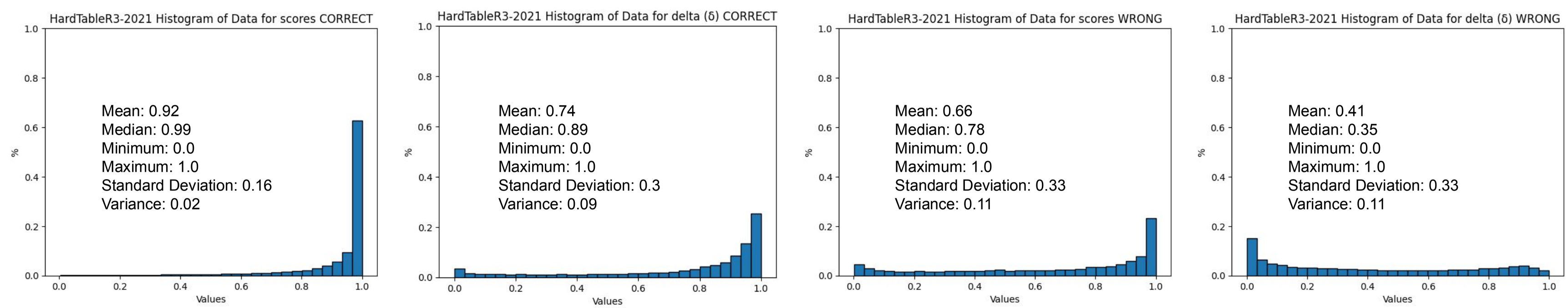


Figure A.22: Distribution of ρ and δ over *HardTableR3*

Figure A.21 illustrates the results obtained for the 2T dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.9 to 1. Additionally, the values of δ are primarily concentrated within the 0 to 0.1 range, indicating a not effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.7 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.4 range.

Figure A.22 illustrates the results obtained for the HardTableR3 dataset. The findings are highly encouraging, with the 'CORRECT' cases demonstrating a distribution of scores ranging from 0.8 to 1. Additionally, the values of δ are primarily concentrated within the 0.7 to 1 range, indicating effective disambiguation for the 'CORRECT' cases. On the other hand, the 'WRONG' cases exhibit a concentration of false positives of scores ranging from 0.8 to 1, Additionally, the values of δ are primarily concentrated within the 0 to 0.2 range.

Bibliography

[1] Nora Abdelmageed and Sirko Schindler. "Jentab: A toolkit for semantic table annotations". In: *Second International Workshop on Knowledge Graph Construction*. 2021, pp. 1–15.

[2] Claudio A Ardagna, Paolo Ceravolo, and Ernesto Damiani. "Big data analytics as-a-service: Issues and challenges". In: *2016 IEEE international conference on big data (big data)*. IEEE. 2016, pp. 3638–3644.

[3] R. Avogadro et al. "Estimating Link Confidence for Human-in-the-loop Table Annotation". In: *2023 IEEE/WIC International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Venice,Italy, 2023.

[4] Roberto Avogadro et al. "LamAPI: a Comprehensive Tool for String-based Entity Retrieval with Type-base Filters". In: *17th ISWC workshop on ontology matching (OM)*. 2022.

[5] Hiteshwar Kumar Azad and Akshay Deepak. "Query expansion techniques for information retrieval: a survey". In: *Information Processing & Management* 56.5 (2019), pp. 1698–1735.

[6] Ines Chami et al. "Low-dimensional hyperbolic knowledge graph embeddings". In: *arXiv preprint arXiv:2005.00545* (2020).

[7] Jiaoyan Chen et al. "ColNet: Embedding the Semantics of Web Tables for Column Type Prediction". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 29–36.

[8] Shuang Chen et al. "LinkingPark: An automatic semantic table interpretation system". In: *Journal of Web Semantics* 74 (2022), p. 100733.

[9] Yahui Chen. "Convolutional neural network for sentence classification". MA thesis. University of Waterloo, 2015.

[10] Michele Ciavotta et al. "Supporting semantic data enrichment at scale". In: *Technologies and Applications for Big Data Value*. Springer, 2022, pp. 19–39.

[11] Marco Cremaschi, Roberto Avogadro, and David Chieregato. "s-elBat: a Semantic Interpretation Approach for Messy taBle-s". In: *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), CEUR-WS. org* (2022).

[12] V. Cutrona et al. "Results of SemTab 2021". In: *20th International Semantic Web Conference*. Vol. 3103. CEUR Workshop Proceedings, 2022, pp. 1–12.

[13] Vincenzo Cutrona et al. "ASIA: a Tool for Assisted Semantic Interpretation and Annotation of Tabular Data". In: *Proceedings of the ISWC 2019 Satellite Tracks*. Vol. 2456. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 209–212.

[14] Vincenzo Cutrona et al. "NEST: Neural Soft Type Constraints to Improve Entity Linking in Tables." In: *SEMANTiCS*. 2021, pp. 29–43.

[15] Vincenzo Cutrona et al. "NEST: Neural Soft Type Constraints to Improve Entity Linking in Tables." In: *SEMANTiCS*. 2021, pp. 29–43.

[16] Vincenzo Cutrona et al. "Results of semtab 2021". In: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* 3103 (2022), pp. 1–12.

[17] Vincenzo Cutrona et al. "Tough Tables: Carefully Evaluating Entity Linking for Tabular Data". In: *The Semantic Web – ISWC 2020*. Cham: Springer International Publishing, 2020, pp. 328–343. ISBN: 978-3-030-62466-8.

[18] Nicola De Cao et al. "Autoregressive entity retrieval". In: *arXiv preprint arXiv:2010.00904* (2020).

[19] Xiang Deng et al. "Turl: Table understanding through representation learning". In: *ACM SIGMOD Record* 51.1 (2022), pp. 33–40.

[20] José Devezas and Sérgio Nunes. "A review of graph-based models for entity-oriented search". In: *SN Computer Science* 2.6 (2021), p. 437.

[21] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30 (2020), pp. 681–694.

[23] Emma J Gerritse, Faegheh Hasibi, and Arjen P de Vries. "Graph-embedding empowered entity retrieval". In: *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I 42*. Springer. 2020, pp. 97–110.

[24] Ian J Goodfellow et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *arXiv preprint arXiv:1312.6211* (2013).

[25] Ben Hachey et al. "Evaluating Entity Linking with Wikipedia". In: *Artificial Intelligence* 194 (2013). Artificial Intelligence, Wikipedia and Semi-Structured Resources, pp. 130–150.

[26] Kailash A Hambarde and Hugo Proenca. "Information Retrieval: Recent Advances and Beyond". In: *arXiv preprint arXiv:2301.08801* (2023).

[27] Tyler L Hayes et al. "Remind your neural network to prevent catastrophic forgetting". In: *European Conference on Computer Vision*. Springer. 2020, pp. 466–483.

[28] Nicolas Heist and Heiko Paulheim. "NASTyLinker: NIL-Aware Scalable Transformer-based Entity Linker". In: *arXiv preprint arXiv:2303.04426* (2023).

[29] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. "Elasticity in cloud computing: What it is, and what it is not". In: *10th international conference on autonomic computing (ICAC 13)*. 2013, pp. 23–27.

[30] Madelon Hulsebos et al. "Sherlock: A deep learning approach to semantic data type detection". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1500–1508.

[31] Viet-Phi Huynh et al. "From Heuristics to Language Models: A Journey Through the Universe of Semantic Table Interpretation with DAGOBAH". In: *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)* (2022).

[32] Filip Ilievski et al. "The role of knowledge in determining identity of long-tail entities". In: *Journal of Web Semantics* 61 (2020), p. 100565.

[33] Anastasiia Iurshina et al. "NILK: entity linking dataset targeting NIL-linking cases". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 4069–4073.

[34] E. Jimenez-Ruiz et al. "Results of SemTab 2020". In: *CEUR Workshop Proceedings* 2775 (2020), pp. 1–8.

[35] Ernesto Jimenez-Ruiz et al. "SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems". In: *The Semantic Web*. Cham: Springer International Publishing, 2020, pp. 514–530.

[36] Ernesto Jiménez-Ruiz et al. "Results of SemTab 2020". In: *SemTab@ISWC*. 2020.

[37] Ernesto Jiménez-Ruiz et al. "Results of semtab 2020". In: *CEUR Workshop Proceedings*. Vol. 2775. 2020, pp. 1–8.

[38] Ernesto Jiménez-Ruiz et al. "SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems". In: *The Semantic Web*. Cham: Springer International Publishing, 2020, pp. 514–530.

[39] Ernesto Jiménez-Ruiz et al. "Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems". In: *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*. Springer. 2020, pp. 514–530.

[40] Daniel Martin Katz et al. "Gpt-4 passes the bar exam". In: *Available at SSRN 4389233* (2023).

[41] Mayank Kejriwal, Craig A Knoblock, and Pedro Szekely. *Knowledge graphs: Fundamentals, techniques, and applications*. MIT Press, 2021.

[42] Ronald Kemker et al. "Measuring catastrophic forgetting in neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[43] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[44] Tuan Manh Lai, Heng Ji, and ChengXiang Zhai. “Improving candidate retrieval with entity profile generation for wikidata entity linking”. In: *arXiv preprint arXiv:2202.13404* (2022).

[45] Xiuxing Li et al. *Effective Few-Shot Named Entity Linking by Meta-Learning*. 2022.

[46] Jixiong Liu et al. “From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods”. In: *Journal of Web Semantics* (2022), p. 100761.

[47] Jinghui Lu et al. “A sentence-level hierarchical bert model for document classification with limited labelled data”. In: *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*. Springer. 2021, pp. 231–241.

[48] Mattia Marzocchi et al. “MammoTab: a giant and comprehensive dataset for Semantic Table Interpretation”. In: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab2022* (2022).

[49] Mattia Marzocchi et al. “MammoTab: a giant and comprehensive dataset for Semantic Table Interpretation”. In: *CEUR WORKSHOP PROCEEDINGS*. Vol. 3320. CEUR-WS. 2023, pp. 28–33.

[50] Eduardo Mosqueira-Rey et al. “Human-in-the-loop machine learning: A state of the art”. In: *Artificial Intelligence Review* 56.4 (2023), pp. 3005–3054.

[51] Phuc Nguyen et al. “SemTab 2021: Tabular Data Annotation with MTab Tool.” In: *SemTab@ ISWC*. 2021, pp. 92–101.

[52] Phuc Nguyen et al. “SemTab 2021: Tabular Data Annotation with MTab Tool.” In: *SemTab@ ISWC*. 2021, pp. 92–101.

[53] Peter O’Donovan et al. “An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities”. In: *Journal of big data* 2.1 (2015), pp. 1–26.

[54] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: `2303.08774 [cs.CL]`.

[55] Matteo Palmonari et al. “EW-Shopp Project: Supporting Event and Weather-Based Data Analytics and Marketing Along the Shopper Journey”. In: *Advances in Service-Oriented and Cloud Computing*. Cham: Springer International Publishing, 2020, pp. 187–191.

[56] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. “Effect of scale on catastrophic forgetting in neural networks”. In: *International Conference on Learning Representations*. 2021.

[57] Delip Rao, Paul McNamee, and Mark Dredze. “Entity Linking: Finding Extracted Entities in a Knowledge Base”. In: *Multi-source, Multilingual Information Extraction and Summarization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 93–115.

[58] Lev Ratinov and Dan Roth. “Design Challenges and Misconceptions in Named Entity Recognition”. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*. Boulder, Colorado: Association for Computational Linguistics, June 2009, pp. 147–155.

[59] Payam Refaeilzadeh, Lei Tang, Huan Liu, et al. “Cross-validation.” In: *Encyclopedia of database systems* 5 (2009), pp. 532–538.

[60] Dominique Ritze and Christian Bizer. “Matching Web Tables To DBpedia - A Feature Utility Study”. In: *20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*. Ed. by Volker Markl et al. OpenProceedings.org, 2017, pp. 210–221.

[61] Dumitru Roman et al. “Big data pipelines on the computing continuum: tapping the dark data”. In: *Computer* 55.11 (2022), pp. 74–84.

[62] Pedro Ruas and Francisco M Couto. “NILINKER: attention-based approach to NIL entity linking”. In: *Journal of Biomedical Informatics* 132 (2022), p. 104137.

[63] Christophe Sarthou-Camy et al. “DAGOBAH UI: A New Hope for Semantic Table Interpretation”. In: *European Semantic Web Conference*. Springer. 2022, pp. 107–111.

[64] Hiba Sebei, Mohamed Ali Hadj Taieb, and Mohamed Ben Aouicha. “Review of social media analytics process and big data pipeline”. In: *Social Network Analysis and Mining* 8.1 (2018), p. 30.

[65] Harald Semmelrock et al. “Reproducibility in Machine Learning-Driven Research”. In: *arXiv preprint arXiv:2307.10320* (2023).

[66] Dahlia Shehata, Negar Arabzadeh, and Charles LA Clarke. “Early stage sparse retrieval with entity linking”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 4464–4469.

[67] Wei Shen, Jianyong Wang, and Jiawei Han. “Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2015), pp. 443–460.

[68] Renat Shigapov et al. “bbw: Matching csv to wikidata via meta-lookup”. In: *CEUR Workshop Proceedings*. Vol. 2775. RWTH Aachen. 2020, pp. 17–26.

[69] Shahab Saquib Sohail et al. “Decoding ChatGPT: a taxonomy of existing research, current challenges, and possible future directions”. In: *Journal of King Saud University-Computer and Information Sciences* (2023), p. 101675.

[70] Bram Steenwinckel, Filip De Turck, and Femke Ongenae. “MAGIC: Mining an Augmented Graph using INK, starting from a CSV.” In: *SemTab@ ISWC*. 2021, pp. 68–78.

[71] Yoshihiko Suhara et al. “Annotating columns with pre-trained language models”. In: *Proceedings of the 2022 International Conference on Management of Data*. 2022, pp. 1493–1503.

[72] Ian Tenney, Dipanjan Das, and Ellie Pavlick. “BERT rediscovers the classical NLP pipeline”. In: *arXiv preprint arXiv:1905.05950* (2019).

[73] Aleena Thomas et al. “SIM-PIPE DryRunner: An approach for testing container-based big data pipelines and generating simulation data”. In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2022, pp. 1159–1164.

[74] Ruize Wang et al. “K-adapter: Infusing knowledge into pre-trained models with adapters”. In: *arXiv preprint arXiv:2002.01808* (2020).

[75] Gerhard Weikum et al. “Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases”. In: *Found. Trends Databases* 10.2-4 (2021), pp. 108–490.

[76] Ledell Wu et al. “Scalable zero-shot entity linking with dense entity retrieval”. In: *arXiv preprint arXiv:1911.03814* (2019).

[77] Xingjiao Wu et al. “A survey of human-in-the-loop for machine learning”. In: *Future Generation Computer Systems* (2022).

[78] Dan Zhang et al. “Sato: Contextual semantic type detection in tables”. In: *arXiv preprint arXiv:1911.06311* (2019).

[79] Xingxing Zhang, Furu Wei, and Ming Zhou. “HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization”. In: *arXiv preprint arXiv:1905.06566* (2019).

[80] Fangwei Zhu et al. “Learn to Not Link: Exploring NIL Prediction in Entity Linking”. In: *arXiv preprint arXiv:2305.15725* (2023).