University of Milano-Bicocca

Department of Informatics, Systems and Communication

Master's degree program in Data Science

A Systematic Analysis of Filter-Based Strategies for Entity Retrieval in Tabular Data

Supervisor: Matteo Palmonari

Co-supervisor: Federico Belotti

Master's degree thesis by:

Alessandro Belotti

ID number 896985

Academic Year 2023 - 2024

Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Entity linking, the task of associating textual mentions with corresponding entities in a knowledge base, relies on effective candidate retrieval strategies. The primary objective of a retrieval system is to ensure that the correct entity is included in the retrieved candidate set for a given mention, ideally ranking it as high as possible. Incorporating mention type information can enhance retrieval by filtering candidates to only relevant entities, improving both coverage and ranking quality.

This work introduces a structured methodology for enriching the explicit type hierarchy in Wikidata to improve type-aware retrieval. Following a comprehensive review of existing entity linking frameworks, various strategies for integrating type information into the retrieval process are proposed as baseline for this work contributions. The study begins by defining the domain of entity types in Wikidata, analyzing their hierarchical structure and relationships. Methods for extending direct types with the transitive closure and mapping them to Named Entity Recognition (NER) categories are explored.

After integrating these methodologies into the *LamAPI* retrieval system, the study evaluates the impact of type-based filtering strategies on retrieval performance, considering both hard and soft filtering approaches. The analysis focuses on the quality of the candidate set, assessing its coverage and ranking effectiveness across various tabular data domains. Furthermore, the study examines how expanding the number of retrieved candidates influences overall retrieval performance.

The results indicate that soft type-based strategies outperform both hard filtering and simple label matching. In cases where tabular data contain misspellings and typos, a soft filtering approach improves retrieval performance if the number of errors in the label remains within a manageable range. However even soft filtering struggles to retrieve the correct candidates effectively.

These findings highlight the importance of type-aware strategies for entity reconciliation and provide valuable insights for improving retrieval performance in different use cases.

ACKNOWLEDGEMENTS

This journey began with my interest in Entity Reconciliation systems and the optimization of their retrieval phase. I would like to express my gratitude to my supervisor, Professor Matteo Palmonari, for his ability to spark my curiosity and guide my research approach on this topic. I am equally grateful to the members of Professor Palmonari's research group, especially Federico Belotti, who provided invaluable support during the final stages of this thesis, particularly in the experimental part.

I am also deeply thankful for the opportunity to start this research abroad at SINTEF Digital in Oslo, one of Norway's largest research centers, which has maintained a long-standing collaboration with the University of Milan Bicocca. During my time there, I had the privilege of working alongside many professionals and learning from each of them about how a structured research project is managed. I am also grateful to professor Dumitru Roman and Till Lech, my official referrens at SINTEF.

My contribution to an existing retrieval tool provided me the opportunity to understand how state of the art technology is built and deployed.

Last but not least, I am profoundly grateful to my family for their unwavering support in both my academic pursuits and personal goals. A special thank you also goes to everyone who showed interest in my work, offered their help during challenging times, and encouraged me not to give up and always strive for the best.

Chapter 1

Introduction

In today's Information Era, we are surrounded by a vast and diverse amount of data from various sources. This immense and complex data landscape is often referred to as Big Data [3], characterized by three main attributes: Volume, Velocity, and Variety. Every second, data is generated from social media, sensors, transactions, and countless other sources. This data comes in various forms, including structured, semi-structured, and unstructured formats.

The advantages of Big Data are substantial. It enhances decision-making processes, improves customer experiences, and enables the identification of new business and market trends. However, managing and analyzing large volumes of data is a significant challenge. Often, organizations struggle to effectively utilize the data at their disposal.

There are several technical challenges associated with Big Data. Establishing a robust and reliable infrastructure is complex and expensive, typically necessitating the services of major tech companies like Amazon and Google, which offer cloud platforms as a service. One of the most critical and challenging tasks of our era is ensuring the accuracy, completeness, and consistency of this massive volume of data.

An overview of these challenges are presented by *Sandhu et Al.* [4] with a discussion of the main cloud computing frameworks available on the market.

Leveraging Big Data relies on the ability to interpret and connect data of diverse structures originating from various sources. Achieving this requires a deep understanding of the semantic knowledge inherent in the information contained within the data. This thesis focuses on reconciling textual mentions from tables with entities in a predefined knowledge base. The following sections will delve into this scenario in detail.

1.1 Contributions

Tabular data plays a crucial role in many domains, serving as a widely used format for data manipulation and exchange, particularly on the web. The ability to interpret, extract, and process tabular information is essential for knowledge-intensive applications.

Significant advancements have been made in annotating tabular data with ontologies and entities from background knowledge bases, a process known as *Semantic Table Interpretation (STI)*. Automating STI facilitates the construction of knowledge graphs, enriches datasets, and improves web-based question-answering systems.

Entity Linking (EL) plays a crucial role as part of STI. It maps named entities from unstructured text to a Knowledge Graph, thus enabling more effective data integration and enrichment.

The performances of an EL system is closely tied to the accurancy and efficiency of its two sub tasks. An EL algorithm begins with the retrieval task, where a system identifies a subset of potential Knowledge Graph entities that could correspond to a given query (mention). This step is crucial because it directly impacts the subsequent disambiguation phase, where the most appropriate entity is selected from the candidate set.

Ensuring that the correct entity is included in the set while the irrelevant ones are minimized is a key determinant of the overall EL system's performance.

One of the significant challenges in candidate retrieval is the size and complexity of Knowledge Graphs, which can contain millions or billions of entities. Without effective filtering, the system must evaluate a vast number of entities, making the process computationally expensive and prone to errors.

This work has two main objectives. The first is to present a structured methodology for enhancing the explicit type hierarchy in Wikidata, aiming to effectively extend these types. This objective is pursued through a comprehensive overview of entity types in Wikidata and a review of relevant state of the art research. After leveraging direct types in Wikidata entities, a type-aware retrieval approach has been implemented in the lamAPI system. The second objective is to assess the impact of various type-based filtering strategies on the

retrieval process, evaluating the quality of the candidate set in terms of coverage and ranking quality.

The first chapter of this thesis provides a background overview, emphasizing the relevance and importance of this use case for addressing challenges in Semantic Table Interpretation. Define the Knowledge Graph and its data model (*i.e* Wikidata data model) will let you understand how STI works and what are the challenges in Entity Linking (EL).

The second chapter reviews the state of the art EL frameworks and filtering methods, emphasizing their role in the retrieval and disambiguation phases. It provides a comprehensive overview of how type information is incorporated into existing frameworks and examines the approaches used by leading EL systems.

The third chapter explores the first research question by outlining a methodology for enriching Wikidata's type hierarchy. It introduces various techniques for extending types and presents innovative strategies to improve the quality of the candidate set.

The fourth chapter is dedicated to the second research question. This section presents the results of all experiments conducted, with a focus on evaluating the effectiveness of various filtering strategies in terms of coverage and ranking performance with different tabular data.

Through these contributions, this thesis aims to provide a comprehensive evaluation of novel and effective filtering strategies for improving candidate retrieval processes.

1.2 Background

Nowdays most of the information are created, organized and shared in table format. Web tables and spreadsheets are just some examples of how different domains can model data.

- Web tables: in 2008 14.1 billion HTML tables were extracted, and it was found that 154 million were high-quality tables (1.1%). In the Common Crawl 2015 repository, there are 233 million content tables[1];
- Wikipedia tables: the 2022 English snapshot of Wikipedia contains 2 803 424 tables from 21 149 260 articles [5];
- Spreadsheets: there are 750 million to 2 billion people in the world who use either Google Sheets or Microsoft Excel[2]

Despite the simplicity of tabular data, understanding and interpreting their meaning is often challenging. This because it requires the concept of Human Knowledge, a collective understanding and awareness accumulated by humanity through experience, education, and research.

The most popular format for knowledge representation is called Knowledge Graph.

A knowledge graph (KG) is a data structure that organizes and interlinks information, they are a machine-understandable representation of entities (the nodes) and the relationships between them (the edges connecting nodes). Each connection usually has additional attributes and properties.

The formal definition of a **knowledge graph (KG)** describes it as a directed graph $G = (V, E)$, where V denotes entities and E denotes directed edges. Each entity $u \in V$ is uniquely identified and may possess associated attributes or properties. A directed edge $(u_1, r, u_2) \in E$ signifies a relationship r from entity u_1 to u_2, which may include additional attributes or properties.

[1] commoncrawl.org

[2] askwonder.com/research/number-google-sheets-users-worldwide-eoskdoxav

For example in the figure 2.2 the triple (Steve Jobs, Former CEO of, Apple Inc.) is a direct edge which depict the relation between the entities Steve Jobs and Apple Inc.

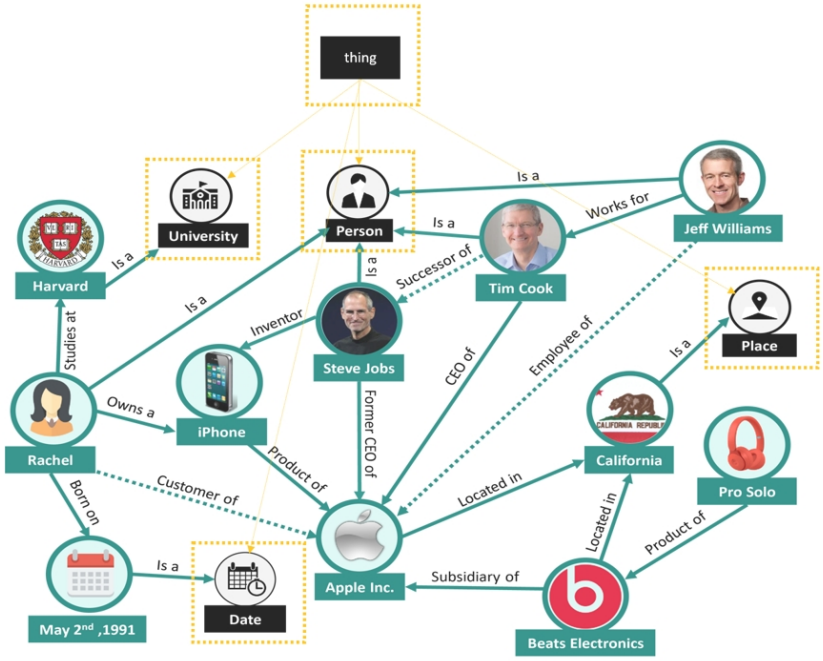

Figure 1.1: Example of knowledge graph.

Often, different knowledge graphs overlap in certain domains, making it crucial to ensure standardization across schemas to achieve interoperability between various knowledge bases. This need for standardization highlights the purpose of ontologies.

An ontology $\mathcal{O}$ serves as a structured framework for organizing information. It defines a set of concepts, categories, and relationships within a particular domain, providing a schema that can be used to model the domain's knowledge.

By standardizing how information is represented, ontologies enable different systems to communicate and integrate data more effectively, ensuring consistency and interoperability across diverse knowledge bases.

For instance, if an ontology defines that all *Mammals* are *Animals* and *Dogs* are *Mammals*, then a knowledge graph can automatically infer that all *Dogs* are *Animals*.

Examples of common ontologies are:

- **Schema.org**: used for structuring web content, providing a common vocabulary for

entities like people, events, and products, enabling search engines to better understand and display web data.

- **FOAF (Friend of a Friend)**: used to describe social networks and user profiles on the web, allowing for interoperability and data exchange between different social networking sites.

- **Gene Ontology (GO)**: used in the fields of biology and genomics to categorize gene functions, biological processes, and cellular components, facilitating consistent annotation and analysis of gene products.

The physical structure of a Knowledge Graph is defined using a specific representation model. The most common standard model is called **RDF (Resource Description Framework)**. It structures data in the form of triples (subject, predicate, object), enabling data to be interconnected and interrelated across different domains.

```
<http://example.org/book1> <http://purl.org/dc/elements/1.1/title> "A Great Book".
```

Since 2007, with the founding of DBpedia and Freebase, numerous organizations have developed their own graph-based knowledge repositories. Examples include the Google Knowledge Graph, Amazon's Product Graph, and WordNet. Among these, Wikidata stands out as prominent player central to this thesis. The primary distinctions between these knowledge graphs typically lie in their schemas and domains.

Most of the frameworks and scientific papers are based on Wikidata, also in this work the reference will be the Wikidata Knowledge Base. Wikidata is a free and collaborative project that supports Wikipedia and other Wikimedia projects. It's represented in RDF format and as described in the scientific literature [4], its domain is incredibly vast, covering subjects ranging from history and science to geography. Wikidata comprises millions of items, each identified by a unique identifier, and includes a wide range of properties and values. This extensive and interconnected dataset makes Wikidata an invaluable resource for various applications.

For a complete understanding of this thesis it's important to define how the data model is defined in Wikidata.

An entity could be both an *item* or a *property*. For instance in the figure 1.2 is shown the item *Neckar*, which is an instance of a more general item called *river*:

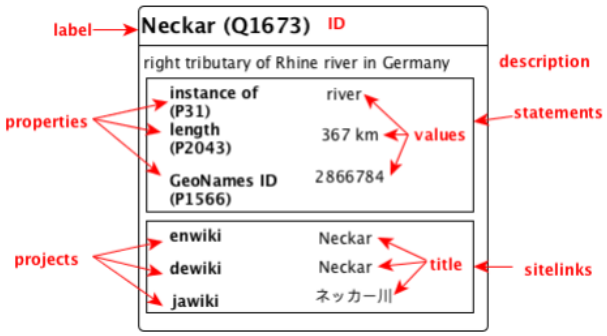

Figure 1.2: Wikidata data model.

All items are identified by numerical identifiers prefixed with a Q (e.g., Q1673 for *Neckar*), whereas properties are identified by numerical identifiers prefixed with P (e.g., P31 for *instance of*). Properties connect items (Q) to values (V), forming a pair known as a statement (P, V). The property in a statement defines the type or class of the value.

The Wikidata hierarchy is based on the knowledge of which item is an instance of which class, from the river in the figure 1.2 is possible to define the following hierarchy:

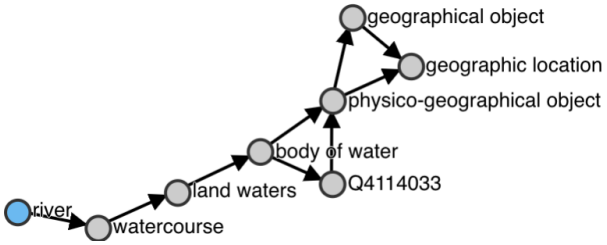

Figure 1.3: *river* class hierarchy.

For example, consider the statement Q1673:P31:Q4022 (*Neckar* is an instance of *river*), in which Q4022 can be seen as a class.

The property *subclass of* is transitive, so *Neckar* is also a subclass of *watercourse*. This means that there is no statement that directly specify *Neckar* to be a *geographic location*, the only way is to extract the transitive hull of the items.

Having a unique identifier for each entity means that is possible to match data in table format against a background Knowledge Graph. This is called Semantic Table Interpretation (STI) and it has emerged as a key approach for identifying and reconciling common entities across datasets and knowledge bases.

1.3 Table interpretation

The interoperability between the huge amount of available tables and datasets is possible and the answer is identifying the same entities across disparate datasets. As seen previously each entity will be uniquely identified in a knowledge base and the identification process is known as the table-to-Knowledge Graph (KG) matching problem.

The recent research community has paid more and more attention to this topic defining a process called *Semantic Table Interpretation* (STI). This is a key step to enrich data and construct and extend Knowledge Graphs (KGs) from semi-structured data. The knowledge of entities with their types and properties allows to do many operations as illustrated in the following figure:

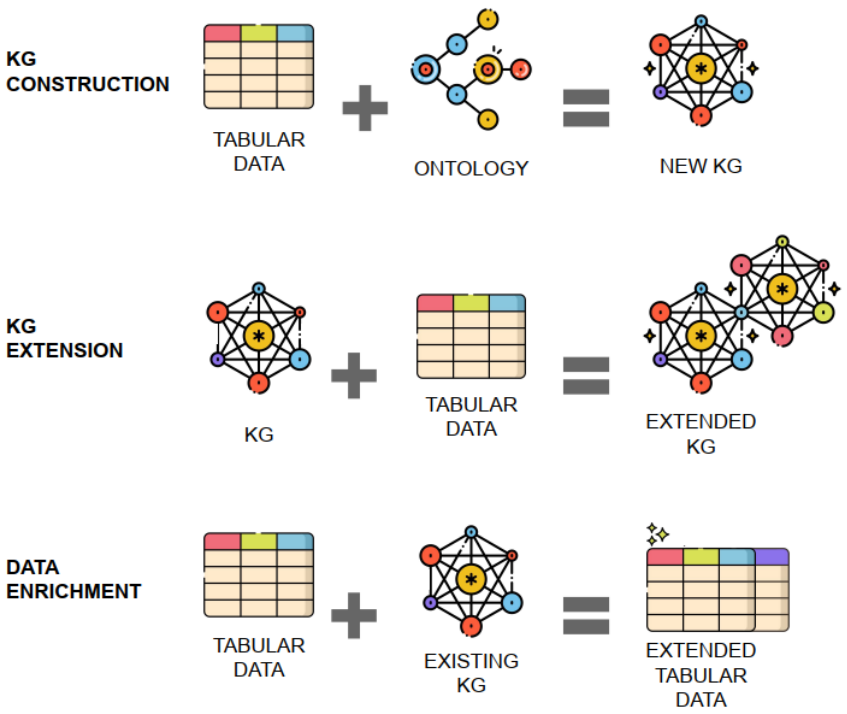

Figure 1.4: Examples of applications supported by STI.

With *KG construction* data are transformed into a graph format with the schema (the so called Ontology of the KG) of the reference KG. STI can also support the *KG extension* processes by adding to the graph only the new entities and triples represented in the table. STI annotations can be useful also in *data enrichment* tasks, where a table is interpreted (or reconciliated) using a KG. The extension is possible thanks to the information taken from the KG end/or from other tables.

Chapter 2

Entity linking in tabular data and the problem of type-based filtering

This chapter aims to introduce the central problem addressed in this thesis, beginning with a formal definition of Semantic Table Interpretation and an explanation of how the Entity Linking task operates within the context of tabular data.

The chapter will present key state-of-the-art algorithms, with a particular emphasis on the role of entity types in the interpretation process. Notably, several studies have proposed using type filtering during the retrieval phase, while others apply it during the disambiguation process. Through this discussion, the chapter provides a comprehensive overview of the current scientific landscape, highlighting the concept of type filtering as a critical component of entity reconciliation.

Cremaschi M. et Al. [6] has proposed a recent analysis with the aim to present an extensive overview of STI tasks.

The STI process consists of producing a *semantically annotated table* given a reference Knowledge Graph and a relational table. The following example has been proposed in the paper [6]:

Name	Coordinates	Height	Range
Le Mout Blanc	45° 49′ 57″ N, 06° 51′ 52″ E	4808	M. Blanc massif
Hohtälli	45° 59′ 20″ N, 7° 48′ 10″ E	3275	Pennine Alps
Monte Cervino	45° 58′ 35″ N, 07° 39′ 31″ E	4478	Pennine Alps

Input Data

Figure 2.1: Example of a well-formed relational table.

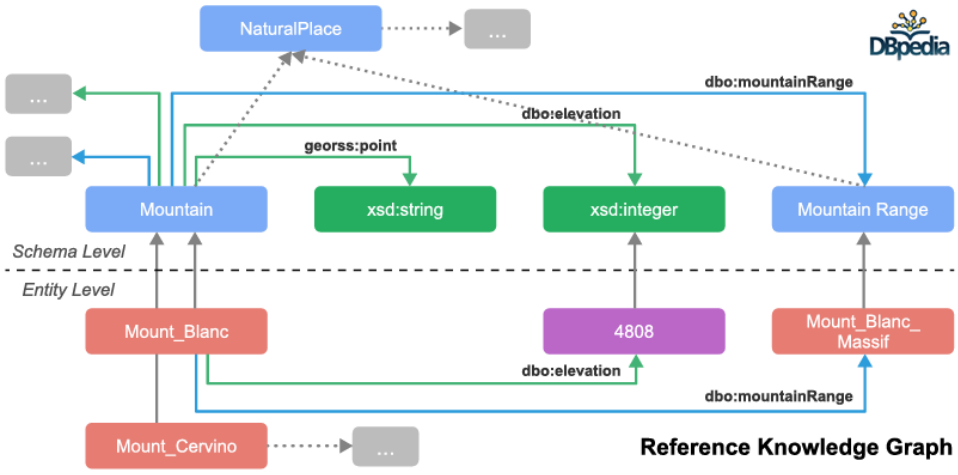

Figure 2.2: A sample of a Knowledge Graph.

The annotation is composed by the following tasks:

- **Column-Type Annotation (CTA)**: provides information about the types of entities extracted for each column. In the example the column *Name* in the figure 2.1 is annotated with the type `Mountain` and the column *Height* with datatype `xsd:integer`.

- **Cell-Entity Annotation (CEA)**: each cell of the table is annotated with an entity of the KG or with NIL if no entity corresponds. The cell *Le Mout Blanc* is annotated with `Mount_Blanc` and *Hohtalli* with NIL because the KG in figure 2.2 has no related entity yet.

- **Columns-Property Annotation (CPA)**: identifies the predicates or relationships between pairs of columns. For example the pairs composed by the columns *Name* and *Height* is annotated with `dbo:elevation`.

The result of the annotation process discussed in the paper is the following:

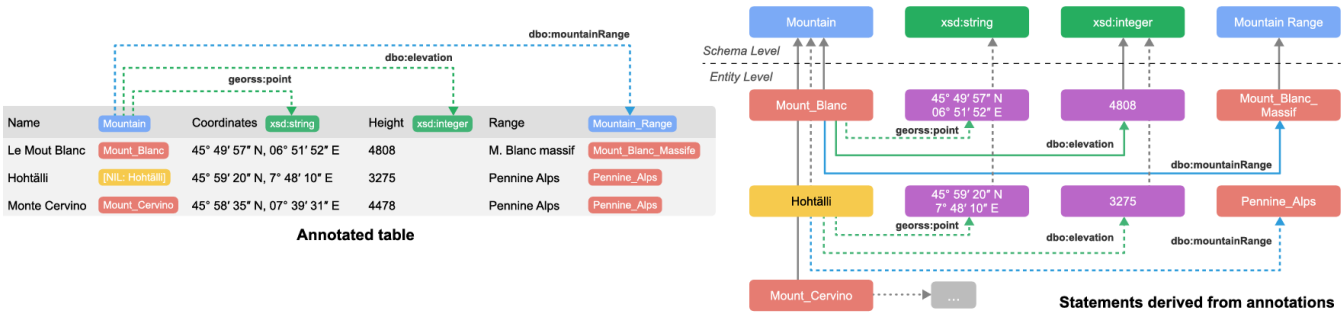

Figure 2.3: Example of an annotated table.

STI tasks such as CEA, CTA, CPA are relevant in the AI research for tabular data. The interest for this thesis is CEA, which is a specific **Entity Linking** (EL) task for tabular data.

2.1 Entity linking in tabular data

The task of linking named entities mentioned in the text to their corresponding entities in a KG is also called *entity resolution*.

EL data pipeline involves several key tasks, following the literature overview [[6], [7]] it's possible to define the following main steps:

1. **Mention Detection**: Identify and extract mentions from tabular data. Detecting mentions can involve various techniques such as Named Entity Recognition (NER) or pattern matching algorithms.

2. **Candidate Generation**: For each extracted mention, generate a list of potential matching entities from a knowledge base (KB). It is also known as *lookup*. The process may utilise techniques as entity recognition, information retrieval or ML models for the generation. The most used techniques used for generating candidates can be grouped into:

    - **Custom Index**: when building a custom index, there is flexibility in defining mappings, analysers, and other configurations based on specific needs

- **External Lookup Services**: it refers to using a separate service or system to perform lookup or queries for retrieving specific information or data. The most common examples are DBpedia Spotlight and Wikidata Lookup Service.
- **Hybrid**: it uses both custom index and external lookup services

3. **Entity Disambiguation**: The process of resolving ambiguous mentions to entities. Infact there can be ambiguity if the same name refers to multiple entities in a KB. This is typically done using a combination of heuristic rules and machine learning algorithms that consider factors such as context, relevance, and similarity.

Let's take the example of the the sentence "Barack Obama was born in Hawaii". The named entity "Barack Obama" can be linked to the corresponding entry in a KG, such as `Barack Obama (Q76)` in Wikidata.

In a formal definition given a KG containing a set of entities E and a collection of named-entity mentions M, the goal of Entity Linking (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:

1. **Entity Retrieval (ER)**: 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 ER module.

2.2 State of the art approaches to EL for tabular data.

Semantic Interpretation has a huge research interests, many surveys and analysis has been conducted in order to create new algorithms and frameworks able to outperform the old ones. Here some of the most updated approaches will be presented with the focus on Cell Entity Annotation, also called Entity Linking in tabular data context.

2.2.1 Main EL approaches

Over the years, there has been a surge in interest in data-driven approaches based on deep learning that have increasingly been combined with heuristic-based ones. In the last period, the advent of Large Language Models (LLMs) has led to a new category of approaches for table annotation.

Many approaches for ED has been proposed and they are grouped in different approaches: those based on heuristics, ML, probabilistic approaches, and based on LLM. Based on the analysis presented in the paper *Belotti et Al.* [8] it's possible to define the following STI approaches:

TableLlama [9] use an LLM model to solve tasks as CEA. In particular the disambiguation phase is done querying the LLM in order to link a particular mention found in the table against a small set of candidates.

TURL [10] uses a pre-trained TinyBERT model to set up a Transformer encoder that understands table structures. It fine-tunes the model to create context-aware representations for each table cell. Then, it calculates matching scores between the representations of knowledge graph (KG) candidates and the cell embeddings using a linear function, converting these scores into probabilities over the candidate entities.

Dagobah [11] is an heuristic-based algorithm based on a multi-step pipeline. The candidate retrieval phase is done by using Elasticsearch. The disambiguation is done by combining a relevance score based on the similarity with the CPA and CTA information.

Alligator [12] is based on a feed-forward neural network trained on a specific dataset called Gold Standard. It's focus on entity disambiguation: after having retrieved the

candidate set it ranks them using two neural networks.

The output of the model is a *confidence score* which assesses the likelihood that a certain candidate (for example **Q3512046**) is the correct entity to be linked to the mention (Jurassic World).

The input is a concatenation of three features vectors defined as follow:

- features for a mention (features such as string length and token count)
- features for a mention - entity (composite of mention and entity, encapsulating features like Levenshtein distance and Jaccard similarity coefficient)
- features for a candidate entity (focuses solely on the entity and includes features such as popularity and the token count of the entity's label)

A first ML model will identify the weigths to assign to each feature vector, learning them from the correct candidates observed during the training phase. The following picture is an example:

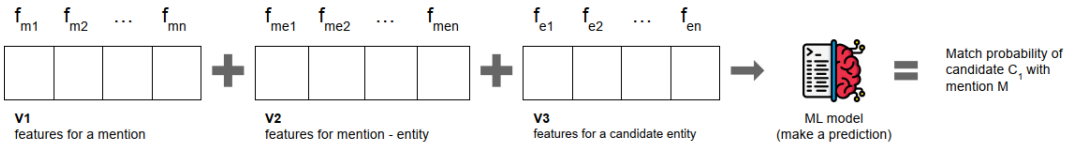

Figure 2.4: Feature vectors used in Alligator.

A second ML model enhances the scoring process by incorporating contextual information from CTA abd CPA, generating an improved candidate ranking.

The disambiguation phase is implemented as a ranking algorithm which prioritize and select the best candidate for each mention. For each candidate is calculated a confidence score ρ which will be used to compute an uncertainty measure ω.

Based on the analysis of this algorithms it's possible to say that the CEA, CTA, and CPA tasks are interlinked and can mutually inform each other, especially in the final ranking phase of the candidates for the disambiguation. For example, entity-level annotations

can suggest or provide evidence for type and property-level annotations, while type-level annotations may help disambiguation for entity-level annotations.

2.2.2 Type filtering in EL

The entity type can be an important information for supporting Entity Linking pipeline. However, type information explicitly stored in KGs has to be in line with the types on the mention table.

An example highlighting the importance of correct type filtering involves the entity *Albert Einstein*, which is classified solely as a *Scientist* in DBpedia. However, there are scenarios where the column type associated with the entity does not perfectly align with its classification in the knowledge graph. For instance, if the entity appears in a column of philosophers, applying a strict filter based on the column type *Philosopher* would result in the exclusion of *Albert Einstein* as a candidate, even though he is a relevant figure. This demonstrates the potential pitfalls of using rigid type constraints without considering broader type contexts.

A significant contribution by *Efthymiou et al. [13]*, who proposed two approaches, one of them called FactBase Lookup uses the type information inside the linking pipeline.

The **FactBase lookup** method uses a knowledge base (KB) to find corresponding entities by scanning all the cells of an entity column using the entity types for candidate filtering. The process begins with a scan that identifies *frequent direct types* and relationships between entities. After this initial scan, it performs stricter lookup using refined data, and for ambiguous results, a second phase introduces more relaxed criteria, such as editing distance tolerance. However considering only direct types could be a limitation.

In the scientific literature there are different discussions about how filters can be relaxed in order to better handling entity types. A notable contribution by *Cutrona et al. (NEST)* [14], in their work the authors propose two distinct strategies for incorporating soft type constraints into the system:

- *type enrichment for filtering by type:* combines direct types of the mention (used as

in FactBase algorithm) with types predicted by a neural model to refine type-based filtering.

- *type enrichment for ranking by distributed entity representations similarity:* evaluating the similarity between candidate entities in a same column helps entity disambiguation. The idea is based on concatenating the vector of the mention with its type creating a vector space representation able to be similarly closer to the others similar types (enhancing the EmbeddingsOnGraph algorithm proposed in *Efthymiou et al. [13]*).

Results suggest that applying a soft constraint principle to enhance strict explicit entity types significantly improves reconciliation strategies.

Entity type is an important information, also the *Alligator* tool [12] combine the candidates types during the final ranking phase. The ML model used during the disambiguation phase has many features including information related to types associated with the candidates. The *cta_t1* to *cta_t5* features represent the top five most frequent types associated with a candidate, derived from a dictionary of type frequencies. The most frequent type is assigned to cta_t1, the second-most to cta_t2, and so on, up to cta_t5.

Many of the approaches discussed in this section leverage entity types as contextual information to enhance ranking or disambiguate candidates. This thesis proposes incorporating type information earlier in the retrieval process, specifically at the Entity Retrieval (ER) stage. The following section provides a brief analysis of the current scientific advancements in type-aware retrieval.

2.2.3 Type-aware Retrieval

The performance of an Candidate Retrieval system (also called Entity Retrieval - ER system) is closely tied to the methods employed for indexing and storing the knowledge graph. Effective Retrieval Systems must strike a balance between meeting minimum requirements and managing trade-offs in efficiency and accuracy.

Key aspects to consider when this kind of system are developed are:

- **Scalability:** Approaches such as Distributed or Hierarchical Indexing to efficiently manage large datasets.
- **Query Performance:** Techniques like Path Indexing (storing frequently accessed paths or patterns) or Adaptive Indexing (reducing unnecessary indexing overhead) to optimize query responses.
- **Dynamic Updates:** Incremental Indexing to minimize the time and computational resources needed for maintaining the index.

Many of the aforementioned tools has been tested in recent papers ([8], [6]) using a state of the art retrieval tool called *lamAPI*. The architecture and techniques used in this tool are presented by *R. Avogadro et Al.* [15].

LamAPI is an ER system based on a complex string-matching algorithm which leverages Elasticsearch for indexing and MongoDB for store the intermediate processing results generated after indexing. Leveraging type information in this system is object of the thesis.

This section will explore the architectural characteristics and interactions of these components in greater detail.

Elasticsearch is highly effective for scenarios that demand both scalability and query performance, thanks to its hybrid indexing approach which integrates:

- **Inverted Index:** A data structure that maps each term in a document collection to the documents containing it (the so called *posting list*).
- **Term Frequency (TF):** The frequency of a term's appearance in a document, indicating the relevance of the document for that term.

The figure 2.5 illustrates the combined use of these two indexing methods to enhance retrieval performance:

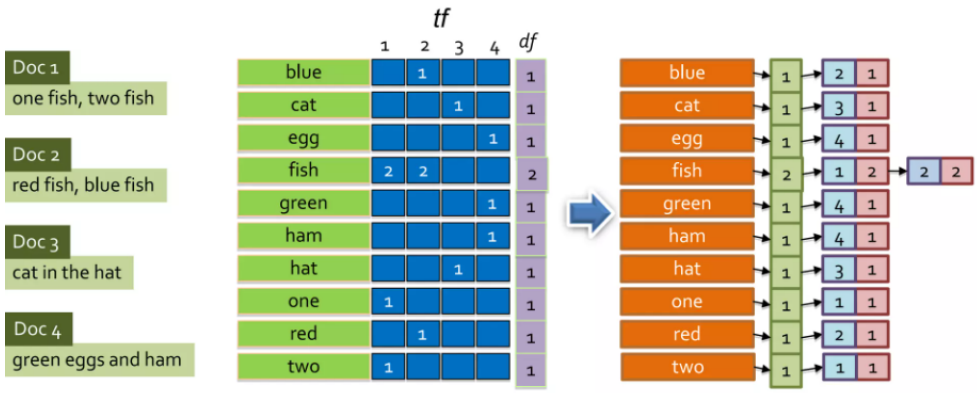

Figure 2.5: Combining TF and Inverted Index.

The green column on the right side table indicates the *document frequency* (number of documents in which a specific term appears). The blue number is the document id which is concatenated to the *term frequency* in red, which indicates how many times a specific term appears within a single document.

MongoDB instead is a NoSQL database designed for flexibility, scalability, and performance. It stores data in BSON (Binary JSON) format, allowing dynamic schemas where documents in a collection can have varied structures.

This data storage architecture ensure a flexible and scalable data model.

LamAPI has different endpoints which allow full-text search over different KG and data retrieve about entities. The following list wants to explain the available services:

- **labels:** given a JSON array, it returns a list of *labels* and *aliases* for each entity.
- **literals:** given a JSON array, it returns the list of *literals* classified as *datetime*, *number*, or *string* for each entity.
- **objects:** given a JSON array, it returns a list of *objects* for each entity.
- **predicates:** given a JSON array, it returns a list of *predicates* between each pair of entities (*subject* and *object*).
- **sameas:** given a JSON array, it returns the associated entities in *Wikipedia* and *DBpedia*.

- **types:** given a JSON array, it returns the associated *types* for each entity.
- **lookup:** given a string as input, the endpoint performs a search in the specified *Knowledge Graph*.
- **column-analysis:** given a JSON array as input composed of a set of array of strings (cell content), the endpoint calculates, for each array, if the content represents *named-entities* or *literals*.

Figure 2.6 illustrates the detailed workflow involved in lamAPI. The process starts with handling a substantial Wikidata dump file, which, when compressed, is around 77GB in size. This file is available for download at `https://dumps.wikimedia.org/wikidatawiki/entities/`.

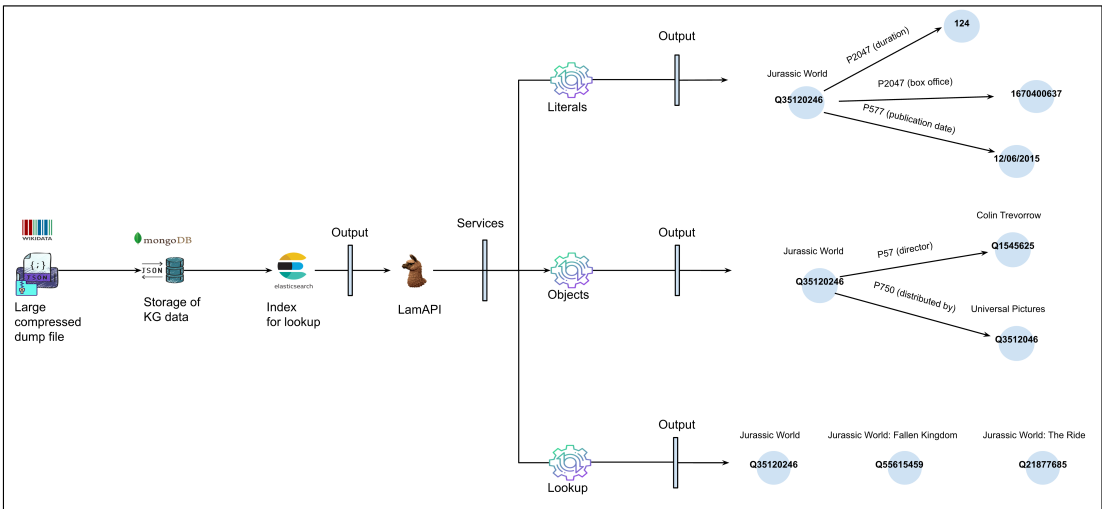

Figure 2.6: Data pipeline used by lamAPI.

The lamAPI tool belongs to the class of candidate generator called ***custom index*** (from the classification in [6]). It refers to building a specialized index for specific requirements or use cases, mainly based on Elasticsearch (like lamAPI).

A similar retrieval strategy has been used in an extended version of MTab [16] focuses on annotating cells to Wikidata entities. It starts by downloading and extracting a Wikidata dump to build an index using hash tables. The lookup process (also called ER) is then performed using a fuzzy search (one of the key difference from the original version of MTab).

The result is a ranking list of entities based on edit distance scores. The candidate type is integrated into the retrieval process and it's crucial for the main tasks of the annotation:

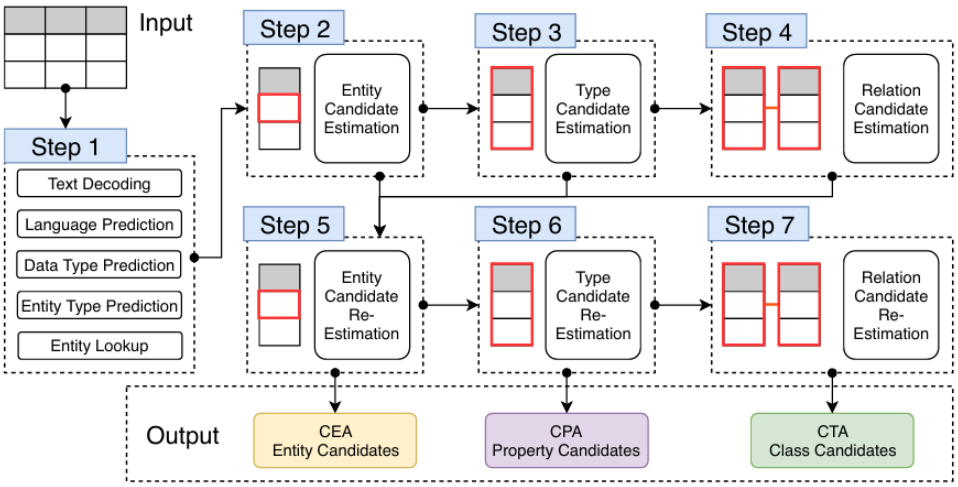

Figure 2.7: The design of the original MTab tool [1].

Many strategies already implement type-aware retrieval by using the CTA task as additional information for the candidate retrieval.

2.2.4 Performances evaluation

Based on the final evaluations in the survey *Belotti er Al.* [8] many tests have been conducted based on various scenarios. Approaches like **Alligator**, despite its limited number of parameters, can still be a viable option when resources and budget are limited.

An international challenge with increasing interest over time is called *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* (SemTab), becoming the most popular benchmark in STI topic. Many of the mentioned frameworks have been used in this challenge. **DAGOBAH** has been the winner of various rounds of the SemTab challenge from 2020 until 2022. **Mtab** is indeed ranked as first place for CEA task in SemTab 2020.

Both *Alligator* and *DAGOBAH* frameworks implements a sort of type-based filtering during the disambiguation phase or the finale scoring step. The aforementioned approaches used in **factBase** and **MTab** instead propose the entity type awarness during the candidates estimation phase.

Based on this comprehensive analysis the goal of this work is to propose a new approach for the candidate estimation able to leverage not only the direct Wikidata types but also their hierarchy extension.

2.3 Methodology for a type level hierarchy definition

Incorporating type context in Entity Linking is essential, as discussed in previous sections. However, using direct entity types for filtering can be restrictive. Instead, leveraging the hierarchical structure of the knowledge base allows for a more flexible and accurate candidate selection process.

By structuring the knowledge graph at a higher level, broader entity types can be utilized to refine filtering constraints more effectively. This approach prevents excessive narrowing of candidates while maintaining relevance.

The next section introduces an entity classification method that forms the foundation for one of the filter strategy of this thesis. By mapping entities to broader categories, this method establishes type-based constraints that enhance filtering accuracy, ensuring a more context-aware candidate selection.

2.3.1 NeckAr tool for Wikidata clustering

For a first type filter methodology a state of the art approach for mapping entities into macro categories is required. These classes will be called in this thesis *NER types*.

The paper titled *"NECKAr: A Named Entity Classifier for Wikidata"* [17] introduces a new tool, which is designed to classify entities in Wikidata into three broad categories: PERSON, LOCATION, and ORGANIZATION. The motivation behind the development of NECKAr stems from the need for an up-to-date and straightforward entity classification system which are usually static and not designed for continuing evolving knowledge bases.

As a result of the classification process, the authors have compiled a dataset of over 8 million entities from Wikidata, each tagged with one of the three classes. This dataset,

called Wikidata NE, is made publicly available[1], providing a valuable resource for researchers and practitioners working in fields related to NER and information extraction.

NeckAr paper [17] make use of the SPARQL using Wikidata Query Service for identifying the subclasses of a root class.

The root classes and relative subclasses to identify then are:

- *geographic location* with the id Q2221906.
- *organization* with the id Q43229.
- *human* with the id Q5.

A more in-depth analysis of this methodology, as applied in the context of this thesis, will be presented in the following chapter dedicated to methodology.

The objective of the NeckAr paper is to develop a new dataset. In contrast, this thesis focuses on assigning an higher level type mapping to the explicit Wikidata types. The goal is enabling the retrieval system (lamAPI) to recognize the NER type of the candidates. Therefore, the specifics of how the new dataset is created and stored will not be covered in this discussion.

However, a few important considerations must be addressed. For some items, it is challenging to clearly determine the appropriate cluster. The most significant overlaps occur between the root classes *geographic location* and *organization*. For example, entities such as *hospital*, *state*, or *library* can be interpreted in multiple ways, making it difficult to assign them to a single, definitive category.

The upcoming section will delve deeper into this issue and offer a possible solution.

[1] event.ifi.uni-heidelberg.de

2.3.2 Addressing ambiguities in entity classification: a methodological update

Class overlapping poses a significant challenge which has to be discussed.

In 2019 a group of researcher from Sri Lanka publish a paper called *Named Entity Extraction of Wikidata Items [2].* They analyze how many items were overlapped over the three classes of *human*, *geographic location* and *organization*. Using a Wikidata dump downloaded on 06-Aug-2018 the results were the following:

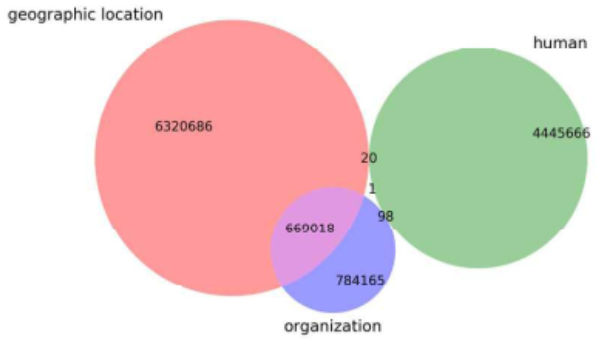

Figure 2.8: logic diagram of Wikidata's geographic location, human and organization classes.

The figure illustrates the overlap between different entity classes. This study adopts a method similar to that used in the NeckAr paper [17] to extract entities categorized as PERS, LOC, and ORG.

The primary improvement in this approach lies in the implementation of stringent filtering for ambiguous classes. Unlike the previous study, which allowed ambiguous items to be present in multiple classes, this method, as detailed by *Shanaz et al.* [2], excludes any subclasses from certain categories to avoid ambiguity.

In particular:

- *educational institution* (Q2385804), *government agency* (Q327333), *international organization* (Q484652), and *time zone* (Q12143) for **LOC**
- *country* (Q6256), *city* (Q515), *capitals* (Q5119), *administrative territorial entity of a single country* (Q15916867), *venue* (Q17350442), *sports league* (Q623109), and *family*

(Q8436) for **ORG**

For example by excluding the *educational institution* subclass tree from the *geographic location* root class, we ensure that all types of schools, kindergartens, colleges, and higher education institutions are exclusively categorized as ORG entities.

The results of this method are shown here:

Entity type Assignment	**Number of entities extracted based on**	
	Our Approach	**NECKAr**
LOC only	1,174,896	1,021,923
PER only	2,116,964	2,117,016
ORG only	431,673	377,294
Multiple entity type assignment	64,123	304,181
Incorrect entity type assignment	496,816	464,058
Total	**4,284,472**	**4,284,472**

Table 2.1: Comparison of the *Shanaz et Al.* [2] approach with NeckAr based on YAGO3.1

The table 2.1 highlights that this approach significantly reduces the number of multiple entity type assignments for Wikidata items. As a result, it extracts a greater number of ORG and LOC entities from Wikidata.

Assigning a category to an entity is a complex task, particularly when the context is unclear, and an entity can have multiple meanings depending on its usage. A rigid classification rule may not be helpful during the candidate retrieval phase due to these complexities.

To address this, the approach proposed in this thesis involves mapping each entity's direct types to a specific Named Entity Recognition type (PERS, LOC or ORG). However, to prevent overly restrictive classification, the solution considers all possible type mappings when assigning a NER type to an entity. As a result, while each entity type may be associated with a univocal NER type (as explained in [2]), the entity can also have multiple NER types assigned based on the broader set of type mappings.

Chapter 3

Modeling and improving type-based filtering

As explained in the chapter above the candidate generation begins with the retrieval task where a system identifies a subset of potential Knowledge Graph entities that could correspond to a given query (mention). This step is crucial because it directly impacts the subsequent disambiguation phase, where the most appropriate entity is selected from the candidate set.

Ensuring that the correct entity is included in the set while the irrelevant ones are minimized is a key determinant of the overall EL system's performance.

The filter strategy that this work wants to proposed is type-based. By restricting the candidate pool to the correct types, the disambiguation phase becomes more focused, reducing ambiguity and improving the likelihood of selecting the right entity.

Let's clarify the problem with an example: a user gives to *lamAPI* retrieval system the string "**Paris**" without any kind of type based filter but only with the string matching algorithm used in lamAPI. For simplicity the candidate set is limit to 20 items.

In the table 3.1 the retrieval is done without take into account the candidate types but only the string name matching. If the expected entity is the French capital Paris, with a limit of 20 entity in the set the correct candidate will not be included. It's possible also to notice that in the set are included entities which have completely different types (for example *photograph*, *song* or *television series*. Knowing how types are modeled can be used as solution for making the candidate set more type aware.

This use case scenario requires the correct knowledge availability of the entity types. Unfortunately Wikidata ontology includes not just problems related to class order (class level in the hierarchy) but general problems with incorrect subclass and instance links

Entity Name	**Entity ID**	**Entity Types**
Paris Paris	Q111210960	*television series*
Paris Paris	Q42291481	*film*
Paris Paris	Q113859728	*television series*
Paris Paris Dix-Sept	Q106582858	*municipal newsletter*
Papilio paris paris	Q22104161	*taxon*
PARIS	Q124817316	*scholarly article*
Paris	Q2249942	*Wikimedia set index article*
Paris	Q974043	*locality*
Paris	Q121046410	*song*
Paris	Q19469915	*dictionary entry*
Paris	Q12503205	*village in Indonesia*
Paris	Q41094569	*biographical article*
Paris	Q28726557	*metro station*
Paris	Q7137193	*unincorporated community in the United States*
PARIS	Q786333	*null*
Paris	Q64162970	*photograph*
Paris	Q28337101	*musical work/composition*
Paris	Q64156617	*photograph*
Paris	Q58733850	*scholarly article*
Paris	Q19222388	*version, edition or translation*

Table 3.1: Candidate set without specifying any type filter

in the ontology. One of the most recent analysis of the problem has been conducted by *Patel-Schneider et Al.* [18]. It found out a non-uniform modelling, incorrect instance and subclass links, vague descriptions of classes and items that are an instance of themselves.

Given the complexity of these challenges, the goal of this work is not to provide a definitive solution to all these issues but to propose a foundational approach for type-based filter strategies. Cases where type domain is unknown or incomplete due to lack of structure in Wikidata will not be solved in this work.

Now let's try in the example above to specify the type requirement *location* to be included in the set. In the table 3.2 the explicit Wikidata types are mapped to common NER categories for simplicity (as explained in the section *Neckar [2.3.1]*).

It's possible to notice that now the correct candidate is included in the set. Of course this contextual information is important for the retrieval. In the following section will show how this example can be extended to a general methodology definition.

Entity Name	Entity ID	Entity Types
Paris	Q974043	*locality*
Paris	Q12503205	*village in Indonesia*
Paris	Q7137193	*unincorporated community in the United States*
Paris	Q6065229	*Corregimientos of Panama*
Paris	Q3560147	*constituency of the French Fifth Republic*
Paris	Q2220917	*civil town of Wisconsin*
Paris	Q44873932	*movie theater, arts centre*
Paris	Q22134091	*unincorporated community in the United States*
Paris	Q2863958	*arrondissement of France*
Paris	Q3181341	*city in the United States, county seat*
Paris	Q960025	*city in the United States*
Paris	**Q90**	***commune of France, department of France, capital city, metropolis, tourist destination, ...***
Paris	Q7137161	*unincorporated community in the United States*
Paris	Q7137175	*unincorporated community in the United States, census-designated place in the United States*
Paris	Q25907009	*kampung of Papua*
Paris	Q576584	*city in the United States*
Paris	Q6922657	*mountain*
Paris	Q110940212	*human settlement*
Paris	Q79917	*city in the United States*
Paris	Q30621726	*unincorporated community in the United States*

Table 3.2: Candidate set with *location* as type filter

3.1 A conceptual framework for type filtering strategies

Consider the task of querying an Information Retrieval (IR) system that applies a type filter to select the Candidate set. On the user side, the **query type** can either be automatically determined by an algorithm or manually specified by the user. In both cases, the input is referred to as the **query type**.

The query type can be defined by a CTA algorithm as discussed in the previous section. Common tasks that implement this are *Column Classification, Type Annotation* and *Datatype Annotation* and they can be used to determine the type of a mention. For example the *MTab tool* (Figure 2.7) is designed to output the column type of the candidates (CTA).

However the focus of this thesis is the **interactive matching**, where query type is

either predefined or directly provided by the user. This approach is particularly valuable in cases where prior domain knowledge is available. By incorporating human-computer interaction, a user-driven strategy can enhance the type-based filtering mechanism of a retrieval system, leveraging human expertise to refine the candidate selection process.

On the system side, the entities returned by the IR system each have their own associated **candidate type**.

Both the **query type** and the **candidate type** can consist of one or more types.

The type domain of the Wikidata entities is defined as following in this work:

- **Explicit entity types** (or WD types): These are the types that are explicitly associated with an entity in Wikidata. They are directly linked to the entity within the Wikidata ontology and represent predefined categories such as *capital city*, *human* and *television series* in the tables 3.1 and 3.2.

- **Extended entity types**: These types are explicitly associated with entities in Wikidata but are extended through deterministic procedures. For example, a type may be extended through the application of a transitive closure, which helps identify indirect relationships or associations that are not directly stated in the Wikidata data itself. For example from an entity type *capital city* it can be extended with *administrative centre* and *geographic location*.

- **NER entity types** (or NER types): This form of explicit type extension maps the Wikidata entity types to a generic type in a flat classification scheme. From the most common categories of Named Entity Recognition task this work chooses four macro classes ORG, LOC, PERS and OTHERS (where entities of other categories belong to). As explained in the following part of this thesis the approached used to implement this mapping is based on the paper *NECKAr* [17].

Given this types domain the scenario presented before can be expanded defining the **query type** and the **candidate type** as sets of these possible types:

- ***WD types*** (*Explicit entity types*): one or more types explicitly defined in Wikidata, either as part of the mention type manually specified by the user during interactive reconciliation or associated with the candidate.

- ***NER type***: one or more Named Entity Recognition (NER) types associated with the mention type defined by the user or related to the candidate.

- ***extended WD types***: the transitive closure of the set of ***WD types*** of the mention or for the candidate.

3.1.1 Definition of type filtering technique

The **filtering operation** can be defined as function:

$$F(l, Q, \text{mode}) \rightarrow \mathcal{P}(C)$$

where:

- The domain consists of l, which is the label of the mention; Q, the *query type* set; and mode, which is a parameter indicating the matching strategy and can be either soft or hard ($\text{mode} \in \{\text{soft}, \text{hard}\}$).

- The codomain of the function is the power set of the candidate set, denoted as $\mathcal{P}(C)$, which represents all possible subsets of candidates.

Each candidate $c_i \in C$ is defined as the triple

$$c_i = \langle \text{id}_i, T_i, s_i \rangle$$

with:

- id_i the WD id of the candidate,

- T_i the set of its types (*candidate type*),

- s_i the candidate score defined by the retrieval system.

lamAPI retrieval system computes the s_i score for each candidate as follow:

$$s_i = \frac{\text{\_score}}{\text{max\_score}}$$

where _score is the relevance score assigned by Elasticsearch to a document, based on BM25 algorithm and computed with the following formula (for more information look the section 2.2.3):

$$\textbf{\_score} = \textbf{boost} \cdot \textbf{idf} \cdot \textbf{tf}$$

This score represents the relative relevance of a document to the search query, normalized against the highest relevance score in the result set. It helps compare how strongly each document matches the query, with values ranging from 0 (no relevance) to 1 (maximum relevance).

Give the set Q and T_i associated with a candidate c_i, the candidate list C is obtained as a result of matching Q and T_i based on an estimate of the **overlap** of Q and the types set T_i of each candidate.

The matching function determines for each candidate whether Q and T_i share any common types based on these criteria:

- **SOFT** matching function returns the list of candidates C, where each s_i, computed with BM25, has been updated to s'_i as follows:

  $$s'_i = s_i + \sum_{q \in Q} \delta\left(T_i, q\right)$$

  Here, $\delta(T_i, q)$ represents the contribution from the presence of type q in the candidate type set T_i. The score increment is greater when more types match, and the matching types are more relevant to the candidate.

  Let's take the example of **Paris (Q91)** to understand how the score is changing. The filter strategies are applied to the explicit types of the entity. The constraint

doesn't affect the match on the name, which must always be in a `must` clause.

```
{
  "query": {
    "bool": {
      "must": [{"match": {"name": {"query": "Paris", "boost": 2.0}}}],
      "should": [
        { "term": { "explicit_WDtype": "Q5119" } },
        { "term": { "explicit_WDtype": "Q484170" } }
      ]
    }
  }
}
```

Figure 3.1: Soft constraint query with Q5119: *capital city* and Q484170: *commune of France*

Documents that match either `"capital city"` ($q_1 \in Q$) or `"commune of France"` ($q_2 \in Q$) will be included in the results. The s'_i is additive, so documents that match *both* terms will receive an higher s'_i than those that match only one term:

- A document that matches only `"capital city"` will have s'_i equal to the BM25 score plus the contribution of `"capital city"`.
- A document that matches only `"commune of France"` will have s'_i equal to the BM25 score plus the contribution of `"commune of France"`.
- A document that matches both `"capital city"` and `"commune of France"` will have a higher s'_i as the two score will receive two contributions.

Both **Paris (Q90)** (French capital) and **Paris (Q576584)** (city in the US) will be included in the candidate set, but the first will have a higher score than the second. Additionally, the photograph **Paris (Q64156617)** will be included in the candidate set, but its score related to the types will be lower than the others because it's only composed by the original BM25 elastic score. Notice that in this case a simple label matching strategy and a soft strategy assign the same score to Paris (Q64156617).

- **HARD** matching function excludes candidates for which $\nexists t \in T_i$ such that $t \in Q$. Each $q \in Q$ does not contribute to the candidate score s_i; instead, they act as filters. The candidates are not penalized, but they are simply filtered.

  Analyzing the same example mentioned earlier:

```
{
  "query": {
    "bool": {
      "must": [
        {"match": {"name": {"query": "Paris", "boost": 2.0}}},
        { "terms": { "explicit_WDtype": [ "Q5119", "Q484170" ] } }
      ]
    }
  }
}
```

Figure 3.2: Hard constraint query with Q5119: *capital city* and Q484170: *commune of France*

A document must match at least one of the values (**OR clause**) in the *terms* query to be included in the results. In this case, s_i is not modified:

  - A document that matches `"capital city"` will have s_i based on its relevance computed with BM25 algorithm.
  - A document that matches `"commune of France"` will have s_i based on its relevance computed with BM25 algorithm.
  - A document that doesn't match any terms will be filtered out.

Both **Paris (Q90)** (French capital) and **Paris (Q576584)** (city in the US) will be included in the candidate set, but the first will have a higher score than the second. The key difference here is that entities with types not included in the clause will be excluded: the photograph **Paris (Q64156617)** will be excluded from the candidate set.

The candidate type set is defined with a function type(c) which will be combined with the **soft** and **hard** matching functions described above. In particular this function will have three different versions:

- **WDType**: This refers to the case where type(c) returns the types explicitly specified in Wikidata for a given candidate. This can also be denoted as WDType(c).
- **NERType**: This refers to the case where type(c) associates an abstract type, borrowed from the most common types used in Named Entity Recognition (NER), using a method specified later in the thesis.
- **extended_WDType**: This refers to the case where type(c) return the set of transitive closure associates to a set of WD types of a given candidate. The extension method is specified later in the thesis.

3.2 Entity types applied to filter strategies

From the domain types definition above is possible to highlight different filter strategies created as combinations of the different types domain and for each combination the matching function can be both hard or soft (table 3.3).

		Candidate types		
		WD type	**NERType**	**extended WD type**
Query types	**WD type**			
	NERType			
	extended WD type			

Table 3.3: Filter strategies feasibility.

The green cells represent combinations of entity types that are valid for retrieval, while the red cells indicate combinations that are either nonsensical or infeasible. In the end the orange cell is related to a baseline use case scenario still feasible but not implemented.

To better illustrate the applicability of the strategies outlined in the table, let's consider a single entity and its explicit type hierarchy in Wikidata as an example. This will provide

a clearer understanding of how these strategies can be applied in practice. **SquareSoft (Q207784)** has types {*video game developer, book publisher, video game publisher*}.

The hierarchy of the *video game publisher* is:

video game publisher → *video game company* → *software company* → *company* → .....

In the following part each cell of the table 3.3 will be described using the entity **SquareSoft (Q207784)** as example. The notation used for referring to a specific strategy (cell) will be [*query type - candidate type*]:

[**WD type - WD type**] This baseline scenario occurs when the query type assigned to a mention is an explicit Wikidata type, and candidate entities are filtered based on their explicit Wikidata types. It does not involve any type hierarchy extensions or inferences but relies strictly on the explicitly defined Wikidata types, ensuring a full overlap between the two sets. This baseline approach is not included in the experiments but is discussed solely in this section:

- *video game developer, book publisher, video game publisher* **WD types** as query types
- *video game developer, book publisher, video game publisher* **WD types** as candidate types

[**WD type - NER type**] The query type has to be a subset of the candidate type for having a non empty matched list. This strategy is not feasible:

- *video game developer, book publisher, video game publisher* **WD types** as candidate types
- *ORG* **NER types** as query types

[**WD type - extended WD type**] Introducing the **transitive closure** the explicit Wikidata types of the candidates will be extended until the root creating a non

empty set with the query type. WD type set is indeed a subset of extended WD type:

- *company* **WD type** as query type
- **extended WD types** as candidate types:
  - *video game developer* → ... → ...
  - *book publisher* → ... → ...
  - *video game publisher* → *video game company* → *software company* → *company* → ...

[**NER type - WD type**] Similar to the reverse strategy, there is no direct overlap between these two sets, making this approach unfeasible.

[**NER type - NER type**] The two sets are joint so the match can produce a non empty set:

- *ORG* **NER type** as query type
- **NER types** as candidate types:
  - *video game developer* = *ORG*
  - *book publisher* = *ORG*
  - *video game publisher* = *ORG*

[**NER type - extended WD type**] NER type is a mapping of a subset of the extended WD types of the candidates. In the example above *video game publisher* type can be extended until its superclass *oraganization* which is mapped to the *ORG* NER type. That's explain why the overlapping of the two sets is not empty:

- *ORG* **NER type** as query type
- **extended WD types** as candidate types:
  - *video game developer* → ... → ...

- *book publisher* → ... → ...
- *video game publisher* → *video game company* → *software company* → *company* → ... → *organization* = *ORG* → ...

If in the transitive closure none of the entity types are related to LOC, ORG or PERS the entity will be mapped to OTHERS NER type.

[**extended WD type - WD type**] The two sets are joint only if the extended WD type of the query type is included in WD type on the candidate type. The following example illustrates why this restriction does not make this strategy optimal for the experiments:

- **extended WD type** as query type:

  company → ... → *organization* → ...

- **WD types** as candidate types:
  - *video game developer*
  - *book publisher*
  - *video game publisher*

[**extended WD type - NER type**] This strategy is very similar to the reverse approach. For simplicity, it is presented here but will not be included in the experiments.

[**extended WD type - extended WD type**] This approach extends both the query and candidate types using the Wikidata hierarchy, which ensures some degree of overlap. However, it is inefficient because expanding both sides significantly increases the number of potential matches, leading to higher computational costs and a loss of precision.

As summary example the following table 3.4 contains the type domain for the entity **SquareSoft (Q207784)**.

If we consider the following types as query type we have that:

- specifying one of the explicit **WD types** of the entity will create a non empty set having WD type and explicit type as filter strategy on the candidate side. By the way as said before the first approach is a baseline, so will not be included in the experiments.

- considering a query type with **NER type** can return a correct matching if either NER type or extended WD type is considered for the candidates.

- if the query type is an **extended WD type** the matching will still be valid when using either an NER type or an extended WD type as the candidate type. However, these combinations are simply the reverse of previously discussed approaches and are mentioned here only for completeness.

Explicit WD types	Extended WD types	NER type
video game developer video game publisher book publisher	video game developer video game company software company company business technology company legal person group of humans commercial organization organization juridical person corporation economic entity enterprise complex system social system person or organization corporate body group of living things system group of physical objects agent object collective agent collective entity entity independent continuant group or class of physical objects continuant book publisher publishing company publisher video game publisher board game publishing company software publisher	ORG

Table 3.4: Candidate types domain for the entity **SquareSoft**.

3.3 Implementation of types extension

The final section of this chapter focuses on illustrating, through examples, how NER (Named Entity Recognition) and extended types are derived from Wikidata. These methodologies enable the retrieval system to utilize generic or higher-level hierarchical types for more effective candidate filtering.

3.3.1 Explicit Wikidata types mapped to Named Entity Recognition categories

Named Entity Recognition (NER), as for many NLP tasks, required the entities mention classification into predefined categories. There are many of them: *Person, Location, Organization, DataTime, Event, Product, Email, URL, ...* For this work the aim is to use some of them for defining a general type schema among the Wikidata type hierarchy. The methodology is based on the approach proposed by *Neckar* [17].

Having a list of explicit WD types mapped into a general class is possible to associate a **NER type** to an entity.

Explicit types are mapped if they are present in one of the following lists:

- `organization_subclass`
- `geolocation_subclass`
- `person`

For each macro class the list with their subclass types are included. There are many overlaps among the categories LOC and ORG as seen in the section 2.3.2.

To address the problem as described in the paper [2] discussed in the section 2.3.2 the macro classes are restricted to only domain related subclasses. For example the class *country* and its subclasses (computed with the same methodology of the general classes 3.3.1) are removed from `organization_subclass`.

Let's first take as example the type *country* (Wikidata Id Q6256) and see how the complete list of types which are ***"subclass of"*** *country* is defined:

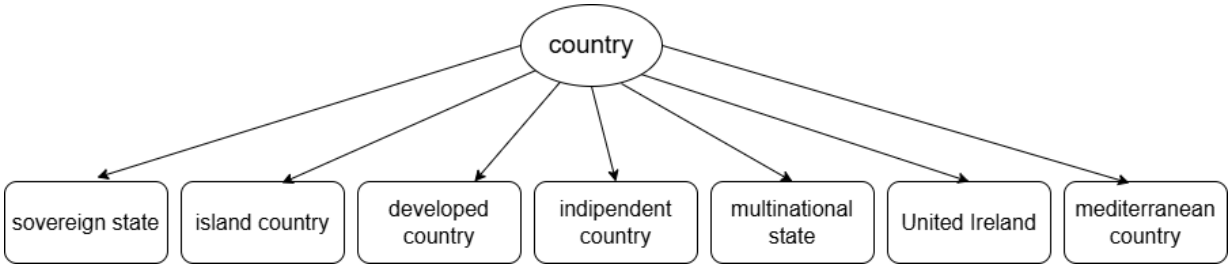

Figure 3.3: A subsample of the first level of country subclasses.

This list of all its subclasses is retrieved using the following recursive SPARQL query:

```
SELECT ?WD_id WHERE {
    ?WD_id (wdt:P279)* wd:Q6256 }
```

The query will recursively traverse backward (`(wdt:P279)*`) through the *subclass of* hierarchy, ensuring that all relevant subclasses are captured.

The same query is central to the process used to generate the lists referenced in 3.3.1. After that the NER type mapping is determined looking in which of the created subclasses lists the entity belongs to. If no match is found the entity will be mapped to NER type category. This mapping is replicated for each Wikidata type associated with an entity. The result is a list of NER types corresponding to the entity.

The overall process is so summarized:

1. **Generate Lists of Subclasses**: Lists of all subclasses for the categories `organization_subclass` (ORG), `person`(PERS), and `geolocation_subclass` (LOC) are created.

2. **Remove Overlapping Categories**: Overlapped categories between ORG and LOC are removed.

3. **Map Entity Types to NER Categories**: Each explicit type of an entity is checked against the generated lists to determine its corresponding NER type. If none of them belong to any list the type is mapped to OTHERS type. The final NER type for the entity is computed based on this mapping.

To better understand the overall workflow consider as example the entity **Belgium**, which has a diverse and comprehensive set of WD types:

Explicit WD types name	Explicit WD types id	NER type
sovereign state	Q3624078	LOC
federation	Q43702	ORG
country	Q6256	LOC
realm	Q1250464	LOC
member state of the European Union	Q185441	LOC
colonial power	Q20181813	ORG

Table 3.5: NER type mapping for **Belgium**.

So **Belgium** (Q31) entity will have [ORG, LOC] as NER type.

3.3.2 Type extension from explicit Wikidata types

A similar approach is used to retrieve the **Extended WD types**, instead of starting from a given type A and retrieve all the types B which are subclass of A, here the opposite query is runned for retrieving the whole superclass list:

```
SELECT ?superclass ?superclassLabel WHERE {
    wd:Q6256 (wdt:P279)* ?superclass.
SERVICE wikibase:label { bd:serviceParam wikibase:language
                "[AUTO_LANGUAGE],en".  }
```

The iterative backward process returns a pair key-value object and taken as example **Belgium** (Q31) with the types listed in the table 3.5:

- ***Q3624078: sovereign state***: *Q6256: country, Q7275: state, Q1048835: political territorial entity, Q4835091: territory, Q155076: juridical person, Q177634: community, Q1896989: subject of international law, Q96196009: former or current state, Q3778211: legal person, Q874405: human social group, ..., Q43229: organization, Q2221906: geographic location, ...*

- ***Q43702: federation*** *Q1140229: political union, Q22676603: federal system, Q484652: international organization, Q1048835: political territorial entity, Q28108: polit-*

ical system, Q1639378: social system, Q56061: administrative territorial entity, Q854457: complex system, Q16334295: group of humans, ..., Q43229: organization, Q2221906: geographic location, ...

- ***Q6256: country*** *Q1048835: political territorial entity, Q4835091: territory, Q56061: administrative territorial entity, Q82794: geographic region, Q15642541: human-geographic territorial entity, Q16562419: political entity, Q618123: geographical feature, Q26713767: region of space, ... Q27096213: geographic location, ...*

- ***Q20181813: colonial power*** *Q3624078: sovereign state, Q6256: country, Q7275: state, Q1048835: political territorial entity, Q4835091: territory, Q155076: juridical person, Q177634: community, Q1896989: subject of international law, Q96196009: former or current state, Q3778211: legal person, Q874405: human social group, ... , Q2221906: geographic location, ...*

- ...

The union set of these type extension is the final Extended WD types of the Wikidata entity **Belgium**. Is possible to notice that NER type is a schema classification included in the extended WD types because it's an extension as well of the explicit WD types. For example the type *federation* is either a subclass of *organization* and *geographic location*.

As discussed in the research by *Patel-Schneider et al.* [18], the Wikidata types hierarchy contains both missing types and loops. This work does not address the issue of **missing types**: candidates without explicit Wikidata types will also lack extended or NER types due to the absence of relevant information in the knowledge base.

For handling possible **loops** in the types hierarchy, the simplest solution has been implemented: a backoff strategy with a limited number of iterations. In this approach, the system temporarily backs off for a short period before retrying the action, with exponentially increasing delays. The number of retries is limited, ensuring that the process does not continue indefinitely when the same outcome is repeatedly retrieved.

3.4 Detailed review of the type extension process in lamAPI

As explained in the next chapter, query types are derived from tabular data, while candidate types are associated with entities from Wikidata, which are retrieved through the lamAPI system. The lamAPI tool, as discussed in the section dedicated to type-aware retrieval systems (subsection 2.2.3), utilizes custom indexing techniques for candidate generation and stores the data in MongoDB.

Implementing an extension to the type hierarchy requires adjustments in how entities are stored in the database and indexed. The first step of this thesis, prior to the experimental phase, involves enabling the lamAPI retrieval system to generate candidates based on the following types:

- Explicit Wikidata (WD) types of the entities.
- Extended WD types, as detailed in section 3.3.2.
- Named Entity Recognition (NER) types, as mapped in section 3.3.1.

The workflow for the lamAPI retrieval system, as illustrated in Figure 2.6 in section 2.2.3, is as follows:

1. **Storaging** of the Wikidata dump file into MongoDB, each iteration processes one entity from the dump, which is stored as a document in MongoDB. This document includes the following information:
    - *WD types* of the entity extracted directly from the dump
    - *NER types* mapping of each WD type using the methodology outlined in section 3.3.1
    - *Extended WD types* are derived from the transitive closure of each WD type associated with an entity. As discussed in section 3.3.2, this extension requires querying Wikidata, making it the most computationally expensive part of this phase. To optimize the storaging phase, the transitive closure of each type has

been **persistently stored** in MongoDB, preventing redundant queries on the same explicit WD type.

2. **Indexing** the MongoDB collection in an ElasticSearch index to facilitate candidate retrieval during the experimental phase of this thesis.

Chapter 4

Evaluation of type-based filtering strategies in Entity Linking

The previous chapter introduced a structured methodology for enhancing the explicit type hierarchy in Wikidata, aiming to effectively extend this hierarchy through a baseline implementation of the transitive closure.

This chapter evaluates the aforementioned filter-based strategies across different levels of type granularity, ranging from explicit Wikidata types to their extended forms. These extensions include the complete transitive closure and mappings to NER macro types. The evaluation aims to assess the quality of the candidate set in terms of coverage and ranking performance (MRR), providing insights into the effectiveness of these strategies.

4.1 Experimental setup for type-based filtering

This section aims to explain how the experiments will be conducted and the reason behind the selection of specific data types and evaluation metrics used to assess the effectiveness of the filter strategies discussed earlier.

The experimental results are closely related to the nature of the tabular data used for entity reconciliation. The experiments were carried out using two distinct types of data:

- **general domain** tabular datasets which include mentions and query types that span multiple categories with varying levels of granularity. This setup provides a broad evaluation of the filtering approach across diverse entity types
- **domain specific** tabular dataset, where the mentions focus exclusively on entities belonging to the organization class (specifically, the *ORG* NER type in the experiments). It is designed to simulate scenarios where the query type is manually

specified, ensuring a targeted evaluation of filtering strategies

4.1.1 General domain datasets overview

Each dataset utilized in the experiments consists of multiple tables and their corresponding CEA, CTA, and CPA annotations, as derived from the SemTab challenge[1].

The following datasets are different in terms of size and the span of classes and properties from diverse domains:

- **Round1-T2D** [19]: extracted from the T2Dv2 Gold Standard 2019[2], these dataset comprises manually annotated correspondences such as row-to-instance, attribute-to-property, and table-to-class, distributed across 779 web tables. It's particularly suitable to benchmark retrieval system due to its large amount of tables [20]

- **Round3** [21]: this dataset, from SemTab challenge 2020, stands out for featuring a substantial number of abbreviated individual names, like 'J. F. Kennedy' introducing a nuanced layer of complexity to the dataset

- **HardTableR2** [22]: derived from SemTab 2022, it's 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

- **HardTableR3** [22]: similar to HardTableR2 but distinguished by the fact that each table contains only one column referred to a specific entity, coupled with an average of 8 rows per table. Some of the mention labels are misspelled or containing typos

- **Round4** [23]: constructed during the SemTab 2020, it encompasses 22 207 tables, averaging 21 rows per table

- **2T-2020**: presented in the paper [24], it has been recognized for its high-quality tables which incorporate cells with ambiguous identifiers, typos, and misspelled entity

[1] cs.ox.ac.uk/isg/challenges/sem-tab/
[2] goldstandardV2.html

names, it houses approximately 70 000 unique cells in 180 tables. It supports the testing of candidate retrieval with misspelled words

In the following table the overall statistics related to these datasets are shown:

Dataset	Tables	Columns	Rows	# Entities (CEA)	# Classes (CTA)	# Predicates (CPA)
Round1-T2D	64	323	9,089	8,078	119	115
Round3	2,161	9,736	152,753	390,456	5,761	7,574
Round4	22,207	78,750	475,897	994,920	31,921	56,475
2T-2020	180	802	194,438	667,243	539	0
HardTableR2	1,750	5,589	29,280	47,439	2,190	3,835
HardTableR3	7,207	17,902	58,949	58,948	7,206	10,694

Table 4.1: Statistics of the Datasets Used in the Experiments.

4.1.2 Domain specific dataset overview

Assessing the ranking quality and coverage of the evaluated strategies in a single-type domain is crucial. A retrieval system can be effectively utilized when the type of the correct candidate is known *a priori*.

To conduct a more in-depth analysis, we introduce a tabular dataset referred to as **ORG_dataset** which contains mentions related only to companies and universities. This dataset has been originally designed and created to evaluate the capability of *Alligator* [12] in reconciling GitHub pages with the corresponding organizations their users belong to.

The dataset was constructed through a multi-step process:

1. **Scraping Phase:** The initial dataset was created by collecting GitHub pages, with each record corresponding to a single page.

2. **Dataset Structure:** The dataset consists of the following columns related to the user profile page on GitHub: *company, user name, bio, website*, and *description*.

3. **Data Cleaning and Annotation:** All records with missing values in the *company* column were removed. The remaining companies were manually annotated with their corresponding Wikidata IDs.

This final step results in a CEA dataset for this experiment, comprising **916 records** related to companies (*ORG* NER type), each annotated with its respective ground truth entity in Wikidata.

4.1.3 Evaluation process of the candidate set

The aforementioned tabular data are used for constructing the query types for the experiments: the CEA tables are important for the Wikidata ID (the ground truth) of each label mention. For each mention the CTA tables are used for deriving the Wikidata ID related to their column type. For each dataset has been created a data structure containing the necessary information as explained in the section 3.1.1: the label, its related type (explicit type from CTA, its mapping to NER type and the list of its transitive closure) and the filter mode (hard or soft).

As explain in the previous chapters lamAPI leverages a label matching retrieval based strategy. In the lookup service a certain tolerance error in the input string has been allowed thanks to the *fuzzy* search. In case a strict label matching constraint doesn't produce any result, a fuzzy search on the label is applied. Elasticsearch allows a particular type of fuzzy search based on the **Levenshtein Distance**. This metric measures the number of single-character edits (insertions, deletions, or substitutions) required to change one string into another. The smaller the edit distance, the closer the match. However it may require slightly more processing time compared to exact token matching.

The validation process starts with a set of mentions M, and a number k of candidates associated with each mention. The maximum number k is the *limit* parameter in the query and it has been set to 100 by default, which has been empirically demonstrated that provide a good level of coverage.

The lookup service returns a set of candidates E_m that includes all the candidates found. The returned set is then checked against the CEA table IDs to verify which among the correct entities are present and in what position in the ranked results they are. Given a set of candidates to find related to the M mentions, the **coverage** is computed following

this formula:

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

It shows how well the system retrieves relevant items from the entire set of possible relevant items. The experiments are conducted also in order to evaluate the ranking capacity of the system in terms of relevant candidates among a list of N retrieved candidates. This measure is called **Mean Reciprocal Rank (MRR)** and it helps to understand the average position of the first relevant candidate across set:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \tag{4.2}$$

Where, $rank_i$ indicates the position of the first relevant candidate c_i for a given query in the top-k results.

4.2 Impact of filter strategies on the candidate set

In this section the goal is to evaluate the proportion of relevant items that appear in the retrieved results (**coverage**) and the system's ability to rank relevant candidates higher (**MRR**).

4.2.1 General domain dataset performances

In the following table are shown the experimental results using the aforementioned general-domain datasets:

Query Type	Candidate Type	Mode	Round1	Round3	Round4	2T	HardTable2	HardTable3
no filter			0.827	0.890	0.823	0.540	0.884	0.741
NERtype	**NERtype**	soft	0.827	0.892	0.801	0.544	0.880	0.750
NERtype	**NERtype**	hard	0.789	0.770	0.698	0.534	0.744	0.542
explicit_WDtypes	**extended_WDtypes**	soft	0.827	**0.899**	**0.869**	**0.550**	**0.914**	**0.826**
explicit_WDtypes	**extended_WDtypes**	hard	0.636	0.752	0.789	0.525	0.855	0.781
NERtype	**extended_WDtypes**	soft	**0.846**	0.859	0.804	0.544	0.868	0.755
NERtype	**extended_WDtypes**	hard	0.731	0.618	0.720	0.539	0.768	0.617

Table 4.2: Coverage results at N = 100

It's clearly visible that hard filters, unlike soft filters, do not improve the original retrieval strategy without type-based filtering.

For all the general domain datasets used, it is clear that using a soft strategy with explicit WD types to match the extended WD types of the candidates improves coverage the most.

Now, let's evaluate how *lamAPI* ranks relevant candidates using a type-based filtering strategy.

Query Type	Candidate Type	Mode	Round1	Round3	Round4	2T	HardTable2	HardTable3
no filter			0.797	0.807	0.861	0.435	0.858	0.800
NERtype	**NERtype**	soft	0.815	0.816	0.878	0.468	0.867	0.832
NERtype	**NERtype**	hard	0.776	0.708	0.712	0.451	0.716	0.556
explicit_WDtypes	**extended_WDtypes**	soft	**0.856**	**0.826**	**0.913**	**0.481**	**0.900**	**0.909**
explicit_WDtypes	**extended_WDtypes**	hard	0.842	0.815	0.707	0.461	0.726	0.600
NERtype	**extended_WDtypes**	soft	0.821	0.824	0.868	0.474	0.881	0.834
NERtype	**extended_WDtypes**	hard	0.712	0.565	0.762	0.469	0.744	0.663

Table 4.3: MRR results at N = 100

In terms of MRR almost all the filtering strategies with soft constraint improve the original retrieval without any filters. In particular the best soft strategy in terms of MRR is [Explicit WD type - Extended WD type]. For *Round1* and for *HardTable* datasets the three tested strategies in soft mode shown the biggest improvement.

Considering the previous retrieval strategy in *lamAPI* we can identify use cases where a type-aware approach significantly outperforms a simple label-matching method. Looking at the tables 4.2 and 4.3 is possible to say that:

- **soft filters** improve the most basic label matching strategy in terms of *coverage* and *MRR*. In particular considering a filtering operation with overlapping between

  1. NER type and NER type
  2. explicit WD type and extended WD type

- For **Round1** and **Round3**, using explicit Wikidata types as query types and filtering candidates based on extended types improves *MRR* also with an hard filtering

- In **HardTableR3** using the overlapping between explicit WD type and extended WD type it's visible the best performances of soft strategies compared with the no filter one.
- **HardTableR3** generally shows lowest values of coverage and MRR for hard strategies compare to the other datsets
- **2T** has been revealed as the most difficult dataset showing the worst performances. More consideration about that will follow

Considering explicit Wikidata types during retrieval has shown the best performance in terms of ranking and coverage when they are matched against extended Wikidata types. It is interesting to evaluate this soft strategy performs against the no-filter baseline changing the size of the candidate set.

The following plots represent the trend of coverage and MRR for **Round4** as the number of candidates in the set increases. Considering [*Explicit WD type - Extended WD type*] soft strategy:

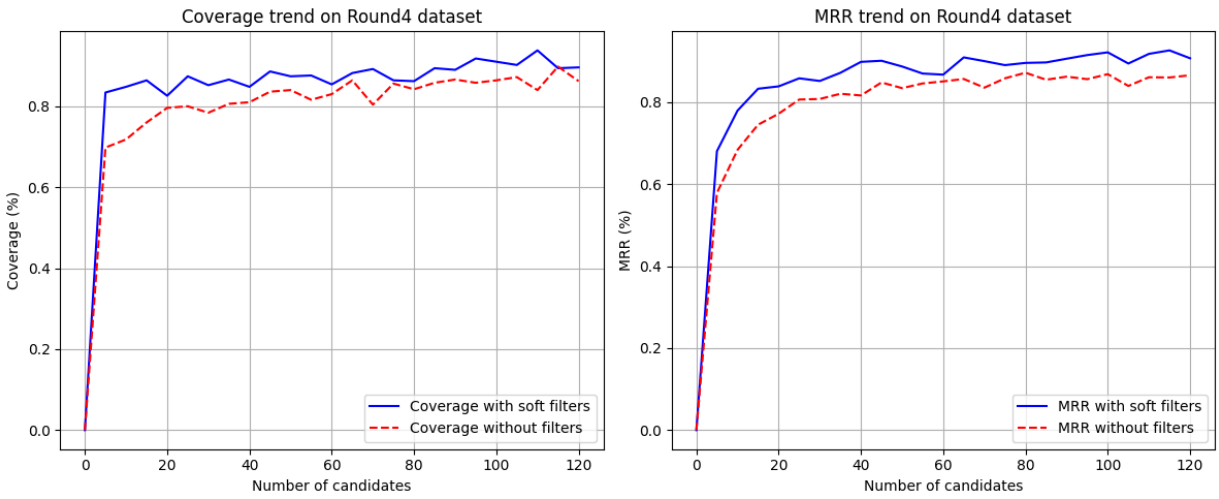

Figure 4.1: Coverage and MRR trends for Round4 with [*Explicit WD type - Extended WD type*] **soft** strategy.

Figure 4.1 illustrates how both metrics consistently outperform the no-filter strategy, regardless of the candidate set size. In terms of ranking quality, MRR achieves its highest values 0.910 when the candidate set reaches approximately 40 candidates.

Coverage peaks with a smaller candidate set compared to MRR, which stabilizes at approximately 0.820 when the candidate count reaches 20.

HardTableR3 exhibits the lowest coverage and MRR in **hard mode**. Taking as example the performance of [*NER type* - *NER type*] strategy in hard mode, it is worth analyzing whether these results are influenced by the chosen candidate set size (100) in the experiments.

To gain further insight, the following figure illustrates the trends of these measures as the candidate set size increases:

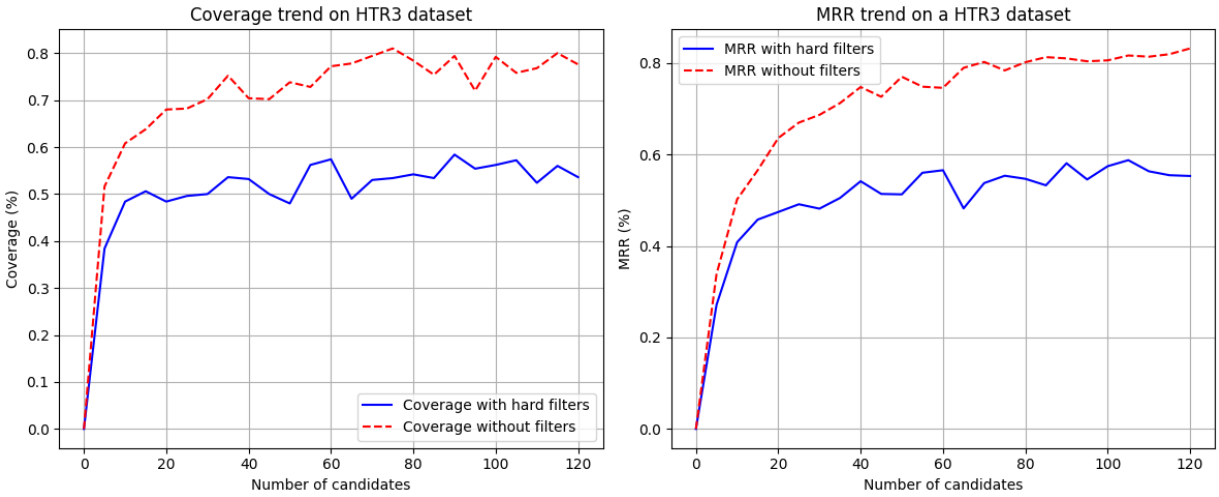

Figure 4.2: Coverage and MRR trends for HardTableR3 with [*NER type* - *NER type*] **hard** strategy.

It's clearly visible how the low values for coverage and MRR are not related to the size of the candidate set. In this experiment HTR3 has been revelead as one of the worst hard filter performances for hard strategy.

About **2T** the main reason of the bad results in terms of coverage and MRR is related to the nature of the dataset. It containes mispelled mentions like *Steve Blackkk* for the entity **Steven Black (Q7614497)** or *Johnh Callahannn* for **John Callahan (Q1699525)**. Even using a fuzzy search the correct candidate is not included in the set of 100 items retrived from lamAPI using both soft or hard constraint strategy.

4.2.2 Domain specific dataset performances

A common use case for a retrieval system is when the category of mentions is known beforehand. The ORG_dataset consists exclusively of mentions mapped to the ORG category and experiments on this domain-specific dataset has been conducted:

Query Type	Candidate Type	Mode	Coverage	MRR
no filter			0.712	0.680
NERtype	**NERtype**	soft	0.722	0.707
NERtype	**NERtype**	hard	0.714	0.666
NERtype	**extended_WDtype**	soft	**0.727**	**0.709**
NERtype	**extended_WDtype**	hard	0.709	0.664

Table 4.4: Domain-specific dataset: ORG_dataset results at N = 100

The ranking capacity (MRR) generally decreases with hard filters compared to the original label-matching-only strategy. However, coverage generally shows a slight improvement, particularly with the strategy [*NERtype* - *extended_WDtype*].

The performances on this dataset are generally worse than the general domain datasets. The introduction of entity types for candidate discrimination provides only a minimal contribution in this use case. This is primarily due to the nature of the mentions contained in *ORG_dataset*, which consists of multi-token mentions, often including abbreviations, which are difficult to match for lamAPI:

Label	Wikidata ID
The Graduate Center, CUNY	Q1024543
Zhejiang University & Westlake University	Q986087
BAAI	Q107518033
Capgemini Spa	Q1034621
Alation	Q107639776

Table 4.5: Example of CEA for ORG_dataset.

Aligned to the general domain analysis done before also here the trend for [*NERtype* - *NERtype*] soft strategy with a candidate set size variable has been evaluated.

Figure 4.3 depicts how the coverage and MRR values change with the number of candidates. In particular it remains constant around 0.790 when the number of candidates ranges from 40 or 60 until reaching the limit. The MRR reaches its peak at a higher

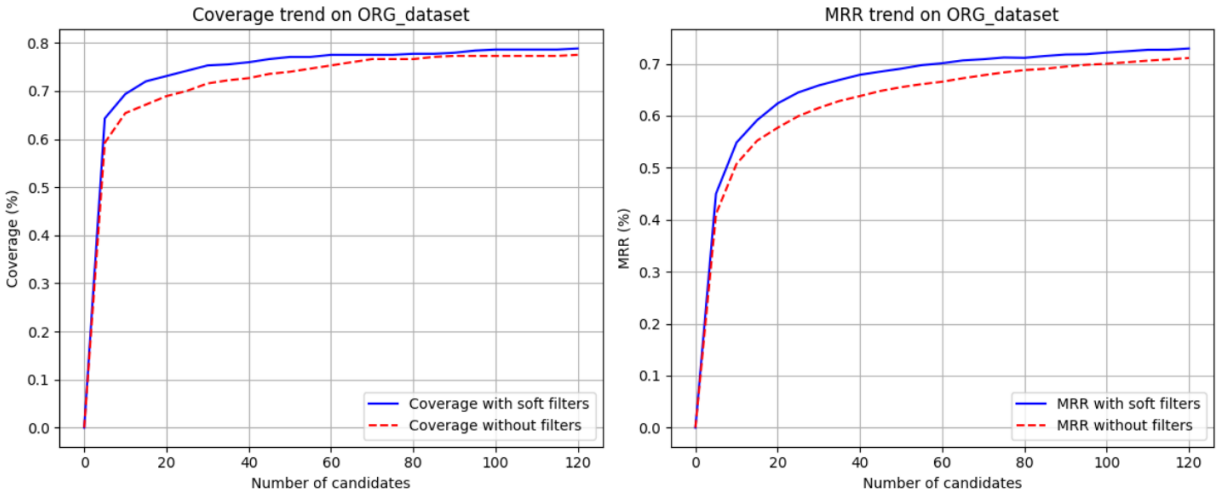

Figure 4.3: Coverage and MRR trend for ORG_dataset with [*NER type* - *NER type*] **hard** strategy.

number of candidates. However, for both coverage and MRR, increasing the candidate set beyond 100 does not lead to any further improvements, making 100 a reasonable upper limit.

The results align with previous experiments, showing that a soft filtering strategy can outperform standard label-matching retrieval without type awareness.

4.3 More insight on the filter strategies behavior

In the previous section, the results demonstrated that *soft filters* enhance both the ranking quality and coverage of LamAPI. However, the experimental outcomes revealed some interesting use cases:

- considering most of the tested datasets, coverage and MRR for **HardTableR3** are the worst comparing the hard strategies with the soft ones. In particular for the strategy [*NERtype* - *NERtype*] the performances are coverage 0.542 and MRR 0.556. As observed from the coverage and MRR trends, the results of the hard strategies are not influenced by the size of the candidate set.

- soft filter strategies outperform the original label matching strategy implemented in *LamAPI* and the hard constraint strategies. It's interesting to examine in detail how

the strategy modifies the candidate set for the same query

To gain deeper insights into these scenarios, it is crucial to analyze how the ground truth mentions are distributed across entity types, particularly NER types.

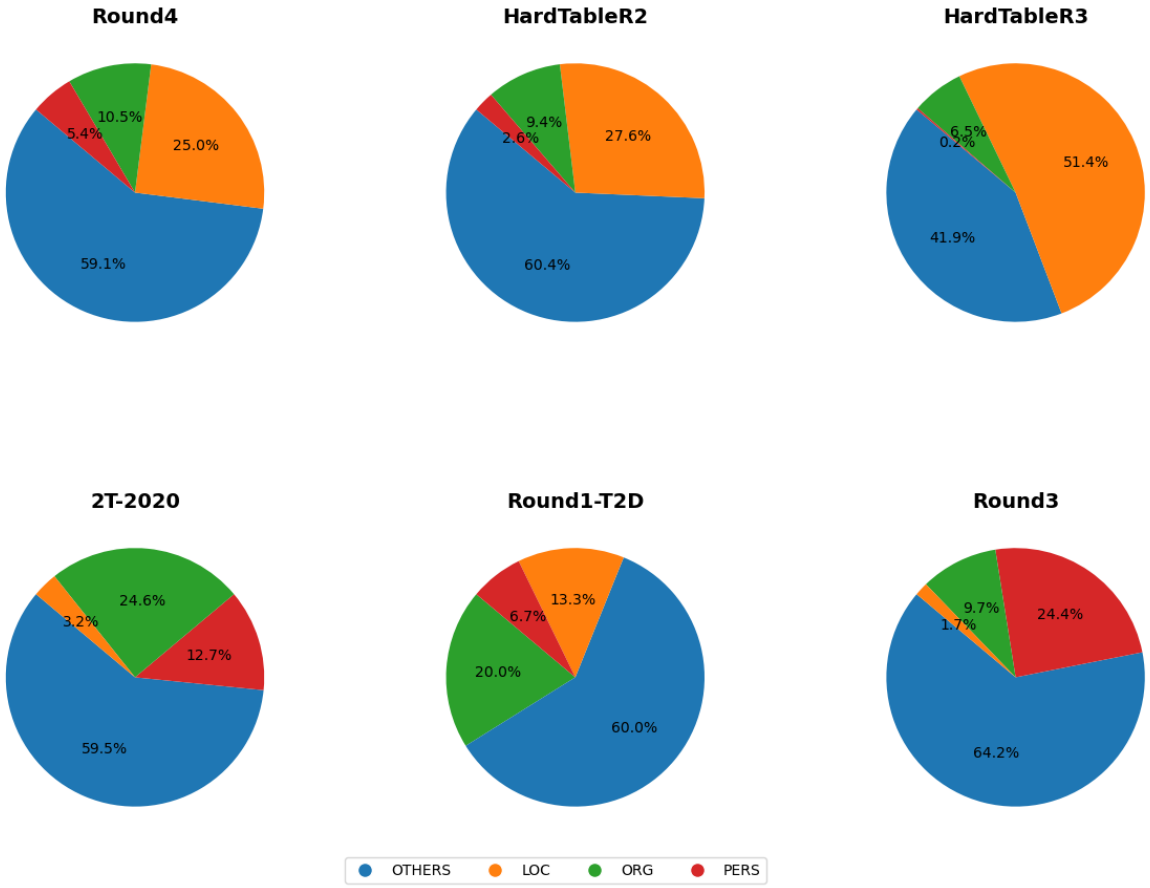

Figure 4.4: Composition of the general-domain datasets considering NER types.

Analyzing the first use case scenario presented above, the reason why **HardTableR3** performs poorly when using hard constraints is due to its nature. It contains many misspelled geographical location entities.

Some of these misspelled mentions are correctly retrieved from lamAPI only with soft constraints, which enhance their score with the contribution of the type match. Considering the [*NERtype* - *NERtype*] filtering strategy, in the table 4.6 some examples that are not retrieved with hard constraints but only with soft are shown.

For example the mention **Indepepdencia** is referred to the entity Q1190620: a department of Argentina called Indipendencia.

Name	ID
Gamma Cassiopeiae	Q13584
Brzachel	Q2033401
Baja California	Q58731
ITC One	Q4977466
Tamly Ho	Q30597885
Indepepdencia	**Q1190620**
USS Penacola	Q475540
Birnm	Q17049450
Carsoo Township	Q5046971
Aylor	Q5356021

Name	ID
Erg em Atchanf	Q23823192
Bauhauu	Q58196690
Kawabf district	Q1042053
L'Abadl	Q21480015
HTC Ong	Q4977466
Ria{or	Q28121742
Sierra de Castuil	Q108063737
Dieougou	Q1767670
Windsor Public Library	Q8024681
Malavalli	Q2451117

Table 4.6: Retrieved mentions from **HardTableR3** with soft but not with hard constraints.

So, considering only candidates filtered by NER type *ORG*:

- for hard filter the fuzzy search resolve the mispelling but the candidate is not included in the set of 100 entities due to its low score
- for soft filter the candidate Q1190620 update it's score thanks to the type matching and the entity is included in the candidate in position 53.

For understanding better how soft and hard filters modify the candidate set composition is important to understand how the relevance score is changing based on the constraint.

Considering the example of the mention *Welington High School*, taken from HardTableR3.

The correct entity is *Wellington High School*, an English private school identified in Wikidata as Q7981412. Using the fuzzy search a certain error tolerance is given for the misspelling.

Without considering any type based filtering strategy but the original lamAPI label matching with *fuzzy* search the correct candidate is included in the first 10 elements of the candidate set, precisely in the seventh position. The candidate set is shown in the table 4.7.

Name	Description	Score	ID
Wellington School, Bebington	former private grammar school in Wirral, England	1.0	Q7981484
Wellington High School	high school in Wellington, Collingsworth County, Texas	0.962	Q17035369
Wellington High School	high school in Palm Beach County, Florida, United States	0.959	Q7981413
Wellington High School	Wellington, NSW - Government - Secondary - 292	0.957	Q101435838
Wellington High School	high school in Ohio, United States	0.957	Q7981414
Wellington High School	Wikimedia disambiguation page	0.955	Q7981410
Wellington High School	**state secondary (year 9-15) school in Wellington, New Zealand**	**0.954**	**Q7981412**
Bebington High School	academy in Bebington, Merseyside, England	0.907	Q4878436
Ellington High School	public high school in Connecticut, U.S.	0.906	Q5365418
Arlington High School	public high school in Riverside, California, United States	0.906	Q4792347

Table 4.7: Candidate set with only a label matching strategy

However considering a soft filtering strategy, in this example [*NERtype* - *NERtype*], with *ORG* as query type, the candidate set has been re-ranked and the correct entities has been scored in the sixth position. The score of *Wellington High School* shown in the table 4.8 shows also how the soft constraint update its candidate score. Thanks to the type matching contribution of ORG both on query type and candidate type the score changes from 0.954 to 0.959.

Using a hard constraint doesn't modify the elasticsearch score of the candidate but the entities which NER type is not ORG are filtered out from the set (like the Wikimedia disambiguation page Q7981410). In the example having specified the correct query type the correct candidate is include and receives the same score as in the label matching strategy:

From these insights, it is clear that the behavior of the strategy affects the composition of the set, filtering out irrelevant entities when hard constraints are applied. In the case of soft constraints, the score update modifies the candidate ranking and can introduce new items into the set.

Name	Description	Score	ID
Wellington School, Bebington	former private grammar school in Wirral, England	1.0	Q7981484
Wellington High School	high school in Wellington, Collingsworth County, Texas	0.970	Q17035369
Wellington High School	high school in Palm Beach County, Florida, United States	0.963	Q7981413
Wellington High School	Wellington, NSW - Government - Secondary - 292	0.961	Q101435838
Wellington High School	high school in Ohio, United States	0.960	Q7981414
Wellington High School	**state secondary (year 9-15) school in Wellington, New Zealand**	**0.959**	**Q7981412**
Wellington High School	Wikimedia disambiguation page	0.911	Q7981410
Bebington High School	academy in Bebington, Merseyside, England	0.911	Q4878436
Ellington High School	public high school in Connecticut, U.S.	0.911	Q5365418
Arlington High School	public high school in Riverside, California, United States	0.91	Q4792347

Table 4.8: Candidate set with a **soft** strategy [*NERtype* - *NERtype*] filtering with ORG.

Label misspellings can pose a problem for hard constraints due to their filtering nature: *Indepepdencia* is an example of how the correct candidate may be excluded from the set. In such cases, a soft constraint strategy may be the best choice.

Name	Description	Score	ID
Wellington School, Bebington	former private grammar school in Wirral, England	1.0	Q7981484
Wellington High School	high school in Wellington, Collingsworth County, Texas	0.959	Q17035369
Wellington High School	high school in Palm Beach County, Florida, United States	0.959	Q7981413
Wellington High School	Wellington, NSW - Government - Secondary - 292	0.958	Q101435838
Wellington High School	high school in Ohio, United States	0.958	Q7981414
Wellington High School	**state secondary (year 9-15) school in Wellington, New Zealand**	**0.954**	**Q7981412**
Bebington High School	academy in Bebington, Merseyside, England	0.908	Q4878436
Ellington High School	public high school in Connecticut, U.S.	0.908	Q5365418
Arlington High School	public high school in Riverside, California, United States	0.904	Q4792347
Arlington High School	secondary school in Arlington, Texas	0.902	Q4792340

Table 4.9: Candidate set with a **hard** strategy [*NERtype* - *NERtype*] filtering with ORG.

Chapter 5

Conclusion

5.1 Summary

Enhancing the quality of the candidate set composition is crucial in an Information Retrieval system. In particularly in the context of Entity Linking for tabular data it's important that the correct candidate is included into the candidate set. This thesis aims to improve the retrieval strategy within this scenario. The first objective is to provide a comprehensive review of state of the art filtering strategies used in Entity Linking frameworks. Most of these strategies incorporate type-aware filtering during the disambiguation phase, while ranking algorithms are commonly employed to select the most relevant candidates.

The tool used in this research is *LamAPI*, a label matching retrieval system integrated into the *Alligator* reconciliation system, which is based on the Wikidata knowledge base. *LamAPI* proposes candidate entities based on mentions found in tabular datasets.

The second objective is to develop a structured methodology for enhancing the explicit type hierarchy in Wikidata. This enhancement aims to improve the retrieval process in *LamAPI* by incorporating type-based filtering strategies. The evaluation process begins with measuring coverage and Mean Reciprocal Rank (MRR) across three different strategies. These strategies are based on two factors: the query type (mention type), which is tested using both explicit Wikidata (WD) types and Named Entity Recognition (NER) types, and the candidate type (the list of types associated with each candidate), tested using NER types and extended WD types. For each tested strategy, two experiments were conducted: one using a hard filter mode, where candidates must match at least one of their types to be included in the set, and another using a soft filter mode, where candidates'

scores are adjusted based on the degree of their type matches.

The experimental results show that a soft type-based strategy can outperform the baseline label matching in *LamAPI*, particularly when using the [*Explicit WD type - Extended WD type*] strategy. These findings remain consistent even when the dataset contains multi-token mentions, provided that misspellings and typos are limited.

When mention labels contain a high number of errors, as seen in the **2T** tabular dataset, neither hard nor soft constraints effectively improve retrieval performance.

HardTableR3, for example, contains labels with a low degree of misspellings. In this case, soft filters enhance ranking performance by dynamically updating candidate scores. However, hard type-based filtering is more restrictive and may eliminate the correct candidate.

In general for misspelled mentions the results have shown that soft type-based strategies are able to include the correct candidate in the set thanks to their score updating mechanism if the misspellings are reduced in numbers.

5.2 Future directions

The filtering strategies tested in this work are closely tied to the datasets used and the quality of the type extension algorithm. Future improvements to this project could explore several key areas.

One valuable improvement involves refining the label-matching strategy in *LamAPI* to better handle misspelled entity mentions, allowing the retrieval system to process noisy data more effectively.

Additionally, the type extension algorithm could be enhanced by integrating a transitive closure methodology into the NER type mapping process. This would lead to a more type-aware approach by ensuring that entity types are extended in a consistent and structured manner. A related research avenue involves adapting transitive closure techniques to address challenges within the Wikidata type hierarchy, such as resolving loops or managing incomplete type assignments

Additionally, incorporating more domain-specific datasets could lead to a more comprehensive understanding of the best use case scenarios for these filtering strategies.

A potential direction for future research is assessing the impact of this new retrieval strategy in *LamAPI* on the overall performance of the Entity Linking (EL) pipeline. Specifically, it would be valuable to investigate how the disambiguation phase in *Alligator* is affected after the candidates' ranking is modified during the retrieval phase.

These represent the most promising future directions that could enhance the analysis presented in this thesis.

6 Appendix

In this section two examples of queries are shown using the mention *Earl Smith*. The correct candidate is the following one:


```
{
  "_index": "wikidata",
  "_id": "32626712",
  "_score": 74.632805,
  "_source": {
    "id": "Q5326089",
    "name": "Earl Calvin Smith",
    "language": [
      "en"
    ],
    "is_alias": true,
    "description": "American baseball player",
    "kind": "entity",
    "NERtype": [
      "PERS"
    ],
    "explicit_WDtypes": [
      "Q5"
    ],
    "extended_WDtypes": [
      "Q103940464", "Q3778211", "Q5", "Q106559804", "Q223557", "Q35120",
      "Q795052", "Q7239", "Q110551885", "Q488383", "Q53617489", "Q53617407",
      "Q4406616", "Q10855152", "Q66394244", "Q26401003", "Q154954", "Q729",
      "Q12898224", "Q164509", "Q215627", "Q27043950", "Q72638", "Q24229398", "Q159344"
    ],
    "length": 17,
    "ntoken": 3,
    "popularity": 0
  }
}
```


Using the strategy [*Explicit WD type* - *Extended WD type*] and choosing for query type

the following set

$$Q = \{"Q31829329", "Q5", "Q60594690"\}$$

the query parameters for the *lookup* service of *LamAPI*, considering a **soft constraint**, is the following:

```
{
  "name": "Earl Smith",
  "token": "lamapi_demo_2023",
  "kg": "wikidata",
  "limit": 1000,
  "query": {
    "query": {
      "bool": {
        "must": [
          {
            "match": {
              "name": {
                "query": "Earl Smith",
                "boost": 2.0
              }
            }
          }
        ],
        "should": [
          {"term": {"extended_WDtypes": "Q31829329"}},
          {"term": {"extended_WDtypes": "Q5"}},
          {"term": {"extended_WDtypes": "Q60594690"}}
        ]
      }
    }
  },
  "sort": [
    {"popularity": {"order": "desc"}}
  ]
}
```

For the same query type set the parameters considering an **hard constraint** is the following:

```
{
  "name": "Earl Smith",
  "token": "lamapi_demo_2023",
  "kg": "wikidata",
  "limit": 1000,
  "query": {
    "query": {
      "bool": {
        "must": [
          {
            "match": {
              "name": {
                "query": "Earl Smith",
                "boost": 2.0
              }
            }
          },
          {
            "terms": {
              "extended_WDtypes": [
                "Q31829329", "Q5", "Q60594690"
              ]
            }
          }
        ]
      }
    }
  },
  "sort": [
    {"popularity": {"order": "desc"}}
  ]
}
```

REFERENCES

[1] P. Nguyen, N. Kertkeidkachorn, R. Ichise, and H. Takeda, "Mtab: Matching tabular data to knowledge graph using probability models," *arXiv preprint arXiv:1910.00246*, 2019.

[2] A. L. F. Shanaz and R. G. Ragel, "Named entity extraction of wikidata items," in *2019 14th Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2019, pp. 40–45.

[3] S. I. H. Shah, V. Peristeras, and I. Magnisalis, "Government (big) data ecosystem: definition, classification of actors, and their roles," *International Journal of Computer and Information Engineering*, vol. 14, no. 4, pp. 102–114, 2020.

[4] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2021.

[5] M. Marzocchi, M. Cremaschi, R. Pozzi, R. Avogadro, M. Palmonari *et al.*, "Mammotab: a giant and comprehensive dataset for semantic table interpretation," in *CEUR WORKSHOP PROCEEDINGS*, vol. 3320. CEUR-WS, 2022, pp. 28–33.

[6] M. Cremaschi, B. Spahiu, M. Palmonari, and E. Jimenez-Ruiz, "Survey on semantic interpretation of tabular data: Challenges and directions," *arXiv preprint arXiv:2411.11891*, 2024.

[7] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, "Named entity extraction for knowledge graphs: A literature overview," *IEEE Access*, vol. 8, pp. 32 862–32 881, 2020.

[8] F. Belotti, F. Dadda, M. Cremaschi, R. Avogadro, R. Pozzi, and M. Palmonari, "Evaluating language models on entity disambiguation in tables," *arXiv preprint arXiv:2408.06423*, 2024.

[9] T. Zhang, X. Yue, Y. Li, and H. Sun, "Tablellama: Towards open large generalist models for tables," *arXiv preprint arXiv:2311.09206*, 2023.

[10] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "Turl: Table understanding through representation learning," *ACM SIGMOD Record*, vol. 51, no. 1, pp. 33–40, 2022.

[11] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, and R. Troncy, "From heuristics to language models: A journey through the universe of semantic table interpretation with

dagobah," in *21st International Semantic Web Conference (ISWC 2022)*, vol. 3320, 2022.

[12] R. Avogadro, M. Ciavotta, F. De Paoli, M. Palmonari, and D. Roman, "Estimating link confidence for human-in-the-loop table annotation," in *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 2023, pp. 142–149.

[13] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, and V. Christophides, "Matching web tables with knowledge base entities: from entity lookups to entity embeddings," in *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*. Springer, 2017, pp. 260–277.

[14] V. Cutrona, G. Puleri, F. Bianchi, and M. Palmonari, "Nest: neural soft type constraints to improve entity linking in tables," in *Further with Knowledge Graphs*. IOS Press, 2021, pp. 29–43.

[15] R. Avogadro, M. Cremaschi, F. D'Adda, F. De Paoli, M. Palmonari *et al.*, "Lamapi: a comprehensive tool for string-based entity retrieval with type-base filters." in *OM@ ISWC*, 2022, pp. 25–36.

[16] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, and H. Takeda, "Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata." *SemTab@ ISWC*, vol. 2775, pp. 86–95, 2020.

[17] J. GeikJ.kg^{-1}K^{-1}, A. Spitz, and M. Gertz, "Neckar: A named entity classifier for wikidata," in *Language Technologies for the Challenges of the Digital Age: 27th International Conference, GSCL 2017, Berlin, Germany, September 13-14, 2017, Proceedings 27*. Springer, 2018, pp. 115–129.

[18] P. F. Patel-Schneider and E. A. Doğan, "Class order disorder in wikidata and first fixes," *arXiv preprint arXiv:2411.15550*, 2024.

[19] D. Roman, R. Prodan, N. Nikolov, A. Soylu, M. Matskin, A. Marrella, D. Kimovski, B. Elvesæter, A. Simonet-Boulogne, G. Ledakis *et al.*, "Big data pipelines on the computing continuum: tapping the dark data," *Computer*, vol. 55, no. 11, pp. 74–84, 2022.

[20] H. Sun, H. Ma, X. He, W.-t. Yih, Y. Su, and X. Yan, "Table cell search for question answering," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 771–782.

[21] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas, "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.

[22] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jimenez-Ruiz, J. Sequeda, and K. Srinivas, "Semantic web challenge on tabular data to knowledge graph matching," in *International Semantic Web Conference*, 2022.

[23] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, and V. Cutrona, "Results of semtab 2020," in *CEUR Workshop Proceedings*, vol. 2775, 2020, pp. 1–8.

[24] V. Cutrona, F. Bianchi, E. Jiménez-Ruiz, and M. Palmonari, "Tough tables: Carefully evaluating entity linking for tabular data (nov 2020)."