Web Table Annotation Using Knowledge Base

eXascale Infolab

MASTER IN COMPUTER SCIENCE

Master Thesis

Yasamin Eslahi

July 2019

Thesis supervisors:

Prof. Dr. Philippe Cudré-Mauroux
Akansha Bhardwaj
and Paolo Rosso

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

UNIVERSITÄT BERN

UNIVERSITÉ DE NEUCHÂTEL

Acknowledgements

I want to thank all the people that helped me, my supervisor, Prof. Dr. Cudré-Mauroux for his support. My special thanks to Akansha Bhardwaj and Paolo Rosso for their advice and understanding throughout all stages of the work. Thank you for everything.

Abstract

This master thesis describes how to automatically link the structured information in Web tables to the Knowledge Base. The Web has over 100 million tables which provide a valuable source of relational data. These tables can be used for Web table search and Knowledge Base augmentation if we understand them semantically. Lack of a unique schema for Web tables makes it a challenge to use this useful source of information and to convert the Web tables into a machine-understandable knowledge. Additionally, lack of table header for a vast majority of these tables and the ambiguity of the table entities are considered as a barrier for this task. *Web table annotation* aims to solve this problem by presenting additional information for the HTML tables and defining their entities by the Web Knowledge source embedded in the Knowledge Bases.
Our work presents two unsupervised approaches to tackle this problem. First, we study the use of Lookup-based methods as a scalable solution and second, we use the word Embeddings to understand the semantics of the Web table and provide the corresponding row annotations. We show that our proposed method significantly outperforms the existing Semantic Embedding methods. Moreover, we compare different Node Centrality measures which can not only be used in Web Table annotations, but also for any graph-based applications. Finally, we contribute by presenting an updated version of Limaye Gold Standard with annotations from DBpedia.

Keywords: Web Table Annotation, Semantic Annotation, Data Mining, Entity Disambiguation

Table of Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1. Motivation

Human beings can learn and understand new concepts by using the data that they have seen. There is no doubt in the importance of the input information for the process of understanding. To understand any word, we need the context. Previous knowledge helps for better understanding.

The same holds for all the sources of information on the Web. We need context to understand it. In case of text understanding, we have surrounding text as the context. We can use the keywords in the same paragraph or the whole text to get a general idea. The machine must take the same steps to understand any text inputs. Extraction of the contextual keywords gives a general idea. These words are usually around the same topics.

Apart from the text on the Web, Web Tables are also a valuable source of information. In the ideal case, we would have the Web tables, with the correct headers, and many rows with correct values inside. We could understand and use this useful information but in real Web Tables, we do not always have the headers, or there are typos or empty cells. Later in this chapter, we state the problem, and we see why it is difficult to extract the information from Web tables and what are the challenges to benefit from this structured data.

Film Title	Released	Studio	Worldwide Gross	Domestic Gross
Frozen	2013	Disney	$1,072,402,000	$398,402,000
Toy Story 3	2010	Disney Pixar	$1,063,171,911	$415,004,880
The Lion King	1994	Disney	$987,483,777	$422,783,777
Despicable Me 2	2013	Universal	$970,761,885	$368,061,265
Finding Nemo	2003	Pixar	$936,743,261	$380,843,261
Shrek 2	2004	DreamWorks	$919,838,758	$441,226,247
Ice Age: Dawn of the Dinosaurs	2009	Fox	$886,686,817	$196,573,705
Ice Age: Continental Drift	2012	Fox	$877,244,782	$161,321,843
Shrek the Third	2007	DreamWorks	$798,958,162	$322,719,944
Shrek Forever After	2010	DreamWorks	$752,600,868	$238,736,787

Tab. 1.1.: Illustration of the ideal Web table

Table 1.1 shows an example of an ideal Web table that contains the header information on each column, and the entities have the correct dictation. Also, as we can observe, the table contains both numeric and textual values. Any combination of numbers, letters, and symbols are possible. In this table, by looking at the first column, we can get the general topic of the table. More details are available in the rest of the columns. For the first row, we have a computer-animated musical fantasy film. To explain the *Film Title* column, we use the other attributes of the table as the released date, the producer, and its gross. By the end, we can decide as a human that this table demonstrates the animated-films information.

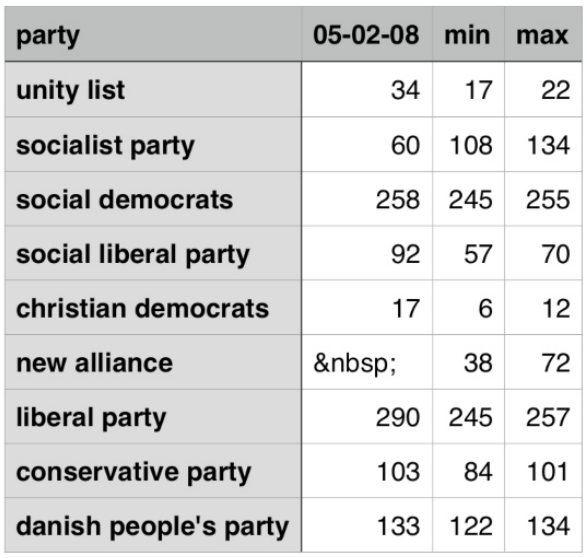
1.2. Web Table Annotation

The Web contains named entities such as people, places, organizations, and products and they can be ambiguous [11]. In this thesis, we specifically deal with the named entities in the Web tables, and our goal is to disambiguate the entities and give the annotation. To annotate each of the entities mentioned in the Web table, we connect them to the Knowledge Base.

Knowledge Bases (KB) contain rich information about the world's entities, their semantic classes, and their mutual relationships. Such kind of notable examples include DBpedia [16], YAGO [26], Freebase [3] and Probase [6] [23]. Depending on the Knowledge Base, each entity has a specific form of a unique link or ID. In our example, DBpedia defines the entity, *Frozen*, as `http://dbpedia.org/page/Frozen_(2013_film)` and Wikidata defines it as `https://www.wikidata.org/wiki/Q386724`. In this thesis, we use Wikidata [33] to annotate entities. The used gold standards contain DBpedia links. Later in Chapter 4, we provide a mapping file with DBpedia links with their equivalent links in Wikidata.

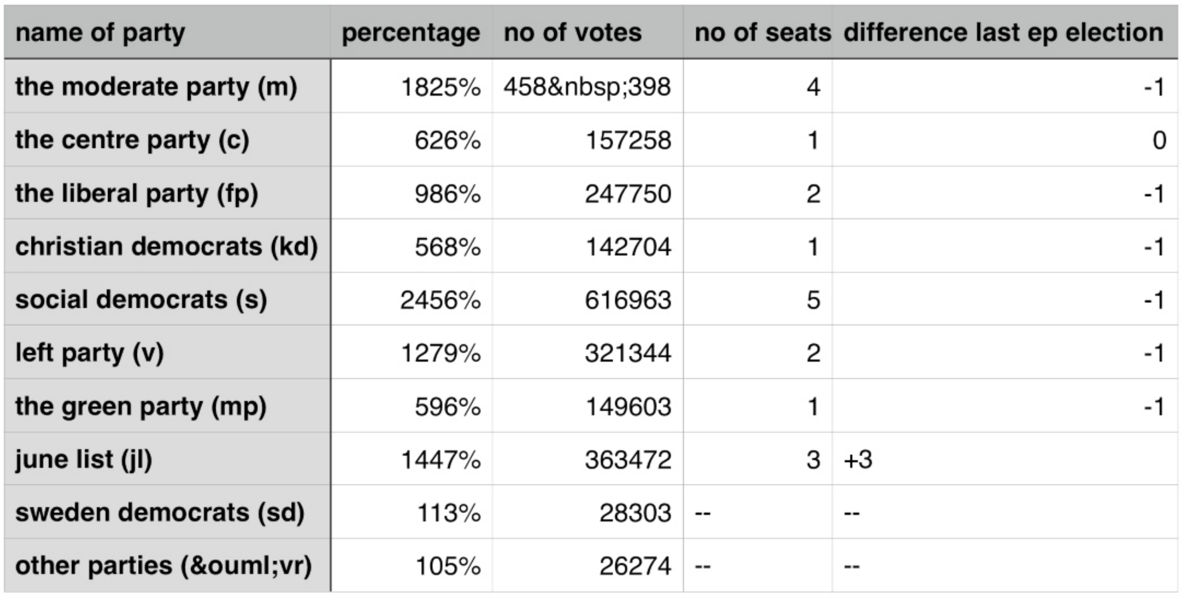
Having these definitions in mind, we define Web Table Annotation as the task of extracting the rows of Web tables and linking them to semantically rich descriptions of entities published in Web Knowledge Bases [5].

Without the header row, it is hard for the machine to understand the main topic of the table. What makes this task even more difficult is the disambiguation of the same name among its different possible meanings. As an example, in Table 1.1, the entity *Fox*, can be annotated as a mammal, `http://dbpedia.org/page/Fox`, a TV series, `http://dbpedia.org/page/Fox_(TV_series)`, a family name, `http://dbpedia.org/page/Fox_(surname)`, etc. In DBpedia, all the possible disambiguations of the entities are accessible on their disambiguation page. For this example, there exist 89 different disambiguations at `http://dbpedia.org/page/Fox_(disambiguation)`. To disambiguate an entity in a row, we can use the information in the other columns. It is also possible to have only one column in a table. In this case, we may use the other rows of the Web table to decide about the correct disambiguation.



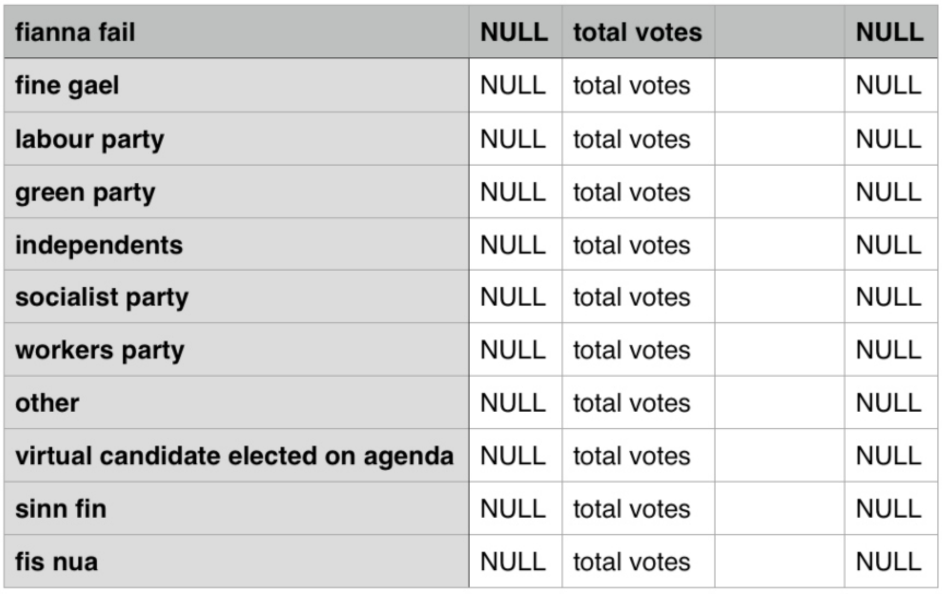
party	05-02-08	min	max
unity list	34	17	22
socialist party	60	108	134
social democrats	258	245	255
social liberal party	92	57	70
christian democrats	17	6	12
new alliance		38	72
liberal party	290	245	257
conservative party	103	84	101
danish people's party	133	122	134

Tab. 1.2.: Political parties in Denmark from T2D dataset



name of party	percentage	no of votes	no of seats	difference last ep election
the moderate party (m)	1825%	458&nbsp;398	4	-1
the centre party (c)	626%	157258	1	0
the liberal party (fp)	986%	247750	2	-1
christian democrats (kd)	568%	142704	1	-1
social democrats (s)	2456%	616963	5	-1
left party (v)	1279%	321344	2	-1
the green party (mp)	596%	149603	1	-1
june list (jl)	1447%	363472	3	+3
sweden democrats (sd)	113%	28303	--	--
other parties (&ouml;vr)	105%	26274	--	--

Tab. 1.3.: Political parties in Sweden from T2D dataset



fianna fail	NULL	total votes		NULL
fine gael	NULL	total votes		NULL
labour party	NULL	total votes		NULL
green party	NULL	total votes		NULL
independents	NULL	total votes		NULL
socialist party	NULL	total votes		NULL
workers party	NULL	total votes		NULL
other	NULL	total votes		NULL
virtual candidate elected on agenda	NULL	total votes		NULL
sinn fin	NULL	total votes		NULL
fis nua	NULL	total votes		NULL

Tab. 1.4.: Political parties in Ireland from T2D dataset

An entity can have different names. For instance, the bird *Green sandpiper*, `http://dbpedia.org/page/Green_sandpiper`, is also called *Tringa ochropus*. Also, an entity's name may change over time [5]. It is a challenge from our early stages of entity disambiguation and entity annotation. We search for the entity to get the initial candidates. If the entity name in the Web table is different from its equivalent page title on Knowledge Base, we do not have access to the correct initial candidates. If there is a difference between the entity in the table and the Knowledge Base, we use an edit distance function such as Levenshtein. Levenshtein edit distance can be useful if there is a typo in the Web table. In our implementation, we accept the similarity ratio of more than 0.85. If the entity *Green sandpiper* was written as *Green sandpip*, the similarity ratio would be 0.92, which is more than our acceptance ratio. When we have two different names for one entity, this solution cannot help. For the entities *Green sandpiper* and *Tringa ochropus*, the similarity ratio is 0.27, and we cannot solve this problem with the edit distance function.

On the other hand, different concepts may be expressed using the same words. Table 1.3, Table 1.2, and Table 1.4 show the HTML tables of political parties in Sweden, Denmark, and Ireland, respectively. *Christian democrats*, *social democrats*, and *liberal party* are the three entities that exist in both Table 1.2 and Table 1.3, although they represent different concepts. Since other columns of these rows are numeric values, we do not use them for disambiguation. To decide about the correct disambiguation, we should either have the table caption or use the information stored in the other rows. In Table 1.2, the entity *danish people's party* can help us to understand that the table is about Denmark. The entity *Sweden democrats (sd)* helps us to understand the topic of Table 1.3. The same problem appears in Table 1.4 for the *green party* and the *socialist party* entities. The entity *green party* exists in Table 1.3, and the entity *socialist party* exists in Table 1.2 too.

1.3. Objectives

In this work we describe the state of the art approaches for Web Table Annotation. First, we give a review on the most common approaches. Then, inspired by a recent paper by Efthymiou et al. [5], we introduce a new method to achieve better results. Before describing our approach, we discuss the modifications that we have made to adapt the baseline to our approach.

1.4. Thesis Layout

The rest of the thesis is as follows:

Chapter 2 presents the recent works on Web Table Annotation and describes the common ideas among the existing solutions. We describe the fundamentals to understand the thesis as well as the state-of-the-art methods.

Chapter 3 covers the proposed methods to disambiguate entities in Web tables. We describe the lookup-based and semantic embeddings approaches, and their combination as well. This chapter contains our baseline methods and our proposed approaches.

Chapter 4 contains the benchmark data sets and brief representation of the available files in each data set. The main purpose of this chapter is to show the results of the implemented methods on each of the data sets. At the end, we mention some useful experimental setup that we exploit for this work which are useful for the efficient execution of the code.

Chapter 5 concludes the thesis and talks about possible future works.

2
State of the Art

2.1. Introduction

In this chapter, we provide an overview of the existing solutions for *Web table annotation*. We compare different techniques in look-up based methods as well as semantic embedding methods. This chapter covers the background techniques on which we base our own propositions and improvements.

The first step before annotating a Web table is to understand what is it about and which entities, columns, or rows, are useful for the annotation task. [1] believes there is a column in every Web table, which demonstrates the main purpose of that table. Other papers like [21], [28], and [5] follow the same idea. Venetis et al. [28] specify the *subject column* as the subject of the Web table, which contains the set of entities the table is about, where the other columns represent binary relationships or properties of those entities. To detect the label column, they labeled a set of tables to train a model. Both the concept identification for columns and relation identification are based on the maximum likelihood hypothesis. On the other hand, Efthymiou et al. rely on a heuristic method [5]. The leftmost column with the maximum number of distinct non-numeric values is the label column. Although the solution for finding this most important column can vary, these approaches agree on the importance of such a column.

To annotate a Web table, we can map the rows, columns, or the individual cells to the Knowledge Base. In the literature, there are several approaches [2, 14, 29], in which the authors have tried to provide annotations for the web table cells. On the other hand, other approaches like [5] have connected each row to one of the entities in the Knowledge Base. In this thesis, we annotate each row by a Wikidata page. In the context of the Web Lists, the entities have the same type. It is the same scenario as in the Web table columns. Shen et al. propose a framework to link the entities of the web lists with the Knowledge Base [24]. Ritze et al. combine rows and columns matching to the entities and schema of DBpedia [21]. Fan et al. use the table columns to map them to one or more concepts in the Knowledge Base [7].

Much work on the potential of supervised methods has shown striking results [2, 14, 24, 1]. Also, employing the crowd-sourcing methodologies for difficult tables is an interesting approach [7], but, we believe that the use of human judgment to help the annotation reduces the generality of the methods [5]. One of the significant drawbacks of adopting supervised methods is the unscalability of such methods. An alternative solution could be to leverage the more scalable approaches like unsupervised methods. Zhang et al. [29] and Efthymiou et al. [5] perform the annotation without any previous hypotheses about the input data and provide unsupervised solutions. In this thesis, we do not make any assumptions about the input data.

2.2. Lookup-based Method

2.2.1. Candidate Generation

In the initial stages of annotation, we should have some candidates for each entity of the table. There exist services like DBpedia lookup service[1] to find the possible candidates for the annotation of each entity. These candidates are refined or enriched based on the different methodologies presented in each paper. [21] uses DBpedia lookup service to find related DBpedia URLs for each keyword. It also provides auto-completion of the keywords if the string is incomplete. [29] uses Freebase for candidate generation phase. This service is now deprecated. In the context of entity disambiguation in text documents, [27] uses *Surface Forms* to prepare a list of unrefined entity candidates from all the sources. It applies the string normalization approach, like removing suffix and prefixes for some specific labels, removes time stamps, plural forms, etc. to clean the input. [31] performs trigram similarity between the index and normalized surface form for candidate generation.

Apart from the already existing lookup services, several existing works define custom candidate generation techniques. Efthymiou et al. [5] create their FactBase method over Wikidata entities. Next, we explain in detail the FactBase Lookup method. To get the first possible results, [1] looks for each entity in a database from the Knowledge Base with the format as (value, entity, score).

Wikification is the process of annotating the mentions of concepts in a document with their corresponding URL of the Wikipedia [13]. In Wikification approaches, a system generates a ranked set of possible candidates for a given mention by querying Wikipedia [4, 10, 18]. The standard approach for doing so is to collect all hyperlinks in Wikipedia and associate each hyperlinked span of text with a distribution over titles of Wikipedia articles it is observed to link to.

The topical coherence was first used by Cucerzan for disambiguation of entities [19]. A large scale named entity disambiguation system uses the contextual and categorical information extracted from Wikipedia and the context of the document. To achieve a surface form for entity matching, the title of the entity page, as well as the reference to entity page in other Wikipedia articles are used. Milne and Witten [18] improve topical coherence by normalized Google distance. They use only the unambiguous entities in the document to get the context. The disambiguation is done simultaneously to keep the coherence among each disambiguated set. The proposed methods by Han [10, 9] utilize all words of Wikipedia to learn entity-word association and category hierarchy in Wikipedia to capture co-occurrence of the entities.

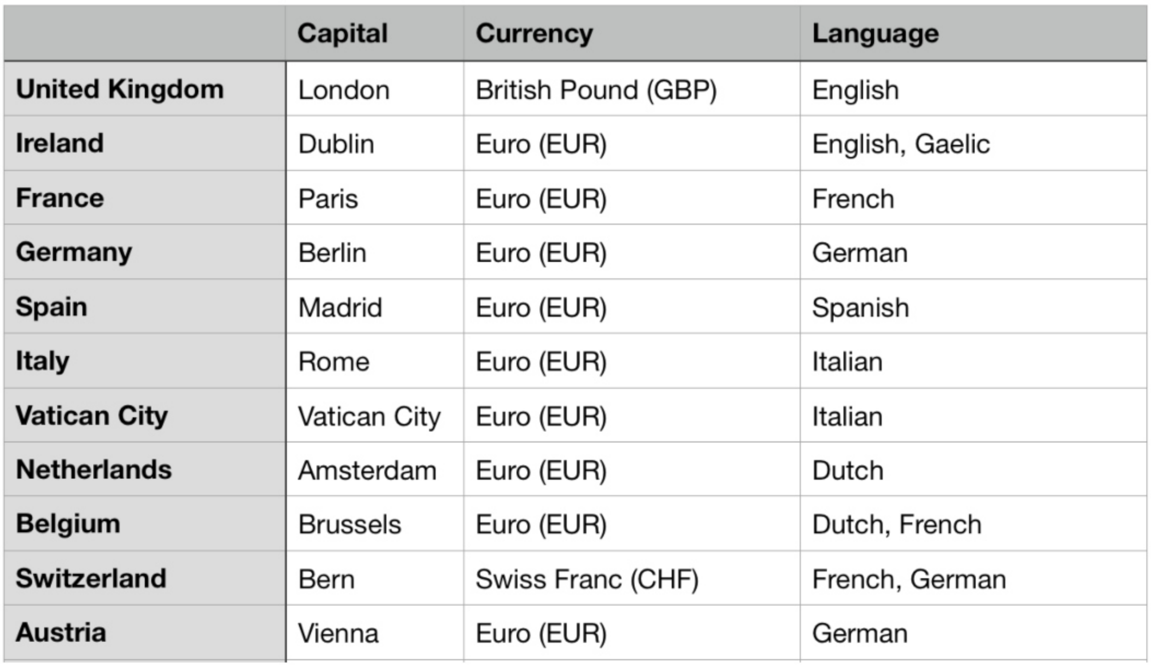
[1]http://wiki.dbpedia.org/lookup/

Shen et al. [25] perform the task of entity linking by unifying Wikipedia and WordsNet[2] Knowledge Bases. To generate the surface form, they use dictionaries of entity pages, redirect pages, disambiguation pages and hyperlinks in Wikipedia articles and use their counts to define the link probabilities. [30] is a ranking-based disambiguation system with the domain-specific evaluation on bio-medical entity disambiguation. Their focus is on the bio-medical domain to analyze the influence of user data on disambiguation result. The top three properties that took into consideration are entity context, user data, entity quantity, and entity heterogeneity.

2.2.2. Columns' Relations

Even though in most situations having a label column can be enough to annotate a table, its binary relations with reference columns may appear relevant too. Reference columns describe the properties of the label column. If there is any relation between the columns, the same pattern should exist for different rows of the Web table. For example, Table 2.1 shows a sample table of the countries. *Capital*, *currency*, and *language* columns are the properties of the column *country*. For the entity *United Kingdom* (`https://www.wikidata.org/wiki/Q145`), there is a *capital* relation (`https://www.wikidata.org/wiki/Property:P36`) with the entity *London* (`https://www.wikidata.org/wiki/Q84`). Since the same pattern exists for the other rows of this table, we can extract the relation *capital* between the first and second column [5]. [1] states that it is not easy to find these relations only by the table itself so they are using the surrounding content of the Web table to extract relations. Surrounding information contains a paragraph before and after the table as well as its caption. To extract relations, [28] uses a database of relations, and [1] uses a dictionary of attributes from the Web text and search queries. Moreover, authors in [22] augment the Knowledge Base by levering a Web text corpus and natural language patterns associated with relations in the Knowledge Base.

[2]https://wordnet.princeton.edu/



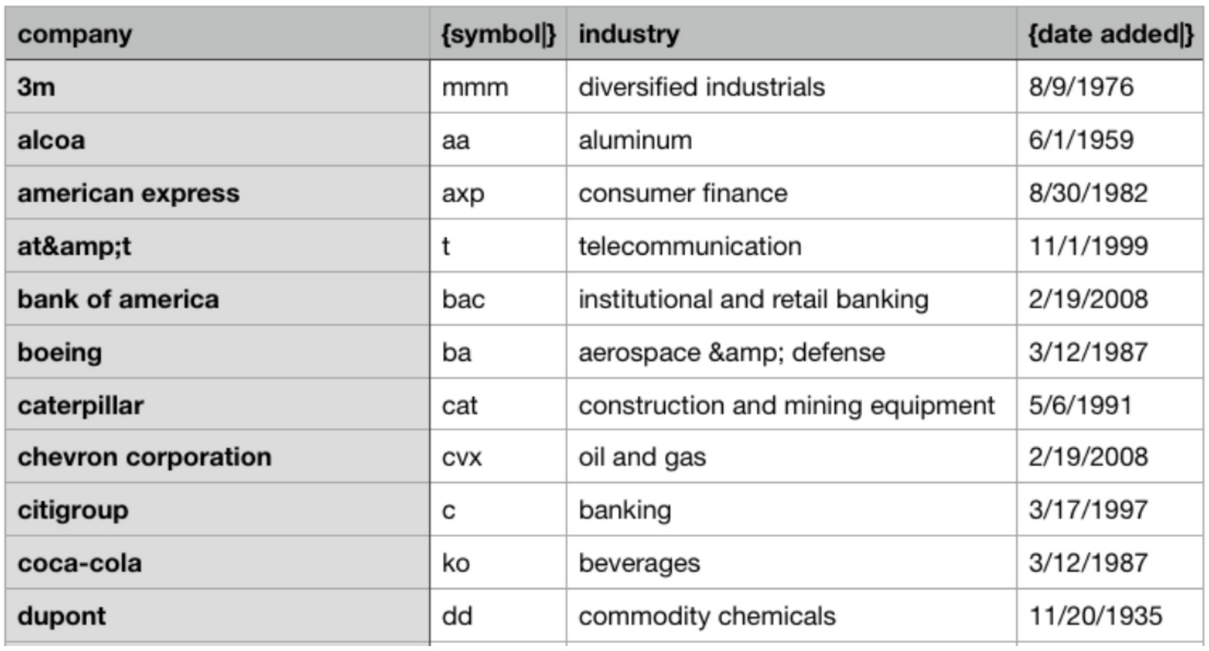
	Capital	Currency	Language
United Kingdom	London	British Pound (GBP)	English
Ireland	Dublin	Euro (EUR)	English, Gaelic
France	Paris	Euro (EUR)	French
Germany	Berlin	Euro (EUR)	German
Spain	Madrid	Euro (EUR)	Spanish
Italy	Rome	Euro (EUR)	Italian
Vatican City	Vatican City	Euro (EUR)	Italian
Netherlands	Amsterdam	Euro (EUR)	Dutch
Belgium	Brussels	Euro (EUR)	Dutch, French
Switzerland	Bern	Swiss Franc (CHF)	French, German
Austria	Vienna	Euro (EUR)	German

Tab. 2.1.: Table of Countries From T2D dataset

2.2.3. FactBase Lookup

FactBase is a lookup-based method for entity annotation and disambiguation [5]. The approach is divided into two phases. In the first phase, we iterate over the whole table to extract the information about the entities of the Web table. This phase only annotates the unambiguous entities. For each entity of the label column, they extract the top possible candidates in the Knowledge Base. At the end of this iteration, they choose the Acceptable types. Acceptable types are the top-5 most frequent types of each column. They extract the entity description tokens from the Knowledge Base. By the end of the iteration, they choose the top description token of the table. Binary relations of the column are the essential concept to extract in this phase.

In the second phase, these relations help to choose the correct annotation. Till here, many rows can be already annotated. For the rest of the rows, two situations hold. First, If there are many possible candidates, they perform a *Strict search*. They choose the candidates who have the acceptable types, and one of the description tokens in their entity description. Second, if no candidate exists, they let a bigger Levenshtein margin distance. For instance, Table 2.2 from T2D dataset shows the companies and their industry. The entity *coca-cola* is titled as *The Coca-Cola Company* (`https://www.wikidata.org/wiki/Q3295867`) in Wikidata. If we look for the entity *coca-cola*, we fail to find its correct candidate (Q3295867). To solve this problem, we accept the candidates with more than 0.85 similarity ratio. This solution helps to find the correct candidate, even if we have typos, nicknames, abbreviations, etc. If we let a bigger Levenshtein margin distance, candidates should contain one of the extracted binary relations. This method is referred to as *Loose search*.



company	{symbol|}	industry	{date added|}
3m	mmm	diversified industrials	8/9/1976
alcoa	aa	aluminum	6/1/1959
american express	axp	consumer finance	8/30/1982
at&amp;t	t	telecommunication	11/1/1999
bank of america	bac	institutional and retail banking	2/19/2008
boeing	ba	aerospace &amp; defense	3/12/1987
caterpillar	cat	construction and mining equipment	5/6/1991
chevron corporation	cvx	oil and gas	2/19/2008
citigroup	c	banking	3/17/1997
coca-cola	ko	beverages	3/12/1987
dupont	dd	commodity chemicals	11/20/1935

Tab. 2.2.: Table of Companies From T2D dataset

2.3. Embedding methods

The core hypothesis of linking approach for text disambiguation task is compatible with Web table annotation task [5]. In text disambiguation, entities are mainly about the same context. Web tables entities also create the coherent set of concepts. DoSeR is one of the linking approach for text documents [31, 32] that employs the Semantic Entity Embeddings on different types of Knowledge Bases, whether RDF-based Knowledge Bases or entity-annotated Knowledge Bases.

DoSeR [31] starts by index generation. The index contains a label, a *semantic embedding*, and a *prior* for each entity. A prior demonstrates the importance of an entity within the text. The entities are gathered from different combinations of Knowledge Bases. Given an input document, the semantic embedding, created on all the entities of a Knowledge Base, help to calculate the semantic similarity of the entities. Candidates are achieved either with an exact match to the surface form or trigram similarity. Finally, by having the semantic embeddings, DoSeR creates a K-partite graph. This complete directed graph has K disjoint subsets. In the context of web table annotation, each subset can be considered as each row (or label column entity). The nodes are the already filtered entity candidates, and edge weights are the normalized cosine similarity of the Semantic Embeddings of the entities. Also, the obtained prior values helps to connect nodes as non-uniformly-distributed random jump values. By applying PageRank with 50 iterations, the node with the highest relevance is the result. The final improvement is made by removing the 25% of the edges, whose sources and target entities have the lowest semantic similarity.

There are some other approaches for text entity disambiguation as DBpedia Spotlight [16], TagMe [8], Wikifier [13], AIDA [12] and WAT [6]. DBpedia Spotlight [16] annotates the text documents using the DBpedia Lexicalization dataset. It starts by candidate selection to reduce the space of disambiguation possibilities, and the candidate whose context has the highest cosine similarity is chosen. Disambiguation is done using vector space model for DBpedia resources. In DBpedia Spotlight, no semantic coherence is estimated among the chosen entities.

TagMe is another graph-based global disambiguation method for text documents [8]. The process is based on the disambiguation of the terms by hyperlinks to the Wikipedia pages. It looks for the Wikipedia anchors in the text, and by having anchors list, TagMe allocates a Wiki page to the anchors. In the end, by having all the anchors' pages, using the in-link graph, it computes the score of the collective agreement among the chosen pages.

We have discussed the Wikification methods previously. Wikifier[13] is another Wikification approach which focuses more on statistical methods. Same as TagMe, Wikifier starts by candidate generation, followed by ranking the candidates of the mentions. After associating the Wikipedia pages to the terms, the learned model refines the candidates through the second phase. The difference of this method is a richer analysis of the text by statistical methods. Another graph-theoric method, AIDA[12] framework combines three measures: the prior probability of the mentioned entity, the similarity between the context of a mention and candidate entity, and the graph-based coherence among candidate entities. WAT [20] is the successor of TagMe, which reimplements all the components of TagMe for collective entity linking(graph-based) and local entity disambiguation(vote-based) algorithms.

What we have explained from the global disambiguation techniques for text disambiguation can be applied to the Web table entity disambiguation as well. [5] believes this method can be adapted for the Web tables because in Web tables, just like the text documents, the entities are forming a coherent sets. In Table 2.1, for the entity *United Kingdom*, we may have two candidates as the country (`https://www.wikidata.org/wiki/Q145`) and the music album (`https://www.wikidata.org/wiki/Q7887906`). Since the entities are coherent, the correct disambiguation should be related to the entity *London* (`https://www.wikidata.org/wiki/Q84`), and other country names in the following rows. Among these two candidates, the United Kingdom with the type as country (`https://www.wikidata.org/wiki/Q6256`) is chosen by this disambiguation technique. As in DoSeR, [5] uses Word2Vec to create semantic word vectors. The annotation in this method is divided into two parts. The first stage, called the off-line stage, provides the prerequisites for the online stage to find the correct semantic mapping between table rows and entities in a Knowledge Base. One of these prerequisites is the Surface Form. Surface form index receives the named entity as an input and provides some possible mappings to the Knowledge Base. It supplies all the existing entities of the Knowledge Base. Also, we create a Word2Vec model to compute the Cosine similarity of various pairs of entities.

We first perform a random walk from the entities to their neighbors in the Knowledge Base and use it as an input corpus for the Word2Vec model.

The online stage needs both created model as well as the surface index form. For each non-numerical value of the Web table, we search for the candidates, i.e., we look for the string entity name and get the candidates' unique IDs or URLs. These candidates become suitable nodes to create our K-partite graph. Next, the weighted directed edges connect the candidates of different entities. The candidates coming from the same table cells are not connected. The final step to form our disambiguation graph is to calculate the weight of the edges. The weight is the normalized Cosine similarity, measured from the semantic similarity of two nodes in Word2Vec model. We need a method to find out the importance of the nodes. Giving the graph to the PageRank algorithm, we find the most relevant nodes. PageRank algorithm computes the ranking dictionary of the nodes, and the node with the highest score is our chosen candidate.

3 Method

3.1. Lookup-based Methods

3.1.1. FactBase Method

The basic idea behind lookup-based method that we are presenting in this section is derived from a paper by Efthymiou et al. [5]. Their method for table annotation is inspired from existing papers such as TableMiner [29] for sampling phase and T2K [21] for the candidate generation phase. Also, the relation extraction method is motivated from another approach by Venetis et al.[28]. Some of these methods are presented for text disambiguation and they are extended to suit the Web table annotation.

To start the annotation, based on FactBase Lookup, we need to prepare some pre-requisites. First, we need a lookup service that provides common names for the entity that we want to annotate. More precisely, we need a lookup service to generate the first set of candidates. In our case, we use the surface index form generated from Wikidata. Because of the typos, or different ways of expressing the same concept in the dataset and the Knowledge Base page, we cannot match some of the table entities to any entity in our surface form. In this case, we also create the Levenshtein version of the surface form for the entities that are not found [5]. This file is generated before starting the Web table annotation.

In the first step, we iterate through the web table. On this first iteration, we perform a cleaning step which includes removing punctuation symbols and non-textual columns, and replacing HTML entities with their corresponding characters. For example, we replace *nbsp;* with its corresponding value which is a non-breaking space. Then, the Web table is ready for label column and reference columns detection. Label column, which contains the main subject of the table, is determined by the column with the most number of distinct non-numeric values. In case there are multiple columns with the same number of unique values, we choose the leftmost column as the label column. All the other columns of the Web table are the reference columns.

Two components are essential in this phase: first, the types of the entities in the label column, and second, the description tokens of these cells. For each row of the Web table, we take the label column entity and look for its candidates in the surface form. If there exists any result from this search, we sort the candidates according to an ascending order of the identifiers, and we keep the first result as our top result. Now that we have the top result of each row, we extract the types and the description tokens of this top result from its corresponding page in Wikidata. In Wikidata, the description tokens are from the "schema:description" property. Also, the types are mentioned in the property as "InstanceOf (wdt:P31)".

From now on, we may have three different paths to go through. First and the most accurate one is when we have only one candidate for an entity. So, the entity is not ambiguous, and we can annotate the row with this result. We try to find the relations of this annotated entity with the reference columns. We crawl the annotated page in Wikidata and look for the reference column value in it. There is a relation between the label column and a reference column if there is any property that mentions the reference column value. For example, in Table 2.1, *Capital*, *currency*, and *language* columns are the properties of the column *country.* For the entity *United Kingdom* (`https://www.wikidata.org/wiki/Q145`), there is a *capital* relation (`https://www.wikidata.org/wiki/Property:P36`) with the entity *London* (`https://www.wikidata.org/wiki/Q84`).

The second path is when the entities have more than one candidate. We define the acceptable type as the most frequent type among all collected types. In our case, we keep the top five types as acceptable types. Also, from all description tokens, we remove the stop words. We choose the most frequent tokens as description token of the table. For the entities with more than one candidate, we perform a more strict search. From the list of candidates, we accept the candidates with at least one of the acceptable types which also contains the selected description token. The unambiguous rows are already annotated. If we have more than one result from our Strict search, we take the first result from the sorted list of results. Finally, the third path is when there is no result from Strict search. It implies that we are too strict. To solve this, we perform another search with a lesser strict criteria called *Loose search.* In our Loose search function, we accept results with one of the extracted binary relations.

3.1.2. Majority-Based Method

The lookup-based method explained in the previous section mostly relies on the top result from the list of Knowledge Base candidates. With this method, we are missing the significant part of the available information. Although the first result has high importance for the annotation, it is not always the case that it is the correct one.

We define the majority-based method to get benefit from the other candidates. In this case, we do not only choose the acceptable types from the top result of each entity but we also select the types in the other available candidates. In other words, we query the types of all the entity candidates, we store them and append to the type collection of all the table. Finally, by majority function, we choose the top n most frequent types among them. It means we take the other candidates into account that can be our correct results.

3.2. Embedding Methods

The basic idea of all the embedding methods in this section is from our baseline paper [5]. The general idea comes from the global disambiguation techniques mentioned previously in [31]. The point is in local disambiguation techniques; we do not keep track of the context in the document. For example, in a sentence as "Sydney cannot be interesting for the kids.", we do not know whether Sydney is a city in Australia(Q3130), a community in Canada (Q932261), or an American comedy series(Q3979019) if we are not aware of the context around our sentence. In global disambiguation techniques, we believe that the context of the entities is the same. The global disambiguation technique was firstly proposed for the text document disambiguation. It can also be extended for the Web Table Annotations since the information in the table is mostly around the same topic.

3.2.1. Baseline Embedding Method

Before beginning the annotation, we need to create two main elements. First, the surface index form is required to find the primary candidates. [5] creates these possible candidates by using the common names of each entity. These candidates are the unique pages on DBpedia. This data is enriched with some of the available properties from the Knowledge Base. In our case, we are collecting the set of common names from Wikidata pages. To get our initial candidates, we apply a pre-processing step to our surface form creation. To be specific, we remove punctuation symbols and transfer all the surface form keys to lowercase letters. In this stage, stemming and stop word removal is not applied.

Apart from having our surface index form, we create the Levenshtein distance version of the surface form separately for each of our datasets. We crawl our Web tables and look for each entity in the surface index form. If the key does not exist, we calculate the Levenshtein distance of this key with all the keywords that we have in the surface index form. If we find any key with the similarity ratio of more than t, we store the not founded key with the candidate results of its similar key. It helps us to overcome the typos or any small mismatches in the entity and surface form key.

Next, we need to prepare the word embedding model. As in [5, 31], we use Word2Vec model presented by Mikolov et al. [17]. The process starts by applying a random walk on the Wikidata pages and following the RDF relations. Whenever there is a pattern match between two entities, they are added to the matching group of entities. These stored trigrams are consumed by Word2Vec to generate our model. The continuous bag of words (CBOW) is instantiated for the model. The parameter *min_ count* of the model is the minimum required repetition of a particular word in our corpus. We calculate our final results based on different min_counts for the words in our generated document. Different results are presented later in the next chapter for different minimum counts of words.

In the next step, we can start to annotate the entities using both self-generated surface form and the Word2Vec model. We iterate the Web table and collect the entities. At this stage, we do not consider any numerical value, except if they appear with string characters. As an example, we keep the entity *alex rodriguez (172)* which appeared in our T2D dataset. There are also a considerable amount of columns with repetitive prefix or suffix. In other words, the prefixes or the suffixes of all the entities in these columns are the same. These columns mostly do not have any value to help our annotation and can sometimes bring problems for detecting the correct label column. As an example, entities followed by their units like *401.00 mio m3z* or *100 m.* These measures are repeated in the whole column, and we need to remove them in the pre-processing phase. Moreover, we only consider the entities with more than 3 characters and as a result, an entity like *CHF* presenting the currency of money, *Swiss Franc*, is not taken into account.

We search each of the collected entities into our surface index form or in Levenshtein version of the surface form to have the possible candidates for the annotation. For each entity, we have a set of candidates V. In case of Wikidata, we omit all the candidate results with the property *Instance Of* (P31), *Wikimedia disambiguation page*, or *Wikimedia category.* The elements inside the set V, are used as the nodes of the disambiguation graph. The disambiguation graph is the union of all these candidates for the entities throughout our table. For example, if we have two entities, e1 and e2, we search each of them in our SF, and we have such a result, e1 = (id1, id2) and e2 = (id3, id4, id5). These ids are the nodes of the disambiguation graph. Each entity candidate is connected to all the candidates of the other entity. In here, there are edges from id1 to id3, id4, and id5. These connections are the direct weighted edges. We calculate the weights by the Normalized Cosine similarity obtained from their vector representations in our Word2Vec model.

$$weight(v1, v2) = \frac{cos(emb(v1), emb(v2))}{\sum_k cos(emb(v1), emb(k))}$$

(3.1)

Finally, we apply the PageRank algorithm to graph G to compute the most important node among all of the candidates of a specific entity. We can decide whether to choose the highest-ranked vertices or the set of top results of each mention. In the baseline method, the highest-ranked node is used, but in our methods, we use the top results from the ranked results of the PageRank.

3.2.2. Looping Method

The approach in section 3.3.1 is to create our disambiguation graph, with all the candidates of the entities. It means that there exist entities with only one candidate from the Surface Form, which is mostly their correct annotation and also entities with several candidates, which we should disambiguate. In the Looping method, we build an initial graph with the already annotated entities. Throughout the iterations, we gradually add ambiguous entities. We believe this method performs better because we strongly focus on the coherency of the nodes.

We collect the candidates, as described in section 3.3.1. Next, we create the initial *looping graph* with the entities with only one candidate. At each round, we add one ambiguous nodes' candidates to the Looping graph. An example of such a scenario is presented in Figure 3.1.

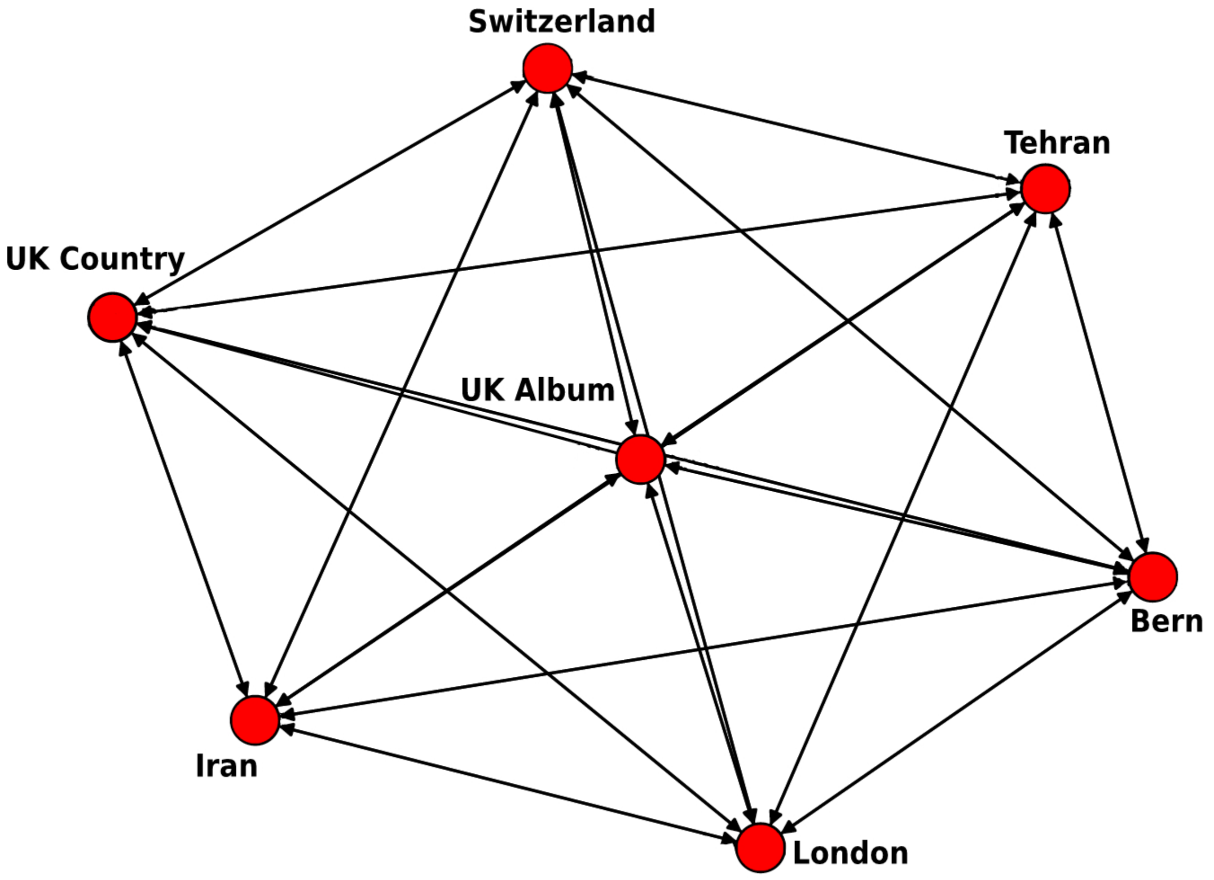


Fig. 3.1.: Looping graph with candidate names

Figure 3.1 shows the looping graph of the countries and their capitals. As mentioned, in each iteration of the Looping algorithm, we take the already annotated entities as well as one ambiguous entity. The nodes of the graph are the candidates of the annotated nodes, as well as all possible candidates of one ambiguous node. In this graph, except *United Kingdom*, all the nodes are already annotated. For *United Kingdom* entity, we have two candidates, a country (Q145) and a music album (Q7887906). To calculate the weights of the graph, we use Equation 3.1. We perform PageRank algorithm on this directed graph to generate the sorted list of candidates. The higher score from PageRank shows the stronger relationship of this node with other nodes of the graph. Figure 3.2 is a more realistic representation of this example. Here, we replace the entity names with their equivalent Wikidata identifiers. For each pair of nodes, we compute the weight of the edge by Equation 3.1. The node's score is the value assigned to it by PageRank. In the end, the candidate with the highest score is the annotation. In this example, the score for node Q145 is 0.11065, and the score of node Q7887906 is 0.10388. The node with type country has a better score compared to the node with type music album and has a stronger relationship. So, we can consider the country (Q145) as annotation of *United Kingdom*. We apply the same process for each ambiguous entity in a different iteration.

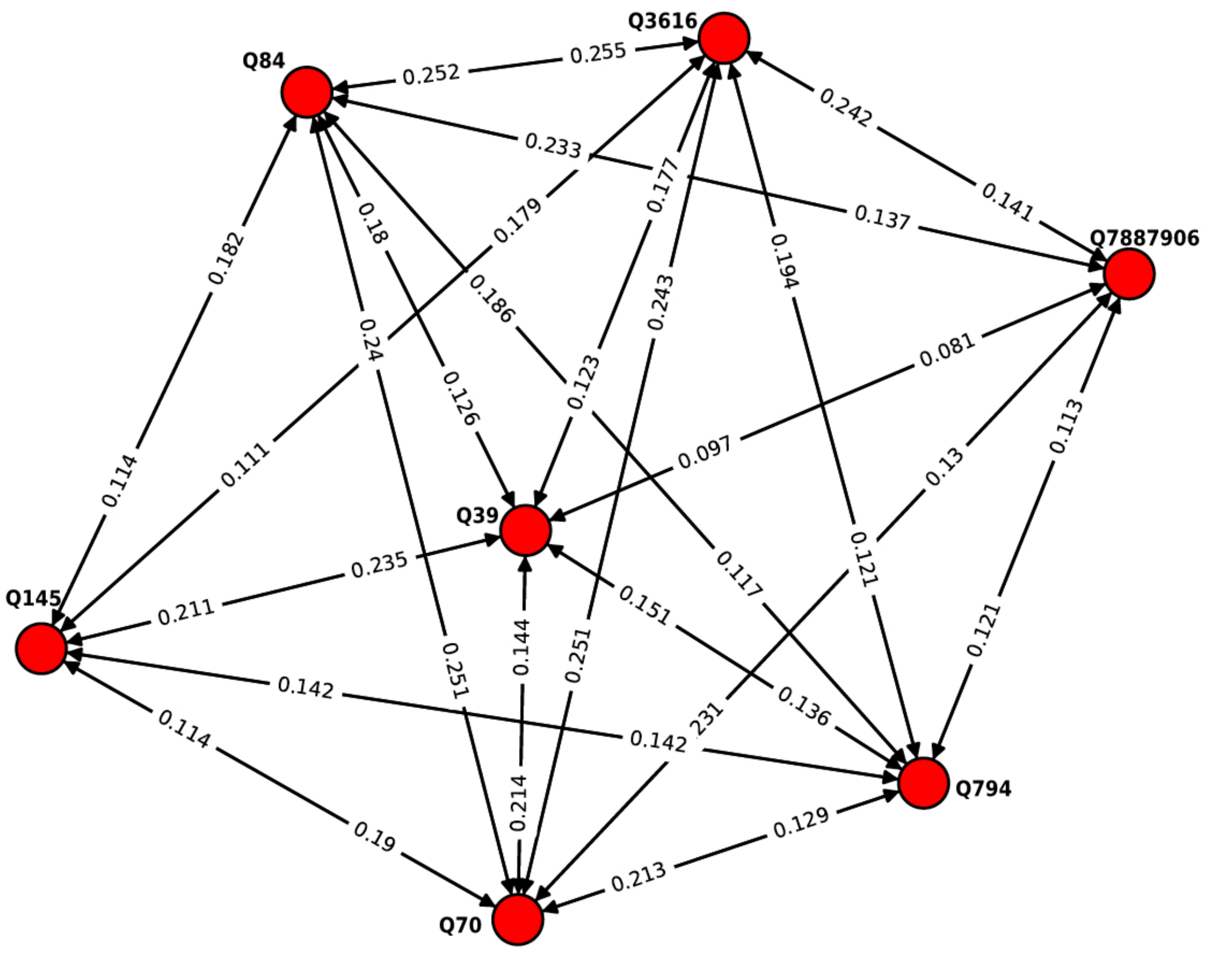


Fig. 3.2.: An example of weighted Looping graph

For the next round, *United Kingdom* (Q145) is already annotated and therefore, it exists in the base graph. We add the candidates of the other ambiguous entities to the looping graph. We continue and annotate one entity in each loop to cover all the entities of the table.

A problem that may occur while building the looping graphs is that the graphs can become huge. Creation of multiple graphs for each table, performing PageRank on each round, as well as annotating calculations add huge complexity to the algorithm. To solve this problem, we propose a solution to take only two columns at a time. We always take the label column, as it is the node for which we need to give the annotation, as well as one reference column at each round. By the end of the annotation, for each reference column, we have a separate dictionary of the results. When we decide about the final results, three situations may happen. Consider we have label column presenting by LC, first reference column as RF1 and second reference column as RF2. The three columns have the textual entities. The first situation is when the annotation from (LC, RF1) agrees with (LC, RF2) annotation on final result value. It can be either they have the same values, or they cannot provide any annotation. If both annotations are the same, we can be sure that our annotation is correct. The other situation, is when only one of the runs among (LC, RF1) and (LC, RF2) returns the annotation result, and the other one cannot annotate. We choose the only annotation result that we have. Finally, there exists a situation where both (LC, RF1) and (LC, RF2) are annotating the entity, but these annotations are not the same. In this case, we choose the annotation provided by the leftmost reference column. This method provides a much faster algorithm in comparison to taking all the columns at the same time.

3.2.3. Centrality Measures

So far, all semantic embedding methods used PageRank to sort the nodes of the graph by their importance. Apart from PageRank, there are other possible solutions to measure the influence of a node in a network. In this section, we apply the same methodology as explained in Looping with different centrality measures. Instead of using the PageRank, we try to see how EigenVector and Katz centrality can result in better precision, recall, and F1, and also how they affect the run time over T2D and Limaye gold standard tables.

3.3. Hybrid

We have currently two principal categories of methods. In each case, the approach can target a certain amount of the entities which are not necessarily the same. The purpose of the Hybrid methods is to use both lookup-based and semantic embedding based methods to annotate more entities and as a result, achieve better recall, precision, and F1.

3.3.1. Hybrid I

In the first iteration, Hybrid I chooses one of our lookup-based methods. If it annotates, whether correct or wrong, we keep the annotation. In any case, where the lookup-based method is not providing an answer, we use the entity annotation from the semantic embedding method. To start, we show the Hybrid I result over our baseline methods, to have a fair comparison with the improved version of the Hybrid I.

3.3.2. Hybrid II

The Hybrid II stands on the same concept as in Hybrid I, except, Hybrid II starts with a semantic embedding approach and enriches the results with a lookup-based method. We show the results of the Hybrid II with baseline methods and compare them to the proposed methods of this thesis.

3.3.3. Hybrid Refinement of Looping

We make some further refinements in our proposed looping method by type checking. As explained in section 3.3.2, in each loop, we create a graph of all candidates for the entities of the Web table. This graph is used as an input of the PageRank algorithm to generate a ranked list of nodes. While adding a new node to the looping graph, we may have the same ranking output for multiple candidates. To choose the most related one, we look for the top 3 types of nodes in our existing looping graph. Also, we find the top 3 types of candidates with the same rank. Our chosen result is the candidate with the most similar type to the looping graph type.

4 Evaluation and Results

In this chapter, we present the benchmark datasets and our experimental results on two of these datasets. In datasets section, we explain the details of each gold standard, the table files, and entity files. Also, we present a refined version of one of the gold standards as well as the evaluation of different methods over these gold standards. Finally, we provide a fair comparison of our baseline method, our proposed methods, and the insight behind the results.

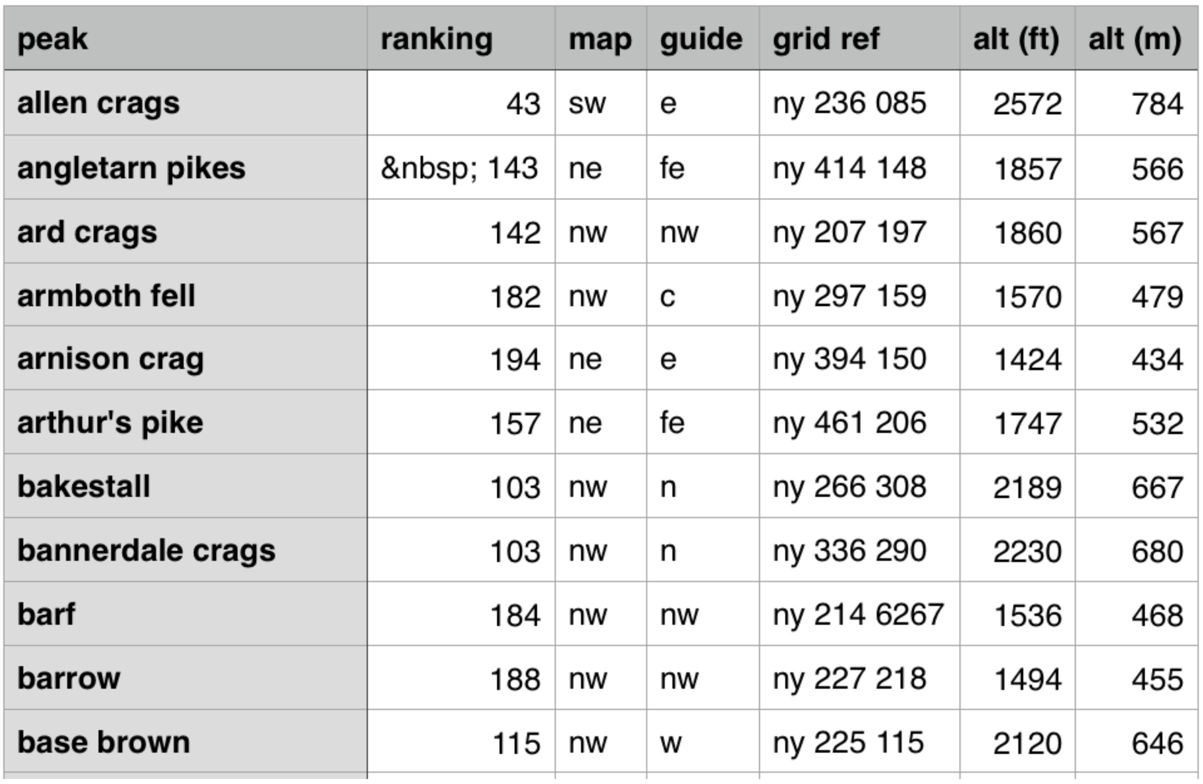
4.1. Datasets

Before we start to evaluate our methods on Web tables, we give an overview of the three datasets. For each of them, we have table files as well as table annotation files. The table annotations files, called entity files, are used as a gold standard to verify our results. We create table files and entity files in both CSV and JSON format.

4.1.1. The T2D Dataset

The gold standard presented in [21], contains HTML tables that cannot be matched with DBpedia, added by matchable Web tables. The authors keep a reasonable ratio for each group. The topics of the table collection are from sport, politics, history, geography, etc. T2D gold standard is divided into two categories of *entity-level* and *schema-level* annotations. Since we do not match the table columns to the Knowledge Base properties, we do not use schema-level annotations. We have the entity-level annotations for 233 tables in the T2D data set. As noted, we have only one annotation per row. There are 26124 rows, annotated by DBpedia unique pages. 2523 rows could not be matched to DBpedia. All the annotations are labeled manually by the authors of [5].

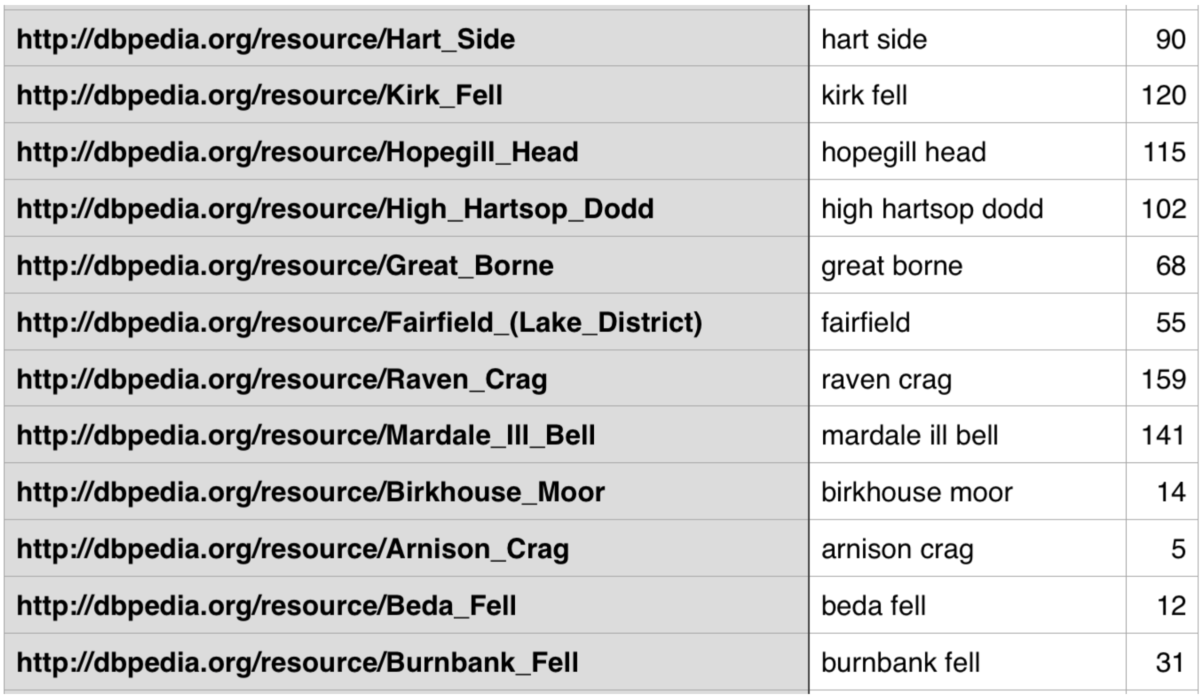
Table 4.1 demonstrates an example of a table file from T2D dataset in CSV format. As we can see, we have a combination of numeric and textual entities. In column *map*, the number of characters is not enough to decide about the entity. The annotation file gives us the mapping for the label column, which is *peak* in this case.



peak	ranking	map	guide	grid ref	alt (ft)	alt (m)
allen crags	43	sw	e	ny 236 085	2572	784
angletarn pikes	143	ne	fe	ny 414 148	1857	566
ard crags	142	nw	nw	ny 207 197	1860	567
armboth fell	182	nw	c	ny 297 159	1570	479
arnison crag	194	ne	e	ny 394 150	1424	434
arthur's pike	157	ne	fe	ny 461 206	1747	532
bakestall	103	nw	n	ny 266 308	2189	667
bannerdale crags	103	nw	n	ny 336 290	2230	680
barf	184	nw	nw	ny 214 6267	1536	468
barrow	188	nw	nw	ny 227 218	1494	455
base brown	115	nw	w	ny 225 115	2120	646

Tab. 4.1.: An example of T2D CSV table file

Table 4.2 presents an example of a CSV entity file of the T2D dataset. The first column shows a mapping of the entity to the corresponding DBpedia page. The second column is the annotated entity from the table row. Finally, the third column contains the index of the row.



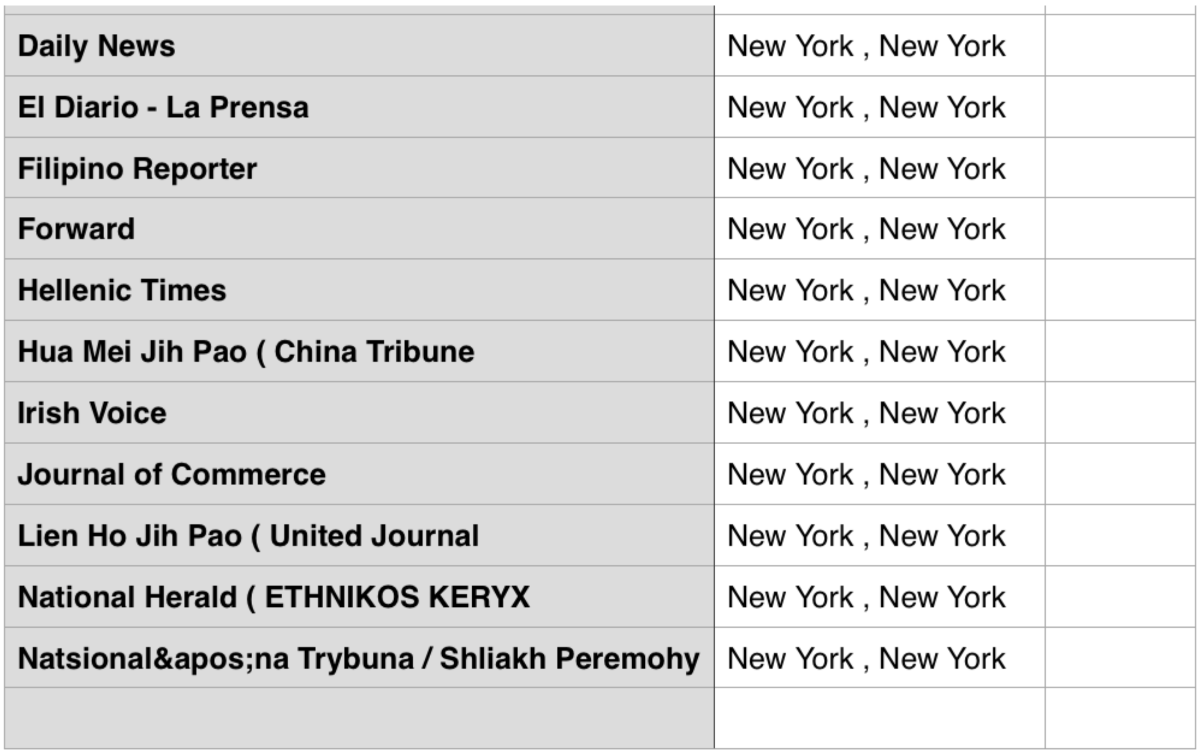
http://dbpedia.org/resource/Hart_Side	hart side	90
http://dbpedia.org/resource/Kirk_Fell	kirk fell	120
http://dbpedia.org/resource/Hopegill_Head	hopegill head	115
http://dbpedia.org/resource/High_Hartsop_Dodd	high hartsop dodd	102
http://dbpedia.org/resource/Great_Borne	great borne	68
http://dbpedia.org/resource/Fairfield_(Lake_District)	fairfield	55
http://dbpedia.org/resource/Raven_Crag	raven crag	159
http://dbpedia.org/resource/Mardale_Ill_Bell	mardale ill bell	141
http://dbpedia.org/resource/Birkhouse_Moor	birkhouse moor	14
http://dbpedia.org/resource/Arnison_Crag	arnison crag	5
http://dbpedia.org/resource/Beda_Fell	beda fell	12
http://dbpedia.org/resource/Burnbank_Fell	burnbank fell	31

Tab. 4.2.: An example of T2D CSV entity file

4.1.2. Limaye

The Limaye dataset [14] contains two groups of Web tables. The first group, which is called Wiki_Manual, contains 36 tables, collected from the texts in Wikipedia. The provided annotation is not only for the entities but also for the column relations and the type. This dataset is used to fetch the second group of the Web tables, mentioned as Web_Manual. Web_Manual presents 371 Web tables, which are noisier in comparison to Wiki_Manual.

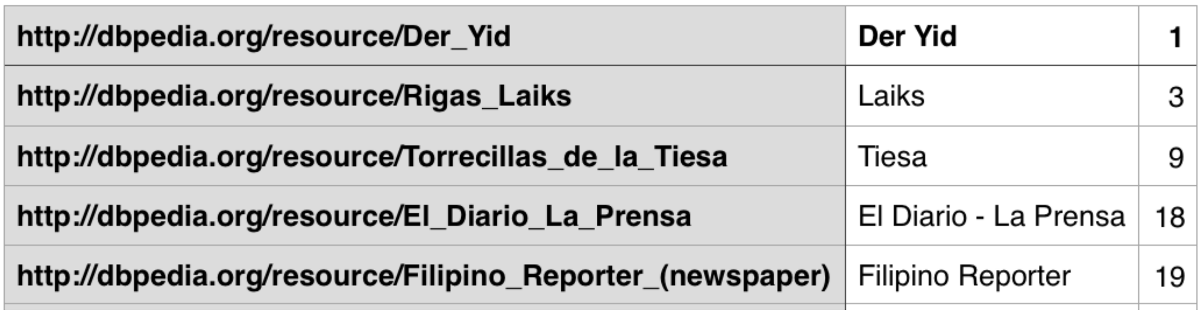
Wiki_Manual contains an average of 37 rows per table, with a total of 1691 entities. The other gold standard, Web_Manual has 35 rows per table on average, and in total, the number of entities in these 371 tables is 35. The manual annotation maps the entities to Wikipedia pages. Moreover, [2] published the refined version of this dataset. We call both Wiki_Manual and Web_Manual together as Limaye gold standard, and we use the corrected version that contains 296 Web tables in total. Out of 8,670 rows in this version of Limaye, we can map 5,278 rows to the Knowledge Base. This dataset has the least value of structuredness, and as we see in Table 4.3, there are several empty rows and columns in the Web table.



Daily News	New York , New York	
El Diario - La Prensa	New York , New York	
Filipino Reporter	New York , New York	
Forward	New York , New York	
Hellenic Times	New York , New York	
Hua Mei Jih Pao (China Tribune	New York , New York	
Irish Voice	New York , New York	
Journal of Commerce	New York , New York	
Lien Ho Jih Pao (United Journal	New York , New York	
National Herald (ETHNIKOS KERYX	New York , New York	
Natsional&apos;na Trybuna / Shliakh Peremohy	New York , New York	

Tab. 4.3.: An example of Limaye CSV table file

Table 4.3 presents a sample Web table from Limaye dataset. This table does not have any header information. Empty column and row, and wrong or noisy entity texts can bring problems for the annotation of such a table.



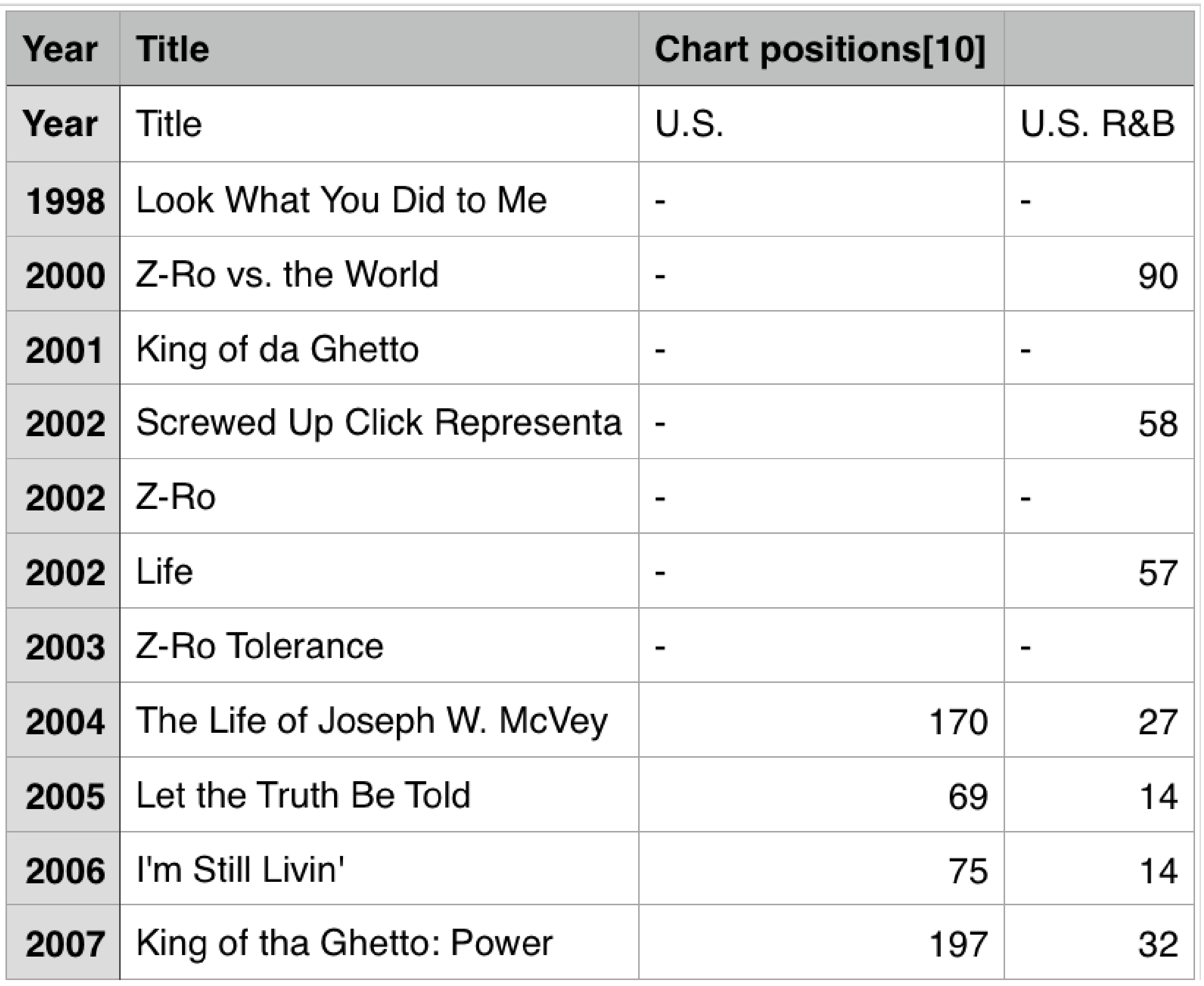
http://dbpedia.org/resource/Der_Yid	**Der Yid**	**1**
http://dbpedia.org/resource/Rigas_Laiks	Laiks	3
http://dbpedia.org/resource/Torrecillas_de_la_Tiesa	Tiesa	9
http://dbpedia.org/resource/El_Diario_La_Prensa	El Diario - La Prensa	18
http://dbpedia.org/resource/Filipino_Reporter_(newspaper)	Filipino Reporter	19

Tab. 4.4.: An example of Limaye entity file

Table 4.4 demonstrates the annotation of the Web table in Table 4.3. Although the Web table has 28 rows, the annotation is available only for 5 rows. Again, like T2D, the first column is the annotation link, the second column provides the annotated entity, and the third column holds the row index. In our methods, we use only one cell from each row and map it to the Knowledge Base. It means that we keep only the label column annotation. The entity annotation contains DBpedia links, which is not the same as the original version of the Limaye.

4.1.3. Wikipedia

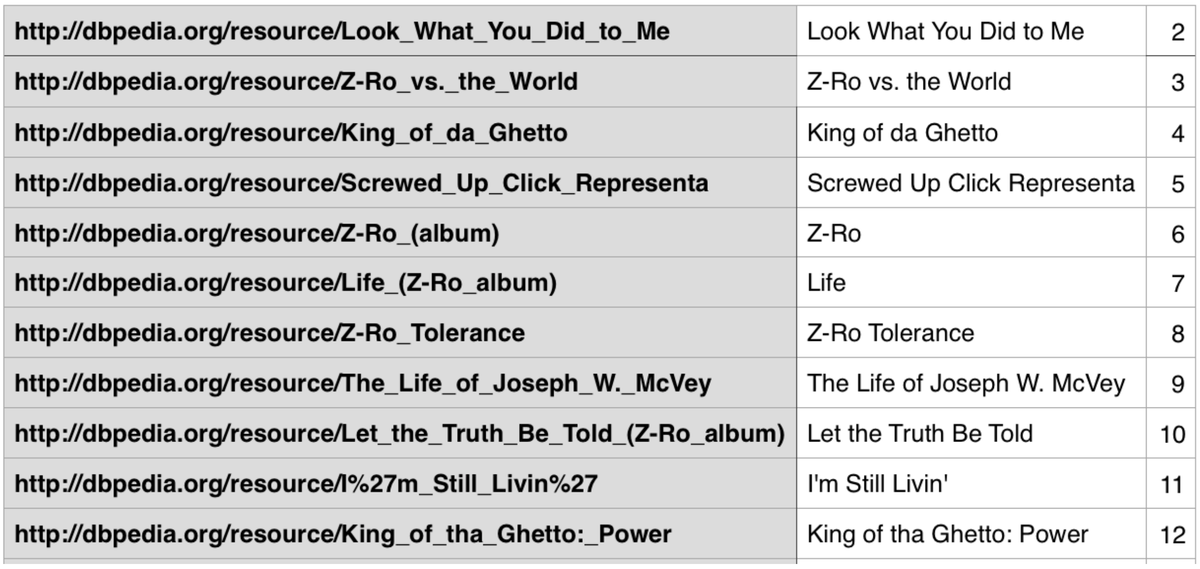
Our largest dataset with 485,096 HTML tables is Wikipedia. The Wikipedia gold standard is created by [5], which replaces the hyperlinks of Wikipedia with the corresponding links in DBpedia. In the Limaye gold standard, there is more than one annotation available per row. For sake of uniformity, we only keep the annotation of the label column. For this dataset, the authors provide only one mapping for each row. The Web tables are gathered from Wikipedia pages. The number of annotated rows, in this case, is 4,453,329 out of 7,437,606 total number of rows. This dataset is noisier than Limaye or T2D. Table 4.5 shows a Web table sample from Wikipedia dataset.



Year	Title	Chart positions[10]	
Year	Title	U.S.	U.S. R&B
1998	Look What You Did to Me	-	-
2000	Z-Ro vs. the World	-	90
2001	King of da Ghetto	-	-
2002	Screwed Up Click Representa	-	58
2002	Z-Ro	-	-
2002	Life	-	57
2003	Z-Ro Tolerance	-	-
2004	The Life of Joseph W. McVey	170	27
2005	Let the Truth Be Told	69	14
2006	I'm Still Livin'	75	14
2007	King of tha Ghetto: Power	197	32

Tab. 4.5.: An example of Wikipedia table file

Finally, Table 4.6 presents a sample of the annotation file from Wikipedia gold standard with DBpedia annotations.



http://dbpedia.org/resource/Look_What_You_Did_to_Me	Look What You Did to Me	2
http://dbpedia.org/resource/Z-Ro_vs._the_World	Z-Ro vs. the World	3
http://dbpedia.org/resource/King_of_da_Ghetto	King of da Ghetto	4
http://dbpedia.org/resource/Screwed_Up_Click_Representa	Screwed Up Click Representa	5
http://dbpedia.org/resource/Z-Ro_(album)	Z-Ro	6
http://dbpedia.org/resource/Life_(Z-Ro_album)	Life	7
http://dbpedia.org/resource/Z-Ro_Tolerance	Z-Ro Tolerance	8
http://dbpedia.org/resource/The_Life_of_Joseph_W._McVey	The Life of Joseph W. McVey	9
http://dbpedia.org/resource/Let_the_Truth_Be_Told_(Z-Ro_album)	Let the Truth Be Told	10
http://dbpedia.org/resource/I%27m_Still_Livin%27	I'm Still Livin'	11
http://dbpedia.org/resource/King_of_tha_Ghetto:_Power	King of tha Ghetto: Power	12

Tab. 4.6.: An example of Wikipedia entity file

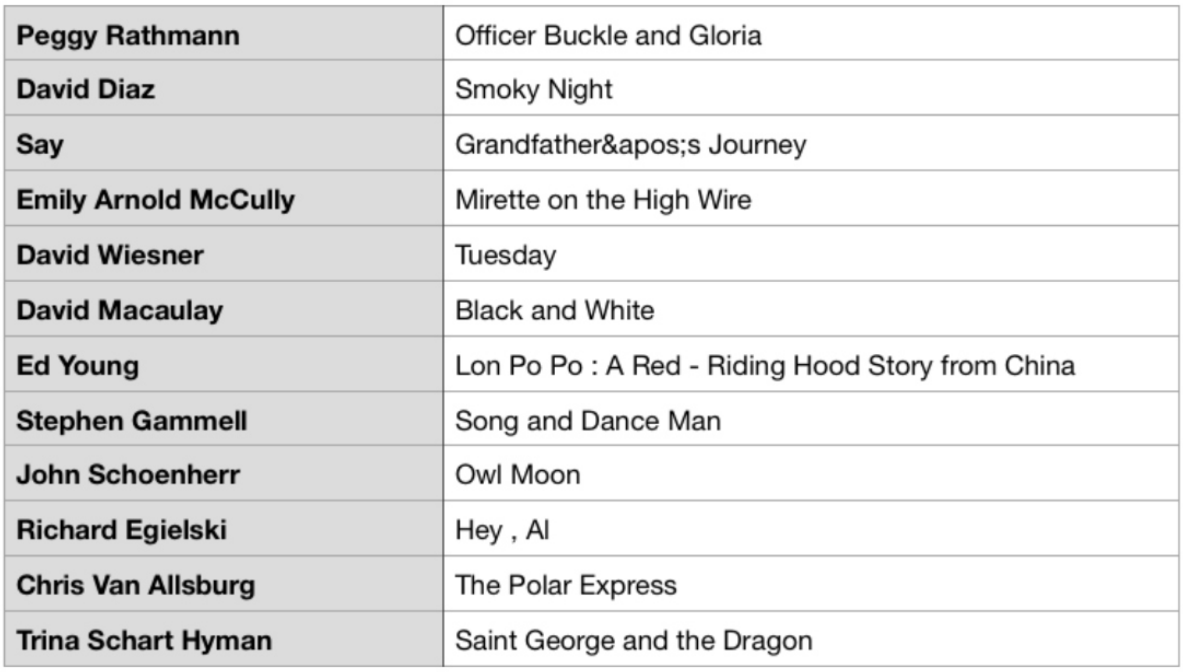
4.1.4. Refined version of Limaye

In this thesis, we initially used the refined version of Limaye gold standard by Bhagavatula et al. [2]. The annotation links of this gold standard were changed to DBpedia by Efthymiou et al. [5]. For a significant amount of tables, the final identifiers from our algorithm were correct, but the reported result of the implementation was different than the manual calculation of the metrics. We checked the annotations in the gold standard, and we found several missing values as well as multiple wrong annotations. As a result, several correct annotations were not counted as True Positive because we did not have their equivalent value in the gold standard. For incorrect annotations of the gold standard, our correct annotations were counted as False Positive. To solve this problem, we manually checked all the Label Column entities in the Web tables. We further checked the refined annotations by the Reference Column entities. For example, Table 4.7 shows the entities from a Limaye Web table. The gold standard does not provide any annotation for the entity *Black and White*. To annotate, we first look for it in `http://dbpedia.org/page/Black_and_white_(disambiguation)`. There exist 31 possible annotations in this disambiguation page. From unambiguous rows of the table and also the provided annotations in the gold standard for the other rows, we infer the type of the columns. In this case, the second column shows a list of books. Out of 31 values, there exist two related links as `http://dbpedia.org/page/Black_and_White_(book)` and `http://dbpedia.org/page/Black_and_White_(novel)`. We look for the Reference Column value *David Macaulay* and choose the link `http://dbpedia.org/page/Black_and_White_(book)` as the correct annotation. In this table, out of 68 rows in total, there were two wrong annotation and 21 rows without any annotation.

The whole process of annotation was performed manually. It took 20 working days to check the entire gold standard. Apart from using the DBPedia disambiguation pages, we used the DBPedia SPARQL endpoint [1] to query the entities. In the DBPedia version

[1]`https://dbpedia.org/sparql`

of the gold standard by Efthymiou et al., the wrong or missing links mostly appear when there exist punctuation symbols in the link. We provide a new version of this gold standard with correct annotations.



Peggy Rathmann	Officer Buckle and Gloria
David Diaz	Smoky Night
Say	Grandfather&apos;s Journey
Emily Arnold McCully	Mirette on the High Wire
David Wiesner	Tuesday
David Macaulay	Black and White
Ed Young	Lon Po Po : A Red - Riding Hood Story from China
Stephen Gammell	Song and Dance Man
John Schoenherr	Owl Moon
Richard Egielski	Hey , Al
Chris Van Allsburg	The Polar Express
Trina Schart Hyman	Saint George and the Dragon

Tab. 4.7.: A Sample Table from Limaye Gold Standard with Missing Annotations Links

4.2. Mappings

For all experiments, we use two gold standards with DBpedia and Wikipedia annotations. These datasets are publicly available from Efthymiou et al. [5][2] and Bhagavatula et al. [2][3]. Each dataset is provided in different formats such as XML, JSON, or CSV. We aim to keep the table and entity files for each dataset in both JSON and CSV format. So, we have generated the missing file formats.

In the annotation files, different Knowledge Bases are used. Our approaches link each row of a Web table to the unique pages of Wikidata. In this case, we have generated SPARQL scripts to generate mapping files. We crawl each dataset separately and store the existing annotation link with its equivalent link in Wikidata. By the end, we have the Wikidata links of annotation results, and we use the mapping files to map each result to its equivalent DBpedia link. As a result, we can compare our results to the gold standard and calculate precision and recall.

[2]http://www.cs.toronto.edu/õktie/webtables/
[3]http://websail-fe.cs.northwestern.edu/TabEL/

4.3. Experimental Set up

In each method, we require some additional information about the entities. As an example, in the case of lookup-based methods, we need to find the entity types and the description tokens. To extract the relations between different columns, we should know if a particular property exists in the current entity page or not. For all these tasks, we should execute a SPARQL query on Wikidata SPARQL endpoint. Since we need to send several queries, we cannot use the Wikidata API. To solve this problem, we use *wikidata-query-rdf* [4]. We download the whole dump of Wikidata and run our queries locally.

4.4. Results

In this section, we show the results of the annotation methods over Limaye and T2D datasets and present the Precision, Recall, and F1-Measure. The metrics calculation is computed in a micro-averaged manner. It means for each table, we calculate true positive, false positive, and false negative annotations. To calculate the metrics of the whole dataset, we sum up the results of all tables. We treat the whole dataset as one Web Table. In these experiments, a result is true positive, when we have the same link as provided in the gold standard. The False Positive occurs when our annotation does not match with the DBpedia link in the gold standard, and finally, the result is false negative if we could not provide any annotation.

4.4.1. Lookup-based Method Results

In this section, we present the lookup-based method results over Limaye and T2D datasets. Moreover, we do different tests to see the effect of different parameter tunings. We decide about the parameters through a trial and error manner.

As mentioned in Chapter 3, in our first step, we take the baseline algorithm of FactBase [5]. The results presented in their approach is based on annotation with DBpedia. We have implemented their algorithm using Wikidata.

	T2D			Limaye		
Method	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
FactBase	0.8784	**0.7814**	**0.827**	0.788	**0.834**	0.81
Majority-3	**0.897**	0.72	0.799	**0.82**	0.82	**0.82**
Majority-5	0.886	0.738	0.805	-	-	-

Tab. 4.8.: Results of lookup-based approaches over T2D and Limaye gold standards

In Table 4.8, FactBase has better results on T2D. FactBase uses all the columns to find the relations and annotates the rows in loose search based on these relations. Because of the better structuredness on T2D, we achieve better results on this dataset in comparison to Limaye. Moreover, in FactBase, we annotate a significant number of

[4]https://github.com/wikimedia/wikidata-query-rdf/blob/master/docs/getting-started.md

rows when they have only one candidate from the surface form. With high certainty, the annotated rows during the initialization phase are correct. So, they are essential to achieve high precision and also crucial for our next phases; because we choose our acceptable types based on these values. In Limaye dataset, we have more difficulties in finding such candidates. There are many entities that we cannot get their initial candidates, because the entity in the table is different from the entity title in the Knowledge Base page, and as a result, it is different than the key in the Surface Form. Although we handle dictation problems with Levenshtein distance, some keys are still not accessible. These situations happen more in Limaye dataset than T2D. It is also another reason that we achieve better result over T2D. Later in this chapter, we see the same hypothesis stands for the semantic embedding methods during the initialization phase.

We repeat the experiment of the lookup-based method with the *majority* method. Table 4.8 shows the result with two different parameters of Majority-3 or Majority-5. In each Majority-n method, we use the top n types of the table. In this proposed method, on both Limaye and T2D datasets, we obtain a higher precision value. Higher precision and lower recall prove that the majority-based method is very strict in choosing the annotations. With the same reason, the majority-3 is a more fine-grained method than majority-5 and as a result, has higher precision and lower recall.

4.4.2. Embedding-based Method Results

In this section, we report the results of semantic embedding method as well as our proposed Looping method. Then, we provide a comparison between different ranking methods on our graphs.

	T2D			Limaye			
Method	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Desc**
Baseline Embedding	0.62	**0.895**	0.734	0.75	**0.92**	0.83	M1, LCol
Baseline Embedding	0.733	0.747	0.74	-	-	-	M5, LCol
Baseline Embedding	0.62	0.70	0.66	0.76	0.82	0.79	M1, AllCol
Looping	**0.86**	0.82	**0.84**	**0.82**	0.87	**0.85**	PageRank, NoLev

Tab. 4.9.: Results of semantic embedding approaches over T2D and Limaye gold standards

We have trained our word embedding models with different parameter tuning. Although we do not have an automatic parameter tuning to build our Word2Vec model, we have tried Min_Count of 1 and 5. Over the T2D dataset, on M5, which shows the model with Min_Count of 5, we see around 15% lower Recall. It is logical because we keep the keywords that appear at least 5 times in our training corpus. It is not the case for many entities, and consequently, if a specific keyword does not exist in our model, we skip it, and we cannot give any annotation for that keyword. So, in the end, there are more entities in the Web table that do not have any annotations.

Considering the same model with the same parameters, we can use either label column entities to build our graph or all columns with non-numeric entries. If we use only label column entities in T2D dataset, we have around 7% drop of the F-measure. As we have explained in the label column detection section, and the pre-processing phase, we are not always able to drop useless columns if the entity contains a mixture of string and numerical values. In case of a Movie table, in an optimistic situation, the useless column may contain an entity like *season 7* which is at least a related entity. In the worst case, it can be a serial number of a movie, which is the combination of text and numbers. Such an entity does not provide any valuable information and reduces the coherency of the nodes in the graph. We may also end up in a worse result if the reference column contains a short description sentence. As we do not have any specific entity in our surface form for this sentence, we try to find it with the Levenshtein distance, and we may end up with an unrelated entity, and wrong results.

In Limaye dataset, we have tried only the models with Min_Count of one. For the same reason as in T2D dataset, we have 4% improvement of F-measure using only the label column entities.

In Table 4.9, we compare the looping method with the baseline method. Both results use PageRank as a ranking algorithm. For T2D dataset, this method has a significantly better precision of 24% and around 18% better F-measure. In the looping method, we create the initial graph with mostly correct entities and disambiguate other entities one by one. This more fine-grained method leads to lower recall. Results of the looping method over Limaye gold standard is 5% better in terms of F-measure and has 6% improvement of precision.

In Table 4.9, by *NoLev*, we mean we do not use the nodes found by Levenshtein in our initial graph even if they have only one possible candidate. More detailed result is shown in Table 4.10. We create the initial graph with the nodes solved by Levenshtein, as well as the initial graph without them. If we use Levenshtein for the first graph, the precision drops significantly. It means there exist at least some wrong results among the Levenshtein results.

Method	**Pr**	**Re**	**F1**
Looping with initial Levenshtein	0.67	**0.86**	0.7
Looping without initial Levenshtein	**0.84**	0.80	**0.82**

Tab. 4.10.: Looping with vs. without Levenshtein in the initial graph

So far, the results for both baseline embedding method and the looping method were based on PageRank. To find the best ranking method in terms of runtime and precision, we run our proposed method, looping, with three different centrality measurements. Table 4.11 presents the results of the looping with PageRank, EigenVector, and Katz centrality.

	T2D				Limaye			
Method	**Pr**	**Re**	**F1**	**RT**	**Pr**	**Re**	**F1**	**RT**
PageRank	0.8653	0.8166	0.8403	12.1h	0.8223	0.870	0.8455	**7.28h**
EigenVector	0.8658	0.8173	0.8403	**11.25h**	0.822	**0.8736**	**0.8468**	7.41h
Katz	**0.8662**	**0.8233**	**0.8423**	**117.2h**	**0.8227**	0.872	0.8466	7.79h

Tab. 4.11.: Results of Looping approach with different centrality functions over T2D and Limaye gold standards

In Table 4.11, although changing the PageRank to either Katz centrality or EigenVector results in a better F1-measure, our two datasets do not agree on the best choice. In terms of F1-measure, Katz centrality performs better on T2D dataset while over Limaye dataset, EigenVector is the winner. In terms of precision, both datasets agree that Katz provides the best result. Over Limaye dataset, changing the ranking algorithm does not significantly affect the runtime. Over T2D dataset, the fastest method is EigenVector while over Limaye, PageRank is faster. Katz has the longest runtime over both gold standards. On the other hand, there was a significant difference in runtime of Katz centrality in comparison to the two other methods over T2D gold standard. Unexpectedly, even with the long waiting time and multiple iterations, we failed to achieve convergence on a considerable number of tables.

As the final additional point, it worth mentioning that the combination of looping with type checking helped us to improve the in F-measure of looping method around 5% and 3% for T2D dataset and Limaye dataset respectively.

4.4.3. Hybrids

In this section, we report the results from the combination of the lookup-based methods with the semantic embedding methods. Since each method targets a different set of nodes, the combination of them should result in a better recall or a better precision.

Hybrid I

Hybrid I methods choose one of the lookup-based methods in the first stage. It can be either FactBase lookup or Majority-Based lookup method. We use different parameters and choose the methods with the best-reported results in the previous sections.

	T2D			Limaye			
Method	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Desc**
FactBase + Embedding	0.86	0.82	0.84	0.788	**0.87**	0.824	PageRank
Majority + Looping	**0.87**	0.82	0.843	**0.80**	0.85	0.824	Maj3, PageRank
FactBase + Looping	0.858	**0.838**	**0.848**	0.788	0.868	**0.827**	EigenVector
FactBase + Looping	0.857	**0.838**	0.847	0.785	0.867	0.824	Katz

Tab. 4.12.: Results of Hybrid I approaches on T2D and Limaye gold standards

In Table 4.12, the baseline Hybrid I has the lowest F1. For all the methods, the difference is not significant in terms of F-measure. The reason is the first method is the main phase for deciding the results. As FactBase is mostly the first method, and also because our Majority-3 method was not performing better than FactBase, the F1 values of all the Hybrid I methods are close on both datasets. Because Majority-3 method was more fine-grained, the best precision is achieved by the combination of majority base and looping.

Hybrid II

The reversed order of hybrid I is hybrid II method. For the first stage, we choose one of the semantic embedding methods, and we enrich the results by lookup-based methods during the second phase. We choose the methods with the best-reported results for each stage.

	T2D			Limaye			
Method	**Pr**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Desc**
Embedding + FactBase	0.67	0.78	0.72	0.77	0.86	0.82	PageRank
Looping + Majority	0.84	0.81	0.826	0.78	0.85	0.813	Maj3, PageRank
Looping + FactBase	**0.854**	0.837	**0.846**	**0.823**	0.87	**0.847**	EigenVector
Looping + FactBase	0.85	**0.84**	0.845	0.822	**0.872**	**0.847**	Katz

Tab. 4.13.: Results of Hybrid II approaches over T2D and Limaye gold standards

The results in Table 4.13 show that there is a remarkable improvement in F1 values in hybrid II compared to the baseline Hybrid II. As in hybrid I, still the best methods come from the combinations of looping and Fact base. Also, the results of F1 from Katz and EigenVector centrality measures are performing better than our best-achieved results. Table 4.13 shows that having the looping in the first phase and augmenting it with FactBase in the second phase can improve the whole algorithm. It is because FactBase has correct answers for the entities where Looping could not provide any results. The most striking point in Table 4.13 is that our looping method ranked by Katz combined with the FactBase is the best result we have achieved from all of the methods explained through out this thesis. The accord of two datasets confirms this conclusion.

4.4.4. Analysis

In this section, we want to study the factors that stopped us from a further improvement of the results. For all of our methods, one of the necessary steps is to extract the label column. There are two situations where we cannot detect the correct column. First, when it is not possible by our heuristic method to choose the right column. For example, there are two columns with the same number of distinct values, and typically, we choose the leftmost, while the correct label column is the second column. These situations are when it is even hard for a human to recognize the label column without any additional information. If there is a table caption or we have the headers, then a human can understand the main column while our algorithm cannot because we do not use any additional information of the table to get the label column. Second, when we have a mixture of numbers and texts in each cell, and we give value to the entities that are not valuable in reality. If the number of distinct values in these columns are more than the number of distinct values in our real label column, we choose the wrong column as Label Column. In this case, even if we annotate this entity correctly, it is not the correct annotation of the row. For this reason, we have the wrong annotation for all rows of the table.

The second problem during the annotation was missing the correct entity in the initial candidates. When we decide to annotate a table cell, we first get the unrefined set of candidates from our surface form. Apart from the incorrect results caused by accepting Levenshtein distance, some entities do not have any result, which we consider as FN. In the worst case, we search for an entity in surface index form, and we get a set of candidates, but the correct entity does not exist in this result set. This problem needs a careful refinement in our surface index form. There are also some Wikimedia Disambiguation Pages which have the same title as the entity that we are looking for, but they do not contain any useful information, and as a result, they are not our correct annotation.

As an example of the third possible scenario, consider a table with a list of movie names which are also novel names. The Wikidata IDs of these entities are so close to each other. For example entity Q2875 describes the film *Gone with the Wind* and entity Q2870 is the novel with the same name. In this case, correct type detection of a table becomes so tricky since, for most of the rows, both movie and novel exists among the candidates. Since we sort the candidates according to an ascending order of the Wikidata identifiers, we choose the smaller ID, and it becomes our acceptable type. In this case, the acceptable type is novel. One possible solution is to show the top n results to the user, so that, we are sure that the correct result exists among them and the incorrect results are still related.

Another common scenario is the different versions of one entity. For example, in the table with the name of the games, there can be different releases of the same game. As an example, the entity Q24589167 defines the video game called *God of War*. This entity is the release of 2018 with the specific name as *A New Beginning* which is not mentioned in the main title. The entity Q817369 is also with the title *God of War*, but this time without any additional name. The entities Q2252689, Q1129452, Q573779, Q2298167, and Q573768, correspond to all different versions of the *God of War* series. Finally, entity Q5576061 defines the whole collection of this game. In this example, our algorithm typically chooses the name of the whole collection as the annotation result. It is because first, it is hard to distinguish between these different versions as they have the same title and same types and also, the collection name has a more strong relation and bigger similarity value with the other games in the table. Again, the solution we propose is as same as the previous problem and to show the top n related results.

5 Conclusion and Future Work

In this thesis, we present the complementary methods for Web table annotation and disambiguation. We tackle the problem by two appraoches; first, the Lookup-based method which uses the types and relations in the Web table to choose the result, and second, the word embedding method that creates the weighted edges in the graph and uses the similarity of the entities to find the most related nodes using the context of the Web table. The semantic embedding method is similar to the text disambiguation techniques in which, we believe the words in a paragraph are mostly around the same context. Finally, we get the best results by the combinations of these two methods as the building blocks. We call these two-phase methods as Hybrid methods. Moreover, we show three different datasets and gold standards used to calculate the efficiency of our methods. Also, we provide a refined version of one of the gold standards annotated by DBpedia.

Based on the baseline lookup-based method, called FactBase [5], we propose the Majority-based method, which is a more fine-grained approach and provided better precision. Still, in terms of F-measure, FactBase outperforms our Majority-based method. For the semantic embedding methods, we propose a method called Looping by which we achieve a remarkably better F-measure in comparison to the state-of-the-art. This method uses iterative graph generation. We start with an unrefined set of candidates and build our initial graph with the unambiguous entities. For the rest of the entities, we annotate one entity in each iteration. Also, we compare several ranking methods in terms of precision of annotation results as well as runtime.

For future work, we plan to improve our surface index form, the way we achieve the initial candidates from the surface form and focus on the solution of getting the candidates even if the Web table text looks different than the key in the surface form. Also, we believe there is a high potential in embedding methods to achieve better annotation results. It is beneficial to refine the label column detection method. It is important to choose the correct label column otherwise the accuracy of the model could drop drastically.

A
Common Acronyms

KB	Knowledge Base
SF	Surace Form
SFI	Surface Form Index
GS	Gold Standard
CSV	Comma Separated Values
JSON	JavaScript Object Notation

B
License of the Documentation

References

[1] Halevy A.Y. Harb B. Lee H. Madhavan J. Rostamizadeh A. Shen W. Wilder K. Wu F. Yu C. Balakrishnan, S. Applying webtables in practice. CIDR, 2015. 8, 9, 10

[2] Noraset T. Downey D. Bhagavatula, C.S. Tabel: Entity linking in web tables. ISWC, 2015. 8, 25, 28, 29

[3] Evans C. Paritosh P. Sturge T. Taylor J. Bollacker, K. Freebase: a collaboratively created graph database for structuring human knowledge. SIGMOD, pp. 1247–1250, 2008. 3

[4] Roth D. Cheng, X. Relational inference for wikification. Empirical Methods in Natural Language Processing, 2013. 9

[5] Rodriguez-Muro M. Christophides V. Efthymiou V., Hassanzadeh O. Matching web tables with knowledge base entities: From entity lookups to entity embeddings. d'Amato C. et al. (eds) The Semantic Web – ISWC 2017. ISWC 2017. Lecture Notes in Computer Science, vol 10587. Springer, Cham, October 2017. 4, 6, 8, 9, 10, 11, 12, 13, 15, 17, 23, 26, 28, 29, 30, 36

[6] P. Ferragina F. Piccinno. From tagme to wat: a new entity annotator. Proceedings of the first international workshop on Entity recognition disambiguation, 2014. 13

[7] Lu M. Ooi-B.C. Tan W. Zhang M. Fan, J. A hybrid machine-crowdsourcing system for matching web tables. ICDE, 2014. 8

[8] Scaiella U. Ferragina, P. Fast and accurate annotation of short texts with wikipedia pages. IEEE Softw. 29(1), 70–75 (Jan 2012), 2011. 13

[9] Sun L. Han, X. An entity-topic model for entity linking. EMNLP-CoNLL. pp. 105–115. ACL, Stroudsburg, PA, USA, 2012. 9

[10] Sun L. Zhao-J. Han, X. Collective entity linking in web text: A graph-based method. SIGIR. pp. 765–774. ACM, New York, NY, USA, 2011. 9

[11] Yosef M. Bordino-I. Fürstenau H. Pinkal M. Spaniol M. Taneva B. Thater S. Weikum G. Hoffart, J. Robust disambiguation of named entities in text. Proceedings of EMNLP, 2011. 3

[12] Yosef M.A. Bordino-I. F¨urstenau H. Pinkal M. Spaniol M. Taneva B. Thater S. Weikum G. Hoffart, J. Robust disambiguation of named

entities in text. EMNLP. pp. 782–792. ACL, Stroudsburg, PA, USA (2011), 2013. 13

[13] D. Downey M. Anderson L. Ratinov, D. Roth. Local and global algorithms for disambiguation to wikipedia. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, 2011. 9, 13

[14] Sarawagi S. Chakrabarti S. Limaye, G. Annotating and searching web tables using entities, types and relationships. PVLDB, 2010. 8, 25

[15] I. Bordino M. Spaniol M. A. Yosef, J. Hoart and G. Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. Proc. VLDB Endow., 2011.

[16] Jakob M. Garc´ıa-Silva A. Bizer C. Mendes, P.N. Dbpedia spotlight: Shedding light on the web of documents. 7th I-Semantics. pp. 1–8. ACM, New York, NY, USA, 2011. 3, 13

[17] Chen K. Corrado-G. Dean J. Mikolov, T. Effcient estimation of word representations in vector space. CoRR abs/1301.3781, 2013. 17

[18] Witten I.H. Milne, D. Learning to link with wikipedia. 17th CIKM. pp. 509–518. ACM, New York, NY, USA, 2008. 9

[19] Ferragina P. Piccinno, F. Large-scale named entity disambiguation based on wikipedia data. EMNLP-CoNLL. pp. 708–716. ACL, Prague, Czech Republic, 2007. 9

[20] Ferragina P. Piccinno, F. From tagme to wat: A new entity annotator. First Int.Workshop on Entity Recognition/Disambiguation. pp. 55–62. ERD '14, ACM, NY, USA, 2014. 13

[21] Lehmberg O. Bizer-C. Ritze, D. Matching html tables to dbpedia. WIMS, 2015. 8, 9, 15, 23

[22] Paolo F.D. Barbosa-D. Merialdo P. Sekhavat, Y.A. Knowledge base augmentation using tabular data. LDOW, 2014. 10

[23] Wang J. Han-J. Shen, W. Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINNERING, 2015. 3

[24] Wang J. Luo-P. Wang M. Shen, W. Liege: link entities in web lists with knowledge. KDD, 2012. 8

[25] Wang J. Luo-P. Wang M. Shen, W. Linden: Linking named entities with knowledge base via semantic knowledge. 21st WWW. pp. 449–458. ACM, New York, NY, USA, 2012. 10

[26] Kasneci G.-Weikum G. Suchanek, F. M. Yago: A core of semantic knowledge unifying wordnet and wikipedia. WWW, pp. 697–706., 2007. 3

[27] Ngonga Ngomo A.C.-Auer S. Gerber D. Zhang M. Usbeck, R. Agdistis - graph-based disambiguation of named entities using linked data. The Semantic Web–ISWC, 2014. 9

[28] Halevy A.Y. Madhavan-J. Pasca M. Shen W. Wu F. Miao G. Wu C. Venetis, P. Recovering semantics of tables on the web. PVLDB, 2011. 8, 10, 15

[29] Z. Zhang. Towards effcient and effective semantic table interpretation. ISWC, 2014. 8, 9, 15

[30] Seifert C. Granitzer-M. Zwicklbauer, S. From general to specialized domain: Analyzing three crucial problems of biomedical entity disambiguation. 26th DEXA 2015, Valencia, Spain. pp. 76–93, 2015. 10

[31] Seifert C. Granitzer-M. Zwicklbauer, S. Doser - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. ESWC, 2016. 9, 12, 17

[32] Seifert C. Granitzer-M. Zwicklbauer, S. Robust and collective entity disambiguation through semantic embeddings. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 425–434. ACM., 2016. 12

Referenced Web Resources

[33] Wikidata. `http://www.wikidata.org` (accessed , 2019). 3

Index