People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Bordj Bou Arreridj Faculty of Mathematics and Computer Science

Graduation Final Project

in view of obtaining the diploma of master in computer science

Master in Computer Science

Title

Proposition and evaluation of an entity linking system based on machine learning methods

Realized by

TOUKALI Sabar & DJAAFRI Badre DDine

Graduate on ...

ACKNOWLEDGEMENTS

“

First of all, we would like to thank Almighty Allah, for His guidance and strength.

We would like to express our sincere gratitude and appreciation to our supervisor, Associate Doctor Belalta Ramla, for her continuous inspiration, support, guidance, encouragement, advice, patience, and individual feedback throughout the course of Graduate.Study. She was an approachable supervisor who motivated us to achieve our goals. We feel very grateful and blessed to have worked under her supervision. We would not imagine carrying out this research without her constant support and help.

We would like to express our gratitude and appreciation to our colleagues in our research group for their discussions, comments and advice. We would also like to thank all of our friends for their encouragement and support. We wish to express our appreciation to our honorable parents. And we would like to dedicate this dissertation to them.

”

Abstract

Named Entity Linking is the task of linking an ambiguous entity mention to a corresponding entry in a knowledge base. Current methods have mostly focused on large unstructured text data to learn representations of entities. Harvesting entity from these large text collections and linking it, is a major challenge.

Solutions to this entity linking problem, we propose an efficient linking method that uses machine learning technics and algorithms. We considered it as a binary classification problem, started with some principal features as mention and its candidate and other feature as mention context and candidate context to train the model and testing it. We use there ML method, one of them was Random forest which complete the task with a positive and satisfactory result.

Keywords : Entity linking, Machine learning, knowledge base, Binary classification, Natural language processing

Résumé

La liaison d'entité nommée est la tâche de lier une mention d'entité ambiguë à une entrée correspondante dans une base de connaissances. Les méthodes actuelles se sont principalement concentrées sur de grandes données textuelles non structurées pour apprendre des représentations d'entités. Récolter des entités à partir de ces grandes collections de textes et les relier est un défi majeur.

Solutions à ce problème de liaison d'entités, nous proposons une méthode de liaison efficace qui utilise des techniques et des algorithmes d'apprentissage automatique. Nous l'avons considéré comme un problème de classification binaire, commencé avec certaines caractéristiques principales comme mention et son candidat et d'autres caractéristiques comme contexte de mention et contexte de candidat pour entraîner le modèle et le tester. Nous y utilisons des méthodes d'apprentissage automatique, l'une d'entre elles était la forêt aléatoire qui termine la tâche avec un résultat positif et satisfaisant.

Mots clés : Entité Nommée, Apprentissage automatique, Base de connaissances, Classification binaire, Traitement du langage naturel..

ملخص

ربط الكيانات المسماة هي مهمة ربط كيان ما ذكر في نص مع الكيان الذي يوافقه في قاعدة المعرفة، الطرق الحالية ركزت على النصوص الكبيرة غير المنظمة لإستكشاف نمط لتمثيل هذه الكيانات. إستخراج هذه الكيانات من هذه النصوص الكبيرة وربطها تعد ت حديا كبيرا.

كحل لهذه المشكلة، إقترحنا نهج ربط فعال يستعمل تقنيات وخوارزميات الذكاء الصناعي. فإعتبرنا المشكل وكأنه مشكل تصنيف ثنائي، بدأنا بتدريب النموذج بإستخدام الخصائص الأساسية كالكيان المذكور في النص ومثيله في قاعدة المعرفة، وقمنا بإختباره بعد ذلك. استخدمنا ثلاث خوارزميات للذكاء الصناعي من بينها الغابة العشوائية والتي أكملت المهمة بنتيجة إيجابية ومرضية.

الكلمات المفتاحية: ربط الكيانات، الذكاء الصناعي، قاعدة المعرفة، التصنيف الثنائي، معالجة اللغة الطبيعية.

Contents

List of Figures

List of Tables

Listings

List of abbreviations and acronyms

NLP	*Natural Language Processing*
NER	*Named Entity Recognition*
NEL	*Named Entity Linking*
NED	*Named Entity Disambiguation*
LTR	*Learning To Rank*
VSM	*Vector Space Model*
IR	*Information retrieval*
SVM	*support-vector machine*
UKP	*Knowledge Processing Lab*
POS	*Part Of Speech*
CBOW	*Continuous Bag Of Words*
FEL	*Fast Entity Link*
HITS	*Hypertext-Induced Topic Search*
CRF	*Conditional Random Fields*

General Introduction

ML algorithms use statistical models that provide systems with the ability to produce predictions without human intervention, having accessed sample data, known as training data, in order to learn and improve human experiences. Available ML algorithms enable the analysis of large volumes of data, including: supervised ML algorithms, unsupervised ML algorithms, semi-supervised ML algorithms.

NLP began in the 1950s as the intersection of artificial intelligence and linguistics. NLP was originally distinct from text information retrieval (IR), which employs highly scalable statistics-based techniques to index and search large volumes of text efficiently. However, NLP and IR have converged somewhat. Currently, NLP borrows from several, very diverse fields, requiring today's NLP researchers and developers to broaden their mental knowledge base significantly.

NLP works behind the scenes to enhance tools we use every day, like chatbots, spell-checkers, or language translators. Combined with machine learning algorithms, NLP creates systems that learn to perform tasks on their own and get better through experience. NLP-powered tools can help you classify social media posts by sentiment, or extract named entities from business emails, among many other things and linking them to their corresponding entity in a knowledge base. This task is called Named Entity Linking.

The correct identification of the link between an entity mention in a text and a known entity in a large knowledge base is important in information retrieval or information extraction. The general approach for this task is to generate, for a given mention, a set of candidate entities from the base and, in a second step, determine which is the best one. In recent years there has been an increase in the amount of research seeking to apply ML to the named entity linking task. Since there are two primary issues with resolving named entity mentions: "variety" and "ambiguity", we have tried and implement three ML methods and we create a disambiguation dictionary and dataset contains many information could help to solve this problems.

Chapter 1

Introduction to Entity Linking

1.1 Introduction

Nowadays, the media have expanded significantly and rapidly, and with the development of the digital age, information has become widely spread, resulting in an exponential increase in the volume of data available online via blogs, newspapers, discussion forums, and other forms of information of all kinds. At this time, this data can only be interpreted by humans. The nature and intelligence of human could makes him understand the meaning of texts by understanding the context of the sentences, while while for the computer it's a hard task. for that a huge work and a lot of research is in progress to deal with this issue.

1.2 Natural Language Processing

Natural language processing (NLP) is a branch of artificial intelligence and linguistics aims to help computers understand human language phrases and words.[11] It is a discipline that focuses on the interaction between data science and human language, and is expanding to cover a lot of industries. NLP thrives today, thanks to massive improvements in data access and increased computational power, which allow practitioners to achieve meaningful results in areas such as healthcare, media, finance, and human resources.

For example in the translation domain has emerged to facilitate the work of users because some users cannot fully understand all languages and may not be well versed in a particular language, and NLP caters to those users who do not have enough time to learn or master new languages.[11]

1.2.1 NLP applications

There are many different fields that apply natural language processing and from this point we mention some of them such as Machine Translation, Email Spam detection, Information Extraction, Question Answering etc.

1.2.1.1 Machine Translation

Given that the majority of the world is online, making data accessible and available to everybody is a difficult undertaking.[11] The main challenge is to provide a lot of data, which in turn was able to break the language barrier; because many different languages have a completely different sentence, structure and grammar phrases than another language.

”Machine Translation is generally translating phrases from one language to another with the help of a statistical engine like Google Translate. The challenge with machine translation technologies is not directly translating words but keeping the meaning of sentences intact along with grammar and tenses. The statistical machine learning gathers as many data as they can find that seems to be parallel between two languages and they crunch their data to find the likelihood that something in Language A corresponds to something in Language B. As for Google, in September 2016, announced a new machine translation system based on artificial neural networks and Deep learning."[11]

In recent years, various methods have been proposed to automatically evaluate machine translation quality by comparing hypothesis translations with reference translations.

1.2.1.2 Text Categorization

"Categorization systems inputs a large flow of data like official documents, military casualty reports, market data, news-wires etc. and assign them to predefined categories or indices. Some companies have been using categorization systems to categorize trouble tickets or complaint requests and routing to the appropriate desks."[11]

A false negative and a false positive are both possible outcomes. Spam filters are at the heart of NLP technology, which is reduced to the task of extracting meaning from text strings. A spam filtering solution for an email system employs a collection of protocols to detect which incoming messages are spam and which are not.[11]

1.2.1.3 Question-Answering

"In contrast to Information Retrieval, which provides a list of potentially relevant documents in response to a user's query, question-answering provides the user with either just the text of the answer itself or answer-providing passages that may contain the right answer."[5]

1.2.1.4 Information Extraction

A relatively recent application field focuses on the identification, labeling, and extraction of specific essential aspects of information into a structured representation.[5]

For many applications, extracting entities such as names, places, events, dates, times and prices is a powerful way to summarize the information relevant to a user's needs, from large collections of text.

1.2.1.5 Dialogue System

Perhaps the most desirable applications of the future, are the systems envisioned by large providers of end user applications, Dialogue systems, which focuses on a narrowly defined applications (like refrigerator or home theater systems) currently uses the phonetic and lexical levels of language. It is believed that these dialogue systems when utilizing all levels of language processing offer potential for fully automated dialog systems.[5]

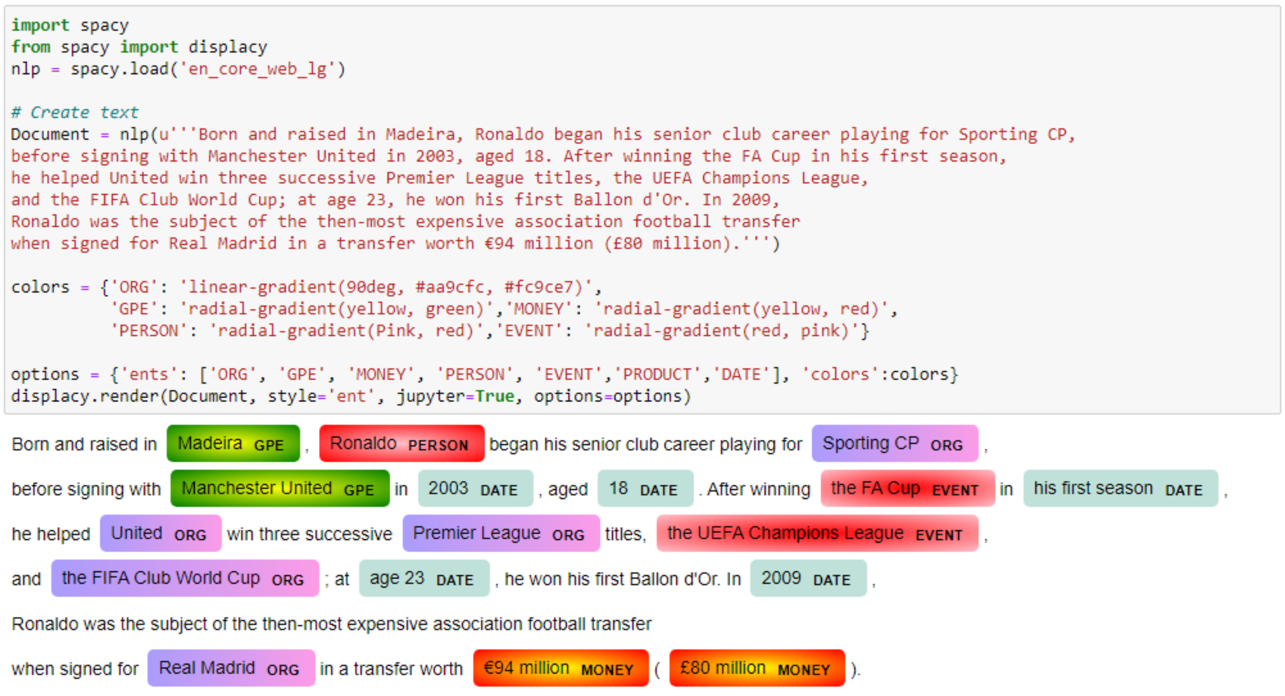
1.2.2 Named Entity recognition

Named entity recognition is the identification of important names in the text, and systems are required to recognize the named entities present in the text. In any text document, there are certain terms that represent specific entities that are more useful and have a unique context. These entities are known as named entities.

From this point of view, Named Entity Recognition (NER) must be used to identify the words of entities and what their types are, such as Person (PER), Organization (ORG), and Geo-Political Entities (GPE) etc. Determining the names of the entities is essential, which is why the information must be retrieved from the knowledge base, where all the information is usually available."For example, in the phrase 'Michael Jordan lives in the United States,' the NER system extracts Michael Jordan referring to the person's name and the United States indicating the name of the country. NER serves as the basis for many critical areas of information management, such as semantic annotations, and answering questions."[19]

From here we can understand the content of the text by extracting those names. Named Entity Recognition (NER) does not need to know the meaning of the word or how it affects the text, nor does it need to demystify these entities using a knowledge base.

Python Developers, and In particular Spacy Developers provide us an easy tool to extract named entities from a text which is more an efficient tool, For example we will try this library on one of the scripts in sports as shown in the figure 1.1:



```
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_lg')

# Create text
Document = nlp(u'''Born and raised in Madeira, Ronaldo began his senior club career playing for Sporting CP,
before signing with Manchester United in 2003, aged 18. After winning the FA Cup in his first season,
he helped United win three successive Premier League titles, the UEFA Champions League,
and the FIFA Club World Cup; at age 23, he won his first Ballon d'Or. In 2009,
Ronaldo was the subject of the then-most expensive association football transfer
when signed for Real Madrid in a transfer worth €94 million (£80 million).''')

colors = {'ORG': 'linear-gradient(90deg, #aa9cfc, #fc9ce7)',
          'GPE': 'radial-gradient(yellow, green)','MONEY': 'radial-gradient(yellow, red)',
          'PERSON': 'radial-gradient(Pink, red)','EVENT': 'radial-gradient(red, pink)'}

options = {'ents': ['ORG', 'GPE', 'MONEY', 'PERSON', 'EVENT','PRODUCT','DATE'], 'colors':colors}
displacy.render(Document, style='ent', jupyter=True, options=options)
```

```
Born and raised in Madeira GPE , Ronaldo PERSON began his senior club career playing for Sporting CP ORG ,
before signing with Manchester United GPE in 2003 DATE , aged 18 DATE . After winning the FA Cup EVENT in his first season DATE ,
he helped United ORG win three successive Premier League ORG titles, the UEFA Champions League EVENT ,
and the FIFA Club World Cup ORG ; at age 23 DATE , he won his first Ballon d'Or. In 2009 DATE ,
Ronaldo was the subject of the then-most expensive association football transfer
when signed for Real Madrid ORG in a transfer worth €94 million MONEY ( £80 million MONEY ).
```

Figure 1.1: example for named entity recognition

The named principal entities are identified by means of this table. To understand in more detail what each named entity means as shown in the table 1.1 :

TYPE	**DESCRIPTION**
PERSON	People, incuding fictional
NORP	Nationalities or religious or political groups
FACILITY	Buildings, airports, highways, bridges, etc
ORG	Companies, cities, states
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, ect (Not services)
EVENT	Named hurricanes, battles, wars, sports events, etc
WORK_OF_ART	Titles of books, songs, etc
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods
TIME	Times smaller than a day
PERCET	Percentage, including "%"
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	"first", "second", etc
CARDINAL	Numerals that do not fall under another type

Table 1.1: the named principal entities are identified by space

1.3 Named Entity Linking

1.3.1 What is Named Entity Linking

Named entity linking is the task of extracting query references in a text to its corresponding entity in the knowledge base. To connect the entity, three important steps must be followed: candidate entity generation, Candidate Entity Ranking and Evaluating the result. The first step is to identify the named entities and then the second step, which has a high importance, where we linking entities to the knowledge base. At last, the system returns the ID of the corresponding entity in the knowledge base or NIL (a label that indicate that there is no matching entity in the knowledge).[13]

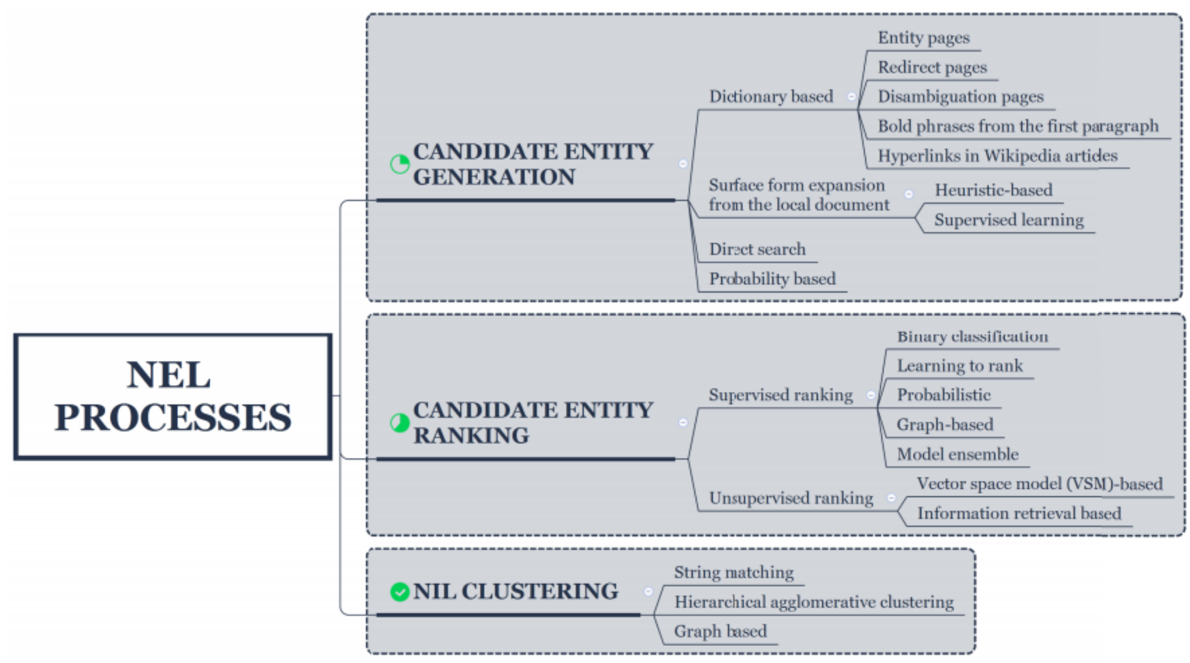


Figure 1.2: the three NEL sub-tasks along with their main techniques

1.3.2 Why Named Entity Linking

Is the task of disambiguating (NED) named entities and linking them to a knowledge base (Wikidata, DBpedia, or YAGO...). ”Entity linking also known as named entity disambiguation. The task of named entity linking refers to map named entity mentions in the text to their corresponding entities in a knowledge base.”[13]

”After overcoming the problem of recognizing an entity in a text, the attentions of researches are changed to entity disambiguation. As early as the 1990s, reference resolution and word sense disambiguation are the important tasks of Natural Language Processing. They deal with disambiguation of words in text so as to achieve the purpose of accurately understanding the meaning of the text.”[13]

1.3.3 Candidate Entity Generation

Given a set of mentions ‘M’ they appear in a text to linking them. So the aim of this step or process is to generate a candidate entity set ‘E_m’ for each entity mention ‘m’ in ‘M’.

There are several methods and techniques to deal with, in order to finish this stage with success. The success is not getting a large number of candidates, but to retrieve the most relative words (or sentences) to the mention ‘m’ in its context. We may get one thousand candidate entities but no one of them is the correct mapping entity, in the other hand we may get only ten candidate entities and pull off the result that we aiming to get, and retrieve the correct entity. So what matters is quality, not quantity.

In this part we will explain and discuss two approaches that was used largely.

1.3.3.1 Name Dictionary Based Techniques

1.3.3.1.1 What is a name dictionary

A name dictionary is a normal dictionary has two column one for an entity and the other one for its possible mapping entities which are a group include alternatives, synonyms, abbreviations, nicknames...etc of that entity. And might refer to it in some context.

We adopt this terminology:

- K_i is a line in the dictionary | were 1 < i < Dictionary length.
- K_i.name is the value of first column (name column) and K_i line.
- K_i.value is the value of second column (possible mapping entities) and K_i line.

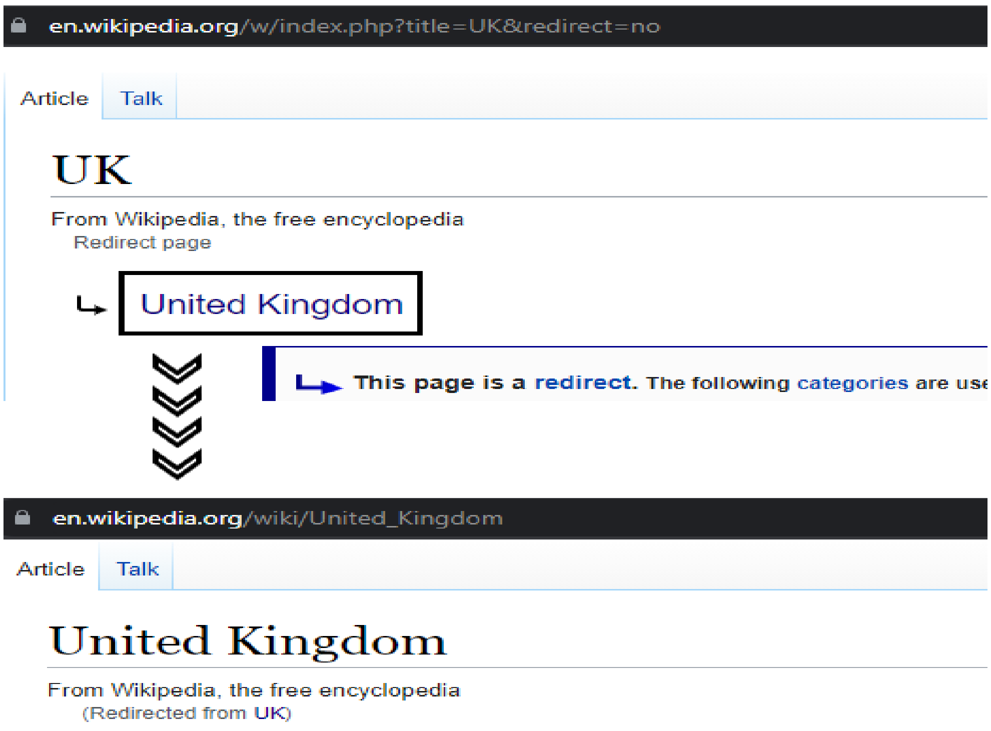
1.3.3.1.2 Construct the name dictionary

The dictionary is constructed by exploiting specifically five features from Wikipedia which are: Entity pages, Redirect pages, Disambiguation pages, bold phrases from the first paragraphs, Hyperlinks in Wikipedia articles.[18]

Entity pages: each Wikipedia entity page has a title and for sure it describe a specific entity for that choosing the title as a key and the entity was described as a value.

Redirect pages: Redirect page is a Wikipedia page that automatically avert visitors to another page. This page Refers to another Wikipedia entity which may was an alternative, synonyms or abbreviation to it. We mark the title of the redirect page as a key and the referred entity as a value.

The figure 1.3 shown below is an example of a redirect page. We note that the page's title is UK (considered as K_i.named) and the page that it redirect to is United Kingdom (considered as K_i.value). And together construct a line K_i.


en.wikipedia.org/w/index.php?title=UK&redirect=no

Article Talk

UK

From Wikipedia, the free encyclopedia
Redirect page

United Kingdom

This page is a redirect. The following categories are use

en.wikipedia.org/wiki/United_Kingdom

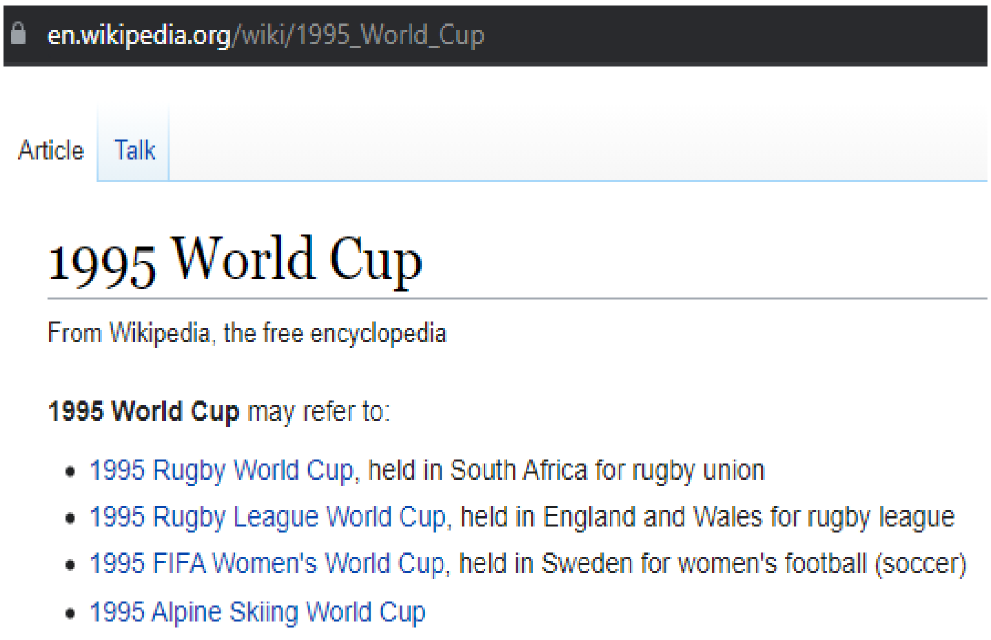
Article Talk

United Kingdom

From Wikipedia, the free encyclopedia
(Redirected from UK)

Figure 1.3: example of redirect page

Disambiguation pages: In Wikipedia site, they created page just to make difference between entities that may are very similar to each other and make one confused and not understand to who the entity referred. This pages contains a list of them. An example is shown in this figure 1.4:


en.wikipedia.org/wiki/1995_World_Cup

Article Talk

1995 World Cup

From Wikipedia, the free encyclopedia

1995 World Cup may refer to:

- 1995 Rugby World Cup, held in South Africa for rugby union
- 1995 Rugby League World Cup, held in England and Wales for rugby league
- 1995 FIFA Women's World Cup, held in Sweden for women's football (soccer)
- 1995 Alpine Skiing World Cup

Figure 1.4: example of disambiguation pages

This entity "1995 World Cup" may refer to many entity such as 1995 Rugby World Cup or 1995 FIFA Women's World Cup. For that this page was created and we set the page's title as a key and the referred entities as a value.

Bold phrases from the first paragraphs: A bold phrase in any paragraphs has an important value and gives you a good glimpse (overview) of what is the article talking about. From Wikipedia pages we take the bold phrases from the first paragraphs as value and the page title as a key.

Hyperlinks in Wikipedia articles: A hyperlink has a similar meaning to redirect page so it refer to another entity. Take the anchor text and mark it as a key and the referred page's title as value.

1.3.3.1.3 Take advantage of the dictionary

You should wondering how we benefit from this dictionary, so here is it, Let 'M' be a set of 'm' entity mentions in a document that we dig into linking it with a Wikipedia entity. And let 'E_m' be a set of candidate entity for the entity mention 'm' $\in$ 'M'.

To generate candidate entity set for an entity mention depending on a name dictionary we do a search in the key column (first column) and add the k_i.vaule to 'E_m' whenever one of these conditions is met:

- Exact matching between the entity mention 'm' and 'k_i.name'
- m $\in$ k_i.name
- k_i.name $\in$ m

With this approach we generate candidates which are probably the synonyms, alternatives of mention entity as we explained above.

1.3.3.2 Methods Based on Search Engines

There are a lot of methods and technics that are used in order to generate appropriate entities to an entity mention in a document. One type from them is search engines methods.

Web search engines considered as the infrastructure of search engines methods because are totally based on them, such as Google, Bing ...etc. There are those who use the Google search engine and make a query using the entity mention and some words that represent its context. And retrieve just the Wikipedia Web pages from the query result to regard them as candidate entities.[18] others like Dredze think differently a little bit and decide to use google search engine too, but they limit the number of Wikipedia pages by adding a new condition to be achieved before considering that page as a candidate entity which is as following: the page must be among and appear in the first twenty results of that search procedure.[6]

1.3.4 Candidate Entity Ranking

After an entity linking system complete the previous process which is Candidate Entity Generation and retrieve a set of candidate entities, all that remains is choosing the most likely entity to be the correct mapping entity for the entity mention. We call this process the Candidate Entity Ranking, where we do the important work, so that we do processing on these candidate entities in order to ranking them depending on several methods and techniques we will talk about them lately, then select the one that got the first place to become the mapping entity which maybe correct or not.

Some systems does not select the first entity in that ranking unless it exceeded a certain rate's value.

1.3.4.1 Features

We start this part with the definition of these two terms the surface and the context of an entity mention.

"The surface srfc (m) of a mention 'm' is the actual word or phrase used to identify the referenced real-world instance in the text."[8]

"The context ctxn (m) of mention 'm' is a multi-set of 'n' consecutive terms surrounding the mention (including its surface)."[8]

There are a lot of features are very helpful in candidate entity ranking with the aim of selecting the correct mapping entity. We'll talking about them briefly. But before that, we want to mention to that we divide the set of features on two categories which are context-independent features and context-dependent features. The difference between the two is that the context-independent category is only care about the surface of mention and ignore any other things. In contrast to this lastly, the context-dependent features take into consideration the mention's context where it appears.

1.3.4.1.1 Context-Independent Features

- **Name String Comparison**

As logic the first thing one shall take into consideration is string comparison for that a lot of measures are used such as Dice coefficient score, character Dice, skip bigram Dice. It is used to calculate how much the entity mention and a candidate entity from the set of candidate entities Em are similar to each other.

Dice coefficient score: it is a statistical method used to compare similarity between two strings the coefficient may be calculated for two strings, x and y using bigrams as follows:

$$d = \frac{2 * n_t}{(n_x + n_y)} \tag{1.1}$$

Where nt is the number of character bigrams found in both strings, nx is the number of bigrams in string x and ny is the number of bigrams in string y.

Skip bigram Dice: A bigram or digram (using Greek numerical prefixes) is a sequence of two adjacent elements from a string of tokens, which are typically letters, syllables, or words. A bigram is an n-gram for n=2. Hence the function is as follows: [1]

sim () is the similarity function.

$$sim(s_1, s_2) = \frac{1}{N-n+1}\sum_{i=0}^{N-n+1} h(i) = \frac{1}{N-2+1}\sum_{i=0}^{N-2+1} h(i) = \frac{1}{N-1}\sum_{i=0}^{N-1} h(i) \qquad (1.2)$$

Where :

h(i) = 1 if n-element subsequence beginning from position i in s_1 appears in s_2.

h(i) = 0 otherwise;

N-n+1 = number of n-element subsequence in s_1.

But sometimes this feature failed to get the correct mapping entity or bring us closer to it. For a simple reason related to the language itself. Some words differ in one letter that change the meaning radically. For example the word night and the word fight are very similar in writing but they have a huge different in meaning.

- **Entity Popularity**

This feature answer us the following question: how often a candidate entity appears given an entity mention m?

It based on that people generally when use a word they mean (mention to) in most cases a single meaning. For example the entity **New York**, the candidate entity **New York** (film) is much rarer than the candidate entity **New York City**, and in most cases when people mention **New York**, they mean the city of **New York** rather than the film whose name is also **New York**.

Several researchers depending in this fact and define the popularity feature Pop (e_i) as the following:[18]

$$Pop(e_i) = \frac{count_m(e_i)}{\sum_{e_j \in E_m} count_m(e_i)} \qquad (1.3)$$

Where count m (e_i) is the number of links which point to the entity e_i and have the mention form m as the anchor text.

- **Entity Type**

From the brilliant features that we face the entity type feature, it is beneficial because it based in the fact that if the entity mention m has the type x, so that it points to an entity that have the same type x. A problem may appears when we can't find the candidate entity's type from the knowledge base. To deal with, some system used Named Entity Recognizer to identify it.

1.3.4.1.2 Context-Dependent Features

For sure Context-Independent Features have a great value and good results to mapping an entity mention with its coordinate in a knowledge base, but it's useful to exploit the entity mention context and retrieve information from its surroundings sentences and words to get better results. Here in this part, we discuss two features which are textual context and coherence between mapping entities.

- **Textual Context**

There are two form to represent the context of entities that are bag of words and concept vector in the aim to know if the entity mention m is similar, not similar or has a certain percentage of similarity to a candidate entity e belongs to E_m.

The first form is bag of words where the context of an entity is represented as a list of words that surrounding the entity where appears or from the whole document. For the entity mention the context is collected from the local document. but for the candidate entities the context is collected from the summary of its Wikipedia page or from the entire page. Another form is concept vector where each entity's context (Whether it is a mention entity or a candidate entity) represented as a vector. A vector a numeric representation of a text.[18]

- **Coherence between Mapping Entities**

In each document there are a set of entities M we wish to link each one of them to another entity in a knowledge base KB. Fortunately, in most cases there is a relation and meaning between them that could lead and help us to do a perfect and success linking.

One can say that two entities are coherence and have a semantic relation if there are many Wikipedia articles that link to both. The coherence of a candidate entity e with each entity $e' \in E$ is calculated using the following formula of topical coherence which is an adoption of Normalized Google Distance:[18]

$$Coh_G(U_1, U_2) = 1 - \frac{log(max(|U1|, |U2|)) - log(max(|U1 \cap U2|))}{log(|WP|) - log(min(|U1|, |U2|))} \qquad (1.4)$$

Where U_1 and U_2 are the sets of Wikipedia articles that link to u_1 and u_2 respectively, and WP is the set of all articles in Wikipedia.

1.3.4.2 Supervised ranking methods

A supervised algorithm it is an algorithm trained on input data that has been labeled for a particular output (training data set). And it is good at classification and regression problems such as determining what category a news article belongs to. For our task (Entity linking) the training data set was constituted by set of pairs of entity mention with its mapping entity.

We talk about two categories of methods which are Binary Classification Methods and Learning to Rank Methods.

1.3.4.2.1 Binary Classification Methods

With this approach the candidate entity ranking problem was considered as a binary classification problem where the goal is to check if an instance is true or false (zero or one), in other word the output label takes only two different classes or values.

To adopt our problem with binary classification, a data set was constructed which contains an entity mention m with a candidate entity e_i and an output label to decide whether the entity mention really refers to the candidate entity. If it is, it takes a positive value, otherwise it take a negative value. We want to mention to that each entity is represented as a numerical vector.

Like any supervised technique, a model will be generated based on that data set. So that it takes each labeled pairs (m, e_i) and used it to learn the classifier. After this important step, a test data (pairs of entities) will passed to the classifier and return us the label (positive or negative) for each one. Because no technique is perfect, a problem appears out of nowhere which is the possibility of appearance of two or more positive label for one pairs. To deal with it many system used SVM (Support Vector Machines) ranking models.

Constructing data set with this method leads to be unbalanced due to the presence of many candidate entities from which we choose just one entity to be the mapping entity. For instance if the candidate entity set's size equal to 20 (generate a data set of 20 lines), so there is one positive instance and nineteen are negative, and so on. The problem was that the model does not learn what makes the other class different and fails to understand the underlying patterns that allow us to distinguish classes. That drive the researcher to think of other ways such as learning to rank methods.

1.3.4.2.2 Learning to Rank Methods

In this part we describe an approach to disambiguating terms that occur in plain text, so they can be linked to the appropriate Wikipedia article. We talking about learning to rank approach.

Learning to rank (LTR) is a class of algorithmic techniques that apply supervised machine learning to solve ranking problems in search relevancy, and the following image shows the typical ”LTR” flow.[12]

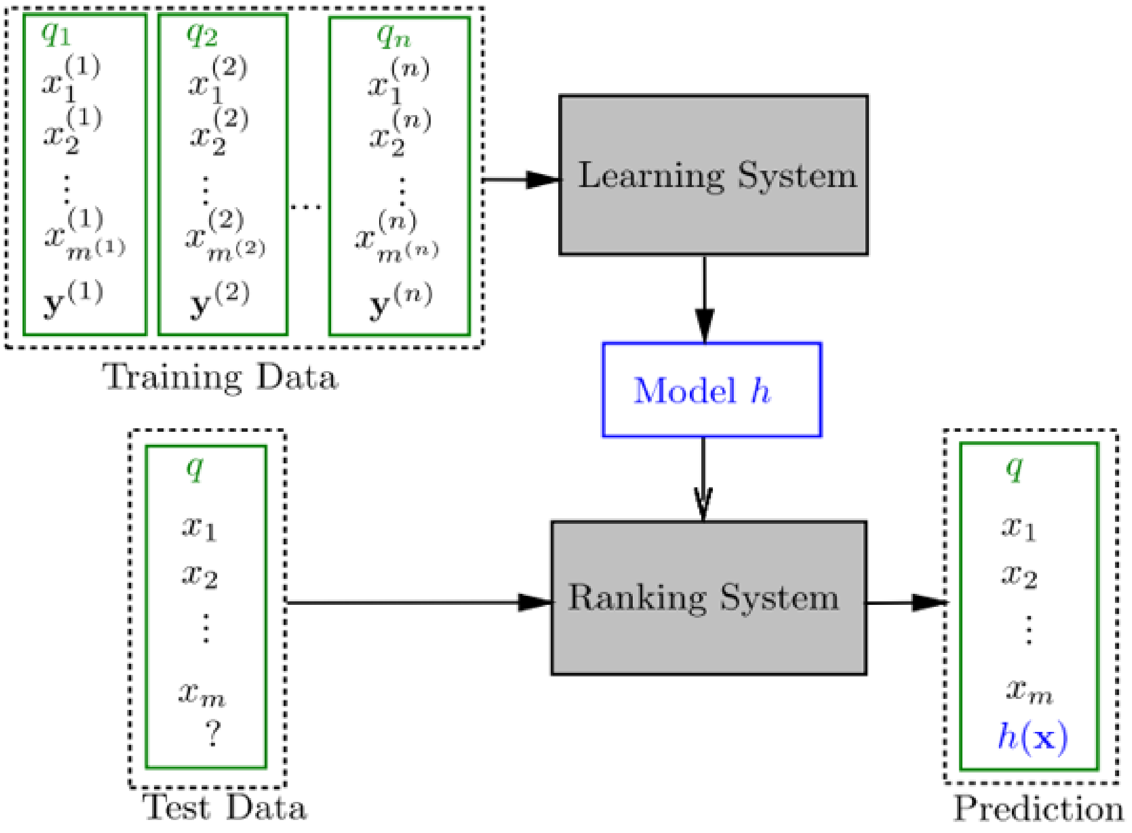


Figure 1.5: the typical “learning-to-rank” flow

Obviously a data set was needed for learning and testing process includes a set of pair of entity mention and a candidate entity and the output is a score label indicates how much the two entities are relevant to each other (for example we can take the number of clicks on a candidate entity when a query was executed with the entity mention as the value of that label). The correct mapping entity must have the higher score. Each pair is represented as a vector. To do the ranking process supervised algorithms will be used such as Polynomial Regression Function.[12]

In this pointwise approach we do not take into consideration the relation between candidate entities, rather than in pairwise approach where the data set examples include the entity mention m and a pair of candidate entities (e_i, e_j) and a label that indicates to one if e_i is more relevant to the entity mention m then e_j, otherwise the label indicate to zero.

1.3.4.3 Unsupervised ranking method

The unsupervised learning algorithms aims to identify patterns in data sets without having any external guidance in performing that task. We talk about two categories of methods which are VSM Based Methods and Information Retrieval Based Methods.

1.3.4.3.1 VSM Based Methods

Vector space models are used to represent words in a continuous vector space A vector space contains a collection of objects called vectors which are numerical representations of words, sentence, and even documents. While a simple vector only has two dimensions, those used in natural language processing can have thousands.[18]

The vector space model is an algebraic model that represents objects (like text) as vectors. This makes it easy to determine the similarity between words or the relevance between a search query and document. Cosine similarity is often used to determine similarity between vectors.

To apply the unsupervised techniques with VSM they generate a representational vector to each of entity mention and candidate entity. Use a similarity measure to select the correct candidate entity like TF-IDF similarity. At final process it cluster the candidate entity with entity mention in the same cluster.

1.3.4.3.2 Information Retrieval Based Methods

Information Retrieval is the producers of finding the most relevant document that has or include the target information the query executer (ex: a person) want to see.

In another more accurate way, Information retrieval (IR) involves retrieving information from stored data, through user queries or pre-formulated user profiles. The information can be in any format. IR typically advances over four broad stages which are identification of text types, document preprocessing, document indexing, and query processing and matching the same to documents.

Some researchers think to use methods that are used in information retrieval field to deal with entity linking problem as an information retrieval problem. In general the most of them follow the following way to represent entities where “each candidate entity is indexed as a separate document, and for each entity mention, they generate a search query from the entity mention and its contextual document. Finally, the search query is given to the candidate entity index. and the candidate entity which has the highest relevant score is retrieved as the mapping entity for the entity mention“.[18]

1.3.5 Unlinkable Mention Prediction

As we discuss earlier an entity linking system described by its two main process which are candidate entity generation and candidate entity ranking. In the first process some methods generate the set of candidate from a knowledge base, and for the second process choose the high ranked entity to be the correct mapping entity for a mention 'm'. until now it looks good, but what if the set of candidate E is Nil and does not include any candidate, also what if the system use other method to generate the candidate set which is not related to a knowledge base and when it pick up the mapping entity notice that the entity does not have a counterpart in the knowledge base.

Unlinkable Mention Prediction is the answer of these questions, for the two cases many researchers mark the mention m as Unlikeable and the case is closed. But this effect on the measures should one use when evaluate the entity linking systems which we will discuss them later in chapter three.

Others include Unlinkable Mention Prediction process in the ranking process, for that there is no problem unless the set of candidate entities E is Null meaning does not include any candidate, they do nothing about that and as simple they mark the mention as Unlikable.[18]

A reason to mark a mention as an Unlinkable entity is when the high succored candidate does not exceed the NIL threshold which is a score value should a candidate surpasses to pick it up as the correct mapping entity. There is measures to fix the value of threshold τ.

After mark an entity as Unlinkable some researcher try to predict its corresponding by using the methods that used in the candidate entity generation such as SVM classifier.

Other solution may be used is to added a NIL entity into the candidate entity set, and considered NIL as a distinct candidate. If the ranker outputs NIL as the top-ranked entity, this entity mention is considered as unlinkable. Otherwise, the top-ranked entity is returned as the correct mapping entity.

1.4 Conclusion

In this chapte, we reviewed and summarize the significance NLP, some of its applications and briefly explained NER. Then, we have presented a comprehensive survey for entity linking. Specifically, we have surveyed the main approaches utilized in the three modules of entity linking systems (Candidate Entity Generation, Candidate Entity Ranking, and Unlinkable Mention Prediction).

Chapter 2

Related Work

2.1 Introduction

Recently, a number of systems have been proposed for linking entity mentions in text to Wikipedia pages. Such systems typically search for candidate entities and then disambiguate them, returning either the best candidate or nil.[9]

In this chapter we take a look on what they did in this field, we discuss some NEL system such as Bunescu and Pasca system, Cucerzan. Each one (researcher) has an idea and how to implement it that create the difference which is a good difference should leads to excellent systems.

2.2 Bunescu and Pasca

2.2.1 Extractor

They use Wikipedia to extract mentions and decide whether the entity is a named entity or not. For that some rules are proposed and put to work on, which are:[3]

If e is a multiple word so that all word must be in capital case to consider e as a named entity.

If e is one word so that the word should include two letters are in capital case to consider e as a named entity.

If e is one word and does not contain two capital letters, and the word repeated multiple times in the article by more than 75 % are capitalized, so e is a named entity.

2.2.2 Searcher

They started from the principle of that articles in Wikipedia has hyperlinks which refer to other entity, wherefore the text anchor in a hyperlink is considered as an entity mention. They build a named entity dictionary D constructed as a list of pair of a string entry d which is mapped to a set of entities d.E that can be denoted by d in Wikipedia. So they take the set d.E as the candidate entity set if there is a dictionary entry matching d, where d is an entity mention extracted from hyperlinks in a Wikipedia article.[3]

2.2.3 Disambiguator

In the candidate entity ranking process they used an SVM (Support Vector Machine) ranking model.

They use cosine similarity between the context of the mention 'q' and the text of the Wikipedia article for candidate e_k as the first feature, like the following function:

$$score(q, e_k) = \cos(q.T, e_k.T) = \frac{q.T}{\|q.T\|}\frac{e_k.T}{\|e_k.T\|} \tag{2.1}$$

Where q.T and $e_k.T$ are represented in the standard vector space model. A term w in V and its score (weight) is the tf-idf score will construct the VSM. V is generated depending on all Wikipedia article and ignore some words that appear many times or rare appear.

After that they denote errors with the first method, so they exploited Wikipedia categories to get rid of these errors. For that they calculate the correlation between context words and the categories to which e_k belongs. And add it as second feature.[3]

Evaluation

They do a good work and achieve 84.8 % of the accuracy measure.

2.3 Cucerzan

2.3.1 Extractor

The ability to determine the named entities in a text has been established as an important task for several natural language processing areas, including information retrieval, Cucerzan uses a hybrid NER tagger based on capitalization rules, web and the CoNLL-03 NER[16] shared task data statistics.[9]

At the first function he takes the document and break it up into sentences, and determines whether the beginning of each sentence is an entity (may just a part of entity) or it is capitalized only because it the first word of a sentence. This is done based on statistics extracted from a one-billion-word corpus, with back-off to Web statistics.[4]

2.3.2 Searcher

Cucerzan uses three features that are entity surface forms, category tags, and contexts. Surface forms set of an entity is considered as candidate entity set and there are four Wikipedia features to generate this set, which are: the titles of entity pages, the titles of redirecting pages, the disambiguation pages, and the references to entity pages in other Wikipedia articles.

2.3.3 Disambiguator

For disambiguate and linking the entities appear in a text he use the contextual and category information extracted from Wikipedia.[4]

His idea revolves around the maximization of the agreement between the contexts of all candidate entities of all mentions appear in a document D and the context of the document D. And also the maximization of the agreement between categories tags of any two candidate entities. Each candidate entity is represented as a vector space model $\delta_e \in \{0,1\}M+N$ where δ_e is defined as the following:[4]

$$\delta_e{}^{\mathrm{i}} = \begin{cases} 1, & \text{if } c_i \text{ is a context for entity e} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

$$\delta_e{}^{\mathrm{M+j}} = \begin{cases} 1, & \text{if } t_j \text{ is a category tag for e} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

1 < i < M ; 1 < j < N

He choose the set of entities $(e_1, e_2, ..., e_n) \in \varepsilon(S_1)X\varepsilon(S_2)X...X\varepsilon(S_n)$ for which they achieve the highest score or result of the following function:

$$\underset{\substack{(e_1,...,e_n)\in \\ \varepsilon(s_1)..\varepsilon(s_n)}}{\arg\max} \sum_{i=1}^{n} < \delta_{e_i}|_C, d > + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} < \delta_{e_i}|_T, \delta_{e_j}|_T >, \tag{2.4}$$

We define the argmax of a function f defined on a set D as:

$$\underset{x\in D}{\arg\max} f(x) = \{x|f(x) \geq f(y), \forall y \in D\} \tag{2.5}$$

The arg max function returns the argument or arguments (arg) for the target function that returns the maximum (max) value from the target function.[17]

Evaluation

This disambiguation system is evaluated on two set of data. One of them is a bunch of article from Wikipedia (350 Random Article) Where achieve 88.3 % in the accuracy measure and the other is a set of news stories where obtained a good result and achieve an accuracy of 91.4 %.

2.4 UKP-UBC Entity Linking at TAC-KBP

2.4.1 Extractor

The first component in this system is the Preprocessing component in which they used tokenization, POS-tagging, lemmatization, chunking, and named entity recognition tools with the helping of the open source project DKPro Core.[7]

DKPro Core[1] was initiated by the Ubiquitous Knowledge Processing Lab (UKP) at the Technische Universität Darmstadt. And it is a software based on JAVA programming language.

2.4.2 Searcher

Their method is a dictionary based method where used two types of them, one is constructed depends on Wikipedia and the other one depends on Google search logs. In the candidates retrieval process the system search about the mention m in the dictionaries, once found it, its value should be added to the set of candidate entities E for the mention m.[7]

A question may come to mind which is what if the searcher component doesn't found the mention in any dictionary? They handle this case by remove parentheses and text in between and do a research again in dictionaries. If the problem is still appear they remove leading " the " from mention and do a research again in dictionaries. If all this solution does not lead to a result the mention m is marked as an unlinkable entity.[7]

2.4.3 Disambiguator

In their study they tried many approaches and features. They use the score of one feature and make result depends only on it. They also use a linear combination of many or all features with a supervised approaches.[7]

A type of features they talked about was the similarity feature where used DKPro similarity software to calculate a score between a mention m and its candidate entities. They apply Levenshtein and Jaro-Winkler distance. This feature is useful for misspellings.[7]

When they was helped by a context feature they used a method similar to Han and Sun (2011) method where words in the context of the mention are compared to words in the context of the link anchors pointing to the article of each candidate entity in Wikipedia. Additionally to that, mention context words are weighted using the frequency in the description and the candidate entity that acquire the high score will be chosen as mapping entity for the mention m.[7]

In a third method they convert a Wikipedia version to a graph G = (V, E) where V representing concepts (Wikipedia articles) and E representing relations (edges) between them, and for more explanation in this point we can say that there is a relation between an article a1 with another one a2, if there exist links in both directions between a1 and a2 in Wikipedia. The graph contains 2,325,876 vertices and 5,549,696 edges. With this approach they attend to disambiguate all mentions that appear in the document collectively. For that the Personalized PageRank algorithm that took a graph as its input was used and they implement it using the UKB package. These ranks are used as input features for the classifier Rank SVM (Tsochantaridis, 2006).[7]

[1]https://dkpro.github.io/dkpro-core/

Evaluation

The result of this system is poor where they have only achieved 31 % in text data, because of a problem in the candidate identification process.[7]

2.5 Lightweight Multilingual Entity Extraction and Linking

2.5.1 Extractor

They use Simple CRFs, with only a few characteristics that may be easily extended to other languages. Their approach performs similarly to state-of-the-art single-language NER systems while being more adaptable to additional languages and computationally economical.[14]

They utilize word2vec with the Continuous-Bag-of-Words (CBOW) method, 5 iterations, and a window size of 5 token to learn nothing embeddings for each language. Morphological characteristics, such as word form and capitalization, token prefixes and suffixes (up to length 4), numerals, and punctuation, are also used. Finally, they try out language-specific part-of-speech (POS) tags, which need very little preprocessing and are available in over 40 languages.[14]

2.5.2 Searcher

They presume that each entity has a fundamental form (CF) in KB. The Wikipedia page address corresponding to each KB entity is used as the base model in their studies. Candidate entities are retrieved in most NEL systems if their basic form or alias is comparable to the tag (extended) text. And they used entity mergers for the candidate entity retrieval system, which is based on the Fast Entity Link (FEL), and which they briefly present below.[14]

"Fast Entity Linker (FEL) is an unsupervised approach which selects the segmentation of an input sequence of words that maximizes the likelihood of all substrings linking to an entity in the Knowledge Base. The model requires to calculate the conditional probabilities of an entity given every substring of the input sequence, but avoids computing entity to entity joint dependencies, which makes the process very efficient. As a byproduct of this segmentation, the model selects the most likely entities that would be linked to each substring in a sequence of words."[14]

2.5.3 Disambiguator

In this work, they compare three general purposes and efficient methods: **the forward-backward algorithm**, **exemplar clustering**, and **label propagation**.

The forward–backward algorithm is an inference algorithm for hidden Markov models which computes the posterior marginals of all hidden state variables given a sequence of observations/emissions. This inference task is usually called smoothing. The algorithm makes use of the principle of dynamic programming to efficiently compute the values that are required to obtain the posterior marginal distributions in two passes. The first pass goes forward in time while the second goes backward in time; hence the name forward–backward algorithm.[14]

They employ exemplar clustering, which allows them to choose candidates for cluster centroids and allocate mentions to these clusters. A preference vector with greater (and positive) values for candidate entities' entity embeddings and zeros for mentions' entity embeddings can be used to start this process.[14]

"**Label propagation** is an umbrella term for a family of graph-based semi-supervised algorithms. A label propagation algorithm propagates labels to unlabeled nodes in a graph, starting with a few labeled nodes."[14]

Evaluation

They show the accuracy of their approach using data from the TAC KBP 2013 multilingual data set (English/Spanish/Chinese) as well as Aida, a typical monolingual data set.

In the TAC KBP-EN datasets, FWBW method achieves an F-measures the highest percentage (61 %) compared to the others methods (1-best, Exemplar, LabelProp), while in the TAC KBP-ES datasets, 1-best achieves the highest percentage (67.3 %) compared to the others.[14]

2.6 AGDISTIS

2.6.1 Extractor

As shown in Figure 2.1, the first step is to enter an input text T and a named entity recognition function (e.g., FOX[20]), they begin by retrieving all named entities from the input text.[21]

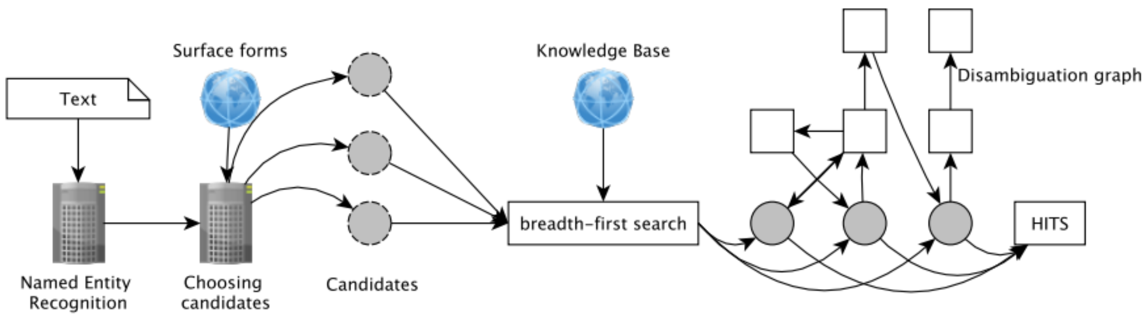


Figure 2.1: overview of AGDISTIS

2.6.2 Searcher

They first identify potential resources in the KB in order to determine the right disambiguation for a certain collection of named entities. They start by making an index of all the labels for each resource. Any collection of attributes can be used as labeling properties in their method.[21]

2.6.3 Disambiguator

AGDISTIS is a novel named entity disambiguation method that is independent of the knowledge base. The Hypertext-Induced Topic Search (HITS : which calculates the most pertinent entities) method is combined with label expansion algorithms and string similarity metrics in their approach. AGDISTIS can efficiently recognize the proper URIs for a given collection of named entities inside an input text.[21]

Evaluation

They compared AGDISTIS approach with TagMe 2 and DBpedia Spotlight on four different datasets are (AIDA/CONLL-TestB, AQUAINT, IITB, MSNBC) using micro F-measure (F1).[21]

Their approach achieves an F-measures 0.59 (AIDA/CONLL-TestB), 0.54 (AQUAINT), 0.31 (IITB), 0.76 (MSNBC), While TagMe 2 and DBpedia Spotlight achieves the lowest percentage of three datasets (AIDA/CONLL-TestB, AQUAINT, MSNBC), as for IITB datasets, DBpedia Spotlight recorded the highest percentage.[21]

2.7 Conclusion

In summary, in this chapter we reviewed works that are related to entity linking systems that was created and implemented previously. Some of them provided good results, and some of them encountered problems and did not achieve the required. Anyway, this domain still developing and definitely a significant number of systems will reveal in the future to repairing imperfections and loopholes of the ancient systems.

Chapter 3

Disambiguation and Linking System

3.1 Introduction

This chapter is the main chapter in this project where we will dive into our work and describe our systems and experiments that we have done. Also we walks through datasets that we have created and talk about the approach we followed to make this work done.

3.2 Datasets

Numerous numbers of datasets are available online for the research community to train and evaluate their approaches. Finding the right dataset that contains all words of a language or all the situation and context where a mention may appear. And with the right dataset properties in terms of usability format and labelling results, is not an easy task.

We found two dataset which are KWDW[1] and YAGO[2], we choose to work on KWDW dataset because it has a simple structure and it is well documented.

3.2.1 Dataset structure

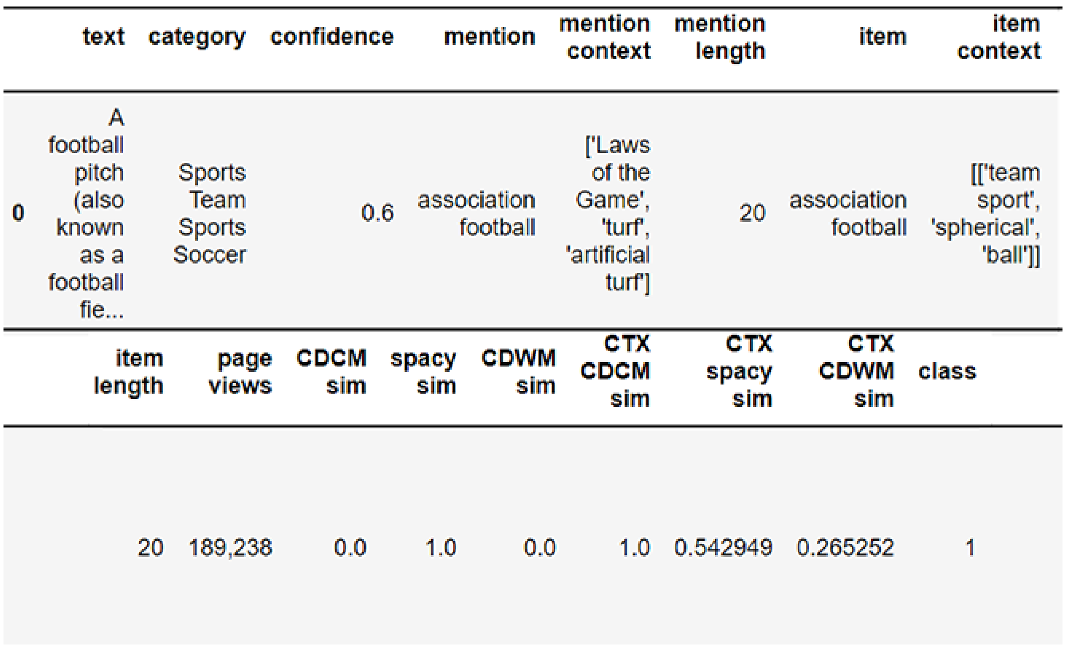
At first glance, we thought to use the dataset as it is. But in fact, it did not meet our requirements at all, which prompted us to change its structure and with emphasis only on the information that we need.

We will not talk about KDWD structure, instead of that we will talk and discuss KWDW_R dataset which is original from KDWD but in other structure.

Before we start, we want to mention to that we have created another dataset with the same structure but the origin was different which is Wikipedia website. We talk about it in details in the 'TD Dataset Creation process' section.

In this figure 3.1 we present an example of it:

[1]https://yago-knowledge.org/downloads/yago-4
[2]https://www.kaggle.com/kenshoresearch/kensho-derived-wikimedia-data



	text	category	confidence	mention	mention context	mention length	item	item context
0	A football pitch (also known as a football fie...	Sports Team Sports Soccer	0.6	association football	['Laws of the Game', 'turf', 'artificial turf']	20	association football	[['team sport', 'spherical', 'ball']]

	item length	page views	CDCM sim	spacy sim	CDWM sim	CTX CDCM sim	CTX spacy sim	CTX CDWM sim	class
0	20	189,238	0.0	1.0	0.0	1.0	0.542949	0.265252	1

Figure 3.1: example of dataset structure

In the following lines we will give a description of each column in dataset:

Text: a paragraph from Wikipedia contains mentions.

Category: describe the type of the text, it may be Science, Mathematics, Sports or any other category. For this feature we use Google Natural Language API[3] which is machine learning project to classify texts into categories. This API given us the opportunity to distinguish texts a little more from each other. The following snippet shows how to call the API.

```python
from google.cloud import language_v1
def classify(text):
    Document = language_v1.Document(
      Content = text, type_= language_v1.Document.Type.PLAIN_TEXT
    )
    language_client = language_v1.LanguageServiceClient()
    response = language_client.classify_text(request={'document': document
   })
    return response
```

Listing 3.1: retrieve category function

Confidence: It express the quality of Google classifier. The closer to one, the better.

Mention: it is a named entity appears in the text.

Mention length: Is the length of mentions.

Mention context: we choose a window of 3 mentions that appear in the same text and save it as mention context.

[3]https://cloud.google.com/natural-language/docs/setup

item: It may be referred to it by the mention, and we extracted it from the dictionary.

item length: is the length of the candidate entity length.

item context: we choose a window of 3 entities which are appear in Wikipedia page related to the candidate and save it as candidate context.

Page views: how many times the Wikipedia page is visited.

CDCM sim: (Cosine Distance Count-vectorizer Method). It is the result of cosine distance between the mention and candidate. We implement the following snippet of code to do the work.

```python
def CosineDistanceCountvectorizerMethod(m, e):

    allsentences = [m , e]

    # text to vector
    vectorizer = CountVectorizer()
    all_sentences_to_vector = vectorizer.fit_transform(allsentences)
    text_to_vector_v1 = all_sentences_to_vector.toarray()[0].tolist()
    text_to_vector_v2 = all_sentences_to_vector.toarray()[1].tolist()

    # distance of similarity
    cosine = distance.cosine(text_to_vector_v1, text_to_vector_v2)

    return cosine
```

Listing 3.2: calculate CDCM similarity function

Spacy sim: is the result of similarity calculation between the mention and the candidate using Spacy library. We implement the following snippet of code to do the work.

```python
def SpacySimilarity(mention, candidate):

    doc1 = nlp(mention)
    doc2 = nlp(candidate)

    if (doc1.has_vector) & (doc2.has_vector):
        return doc2.similarity(doc1)
    else:
        return cosine_distance_wordembedding_method(mention,candidate)
```

Listing 3.3: calculate spacy similarity function

CDWM sim: (Cosine Distance Word-Embedding Method) is the result of the application of the code below and express how much the mention and the candidate entity are similar to each ether using CDW method. We using Glove-Model[4] for word embedding. We implement the following snippet of code to do the work.

[4]https://www.kaggle.com/watts2/glove6b50dtxt

```python
def cosine_distance_wordembedding_method(s1, s2):
    try:
        vector_1 = np.mean([model[word] for word in preprocess(s1)],axis=0)
        vector_2 = np.mean([model[word] for word in preprocess(s2)],axis=0)
        cosine = scipy.spatial.distance.cosine(vector_1, vector_2)

        return cosine
    except:
        return CosineDistanceCountvectorizerMethod(s1,s2)
```

Listing 3.4: calculate CDWM similarity function

CTX CDCM sim: it similar to CDCM sim but the calculation of similarity was between the context of the mention and the context of the candidate.

CTX spacy sim: it similar to Spcay sim but the calculation of similarity was between the context of the mention and the context of the candidate.

CTX CDWM sim: it similar to CDWM sim but the calculation of similarity was between the context of the mention and the context of the candidate.

Class: if it equal to 0 means that the candidate is not the correct mapping entity to the mention. Else(equal to 1) means that the mentions refer to that candidate in that context.

3.2.2 Create Dictionary

We have created three dictionaries one for the First Dataset (KWDW_R Dataset) and the other two for the Second Dataset (our TD Dataset). Below we will explain the steps that we have followed for creating dictionaries in detail:

Note: TD is acronym of our names (Toukali and Djaafri), R is acronym of reconstructed.

3.2.2.1 Dictionary for kwdw dataset

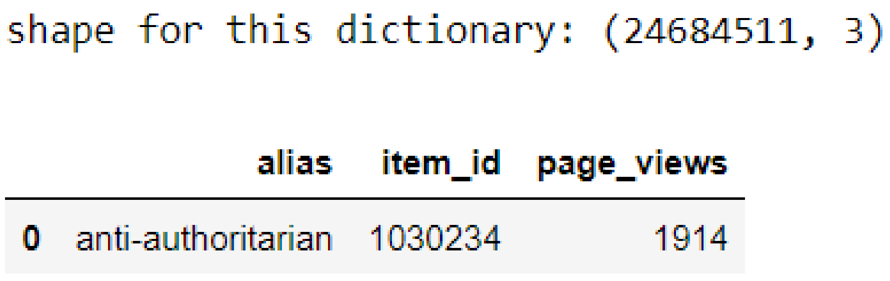
This dataset was created depending on KWDW dataset, exploited two files out of seven. And it has 3 steps which are:

- Extract the text, alias and target id from ”link annotated text.jsonl”.

- Step two we take the target id we extracted from the jsonl file and compare it with the targat ids in the file ”page.csv”. When we find the same target id, we get (page views and item id).

- Put all information that extracted and calculated to a CSV file ”KWDW_R Dictionary.csv”, which consists of (alias, item id and page views).

This figure 3.2 shows all the information we saved for one row in the ”KWDW_R Dictionary.csv” and shape for this dictionary:



```
shape for this dictionary: (24684511, 3)
```

	alias	item_id	page_views
0	anti-authoritarian	1030234	1914

Figure 3.2: example of kwdw dictionary

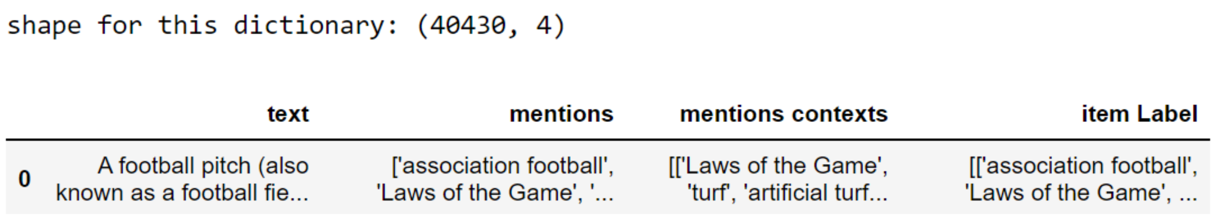
3.2.2.2 Two dictionary for our dataset

We created these two datasets from scratch exploiting the large number of articles in Wikipedia. We follow these steps to make this work done:

At first, we consulting a Wikipedia page, whatever it was. Then we extract all paragraphs in that page and consider it as texts. When extracting these texts, we find that there are texts that do not have any link. In this case we cancel those texts, because we don’t need those texts to create the Dictionary. After that we extract all hyperlinks that are in each paragraph and filter it; that is, delete all the duplicates or wrong hyperlinks. Saving only the links linking to another Wikipedia page. At this point, we extract the context for each mentions from all texts, then we take the first mention hyperlink from the first text and going to the next page via that it. When entering this page, we extract context for an item, then we extract the hyperlink of the page information and enter through it to the next page, in which we find all the information about it, and from here we retrieve the (page id, page views and wiki data item id), from the last we extract his item hyperlink, which helps us retrieve Item label When we finish retrieving all of this information that we mentioned.

- We save part of the information in a CSV file ”first dictionary.csv”, which consists of (text, mentions, mentions contexts and Item label).

This figure 3.3 shows all the information we saved for one row in the ”first dictionary.csv” and shape for this dictionary:



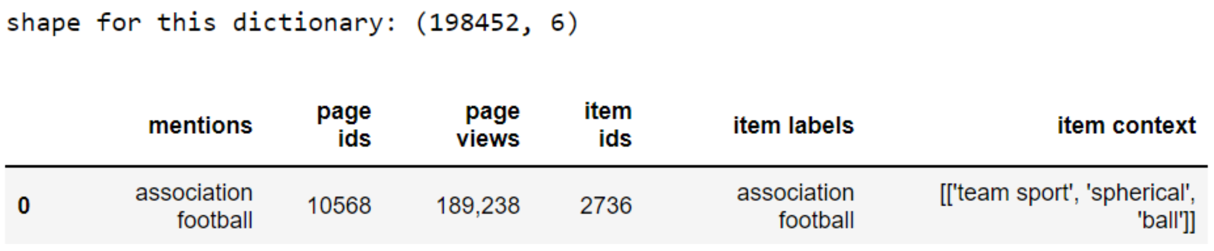
```
shape for this dictionary: (40430, 4)
```

	text	mentions	mentions contexts	item Label
0	A football pitch (also known as a football fie...	['association football', 'Laws of the Game', '...	[['Laws of the Game', 'turf', 'artificial turf...	[['association football', 'Laws of the Game', ...

Figure 3.3: example of first dictionary

- And the second part in another CSV file ”second dictionary.csv”, which consists of (mentions, page ids, page views, item ids, item labels and item context).

This figure 3.4 shows all the information we saved for one row in the ”second dictionary.csv” and shape for this dictionary:



```
shape for this dictionary: (198452, 6)
```

	mentions	page ids	page views	item ids	item labels	item context
0	association football	10568	189,238	2736	association football	[['team sport', 'spherical', 'ball']]

Figure 3.4: example of second dictionary

Then we repeat this process on all hyperlinks in all texts. When you finish saving all information from the first page, we are going to the second page via the first mention hyperlink. And we do the same process that we mentioned above.

3.2.3 Dataset Creation process

3.2.3.1 KWDW_R Dataset Creation process

This dataset was created depending on KWDW dataset, exploited four files out of seven. And it has 8 steps which are:

- Extract the text and mentions from ”link annotated text.jsonl”.

- Execute Google API given it the text as input and retrieve a category as output.

- Console the ”kwdw Dictionary.csv” to get the page views and item id.

- Generate mention context.

- Generate candidate entities for each mention with the help of the dictionary.

- Extract the candidate context from its Wikipedia page.

- Calculate the similarities between mention and each candidate.

- Put all information that retrieved and calculated to a final CSV file ”KWDW_R Dataset .csv”.

- Repeat until the text number 5000 in the ljson file.

3.2.3.2 TD Dataset Creation process

This dataset was created depending on two files (first dictionary.csv and second dictionary.csv), and it has 4 steps which are:

- Recover first text to mention, mention contexts and Item label from the file ”first dictionary.csv”.

- Step tow we take the mention we extracted from the first CSV file and compare it with the mentions in the file ”second dictionary.csv”. When we find the same mention, we get (mentions, page ids, page views, item ids, item labels and item context).

- Execute Google API given it the text as input and retrieve a category as output.

- Calculate the similarities between mention and each item.

This explanation is on one row, when we extracted this information we put it in a final CSV file ”TD Dataset.csv”, then we repeat the same process on all rows.

3.3 First Experiment

As a first experiment that we tried, we started our exploration for this domain (Named Entity Linking) by created a basic system that depends on and Dictionary and Search Engine method to generate the set of candidates regarding to a mention. And depends on page views and popularity features of a candidate to choose the appropriate candidate that likelihood being the mapping entity for a mention.

3.3.1 Named entity recognition

NEL and NER are two parts inseparable, because it’s axiomatic that there is no way to linking any word if we did not run the NER process and extract the entities from a text. So we can say that NER is the first step and NEL is its complement.

3.3.1.1 Spacy NER Implementation

Because it’s not our main part that we should tackle with, we just used the spacy implementation to extract entities. As we know there are many types of entities such person, organization, state ...etc.

In Spacy library they define a set of types from which a type will be assigned to an extracted entity. For a reason we dropped some types like date and time - they might don’t help us - and just used the following ones:

TYPE	DESCRIPTION
PERSON	People, incuding fictional
NORP	Nationalities or religious or political groups
ORG	Companies, cities, states
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, ect (Not services)
EVENT	Named hurricanes, battles, wars, sports events, etc

Table 3.1: Spacy types

Using this snippet of code do the work perfectly:

```python
import spacy

def GetEntities(sentence):
    doc = nlp(sentence)
    entities = []
    for ent in doc.ents:
        if ent.label_ == 'ORG' or ent.label_ == 'GPE' or ent.label_ == 'PERSON' or ent.label_ == 'NORP' or ent.label_ == 'FAC' or ent.label_ == 'LOC' or ent.label_ == 'PRODUCT' or ent.label_ == 'EVENT' or ent.label_ == 'WORK_OF_ART' or ent.label_ == 'LAW':
            entities.append(ent)
            #print(ent.label_)
    return entities
```

Listing 3.5: Spacy NER code

3.3.1.2 Evaluation

Unfortunately, when we evaluated the result we didn't get all mentions that supposed to be extracted from texts. A set of metrics was applied to evaluate the Spacy NER System.

This table 3.2 shows the results on 3000 texts from KWDW dataset:

	precision	**Recall**	**f1 measure**	**Accuracy**
Spacy NER System	0.66	0.68	0.67	0.67

Table 3.2: Spacy NER evaluation

3.3.2 Candidate entity generation

Candidate entity generation is the second step in the pipeline, but in our case it became the first one. We take a text with mentions that appeared in it. And we trying two different methods to retrieve the set of candidate entities.

3.3.2.1 Search engine method

In this method we utilized a Wikipedia Python API[5] to execute a search on Google search engine and excluded the result to be just from Wikipedia and we took only the first twenty page as candidate entities.

[5]https://www.mediawiki.org/wiki/API:Search

3.3.2.2 Dictionary method

For this method we enlisted a dictionary contains a thousands of couples (mention, candidate) we'll talk about in Datasets section.

we did a search operation to find the mention in that dictionary and retrieve the appropriate candidate and add it to the set of candidate entities.

3.3.3 Named entity linking

The third process (second in our case) is linking the entity that appears in a text to one from bunch of entities which was nominated to be the appropriate one to the mention. For this step we exploited two features separately.

3.3.3.1 Using Page views feature

In this approach we utilized page views feature as the only criterion to determine which candidate is the correct mapping entity to the mention. We used it when generating the set of candidate entities with search engine method. Why? Because this information (page views) is accessible and easy to retrieve directly from Wikipedia page. If fact is still remaining one more step but it remains an easy operation. As simple the page that has the higher views is the page that was selected to be the suitable candidate.

3.3.3.2 Using Popularity feature

In this approach we utilized popularity feature, we talked about it in chapter one, to decide whether a candidate entity is the correct one or not, and it's defined as the following:

$$Pop(e_i) = \frac{count_m(e_i)}{\sum_{e_j \in E_m} count_m(e_i)} \tag{3.1}$$

We used this feature when generating the set of candidate entities with the helping of the dictionary. Why? For the same reason we choose page views with search engine because it's easy and simple, we choose popularity with dictionary because is somehow simple to retrieve the count of a candidate regarding to a mention. In the other hand it is a little complex to bring back the page views. In view of this fact we just be content with popularity.

Note: you might be wondering what is relation between a candidate entity and a Wikipedia page, here is the answer. At first, an item is an object in Wikidata[6], it has title and description and other properties. And it is related only to one Wikipedia page.

[6]https://www.wikidata.org/

Second, in this problem we consider that a mention refers to one item or more and candidate entities are items.

Candidate entity $\Rightarrow is \Rightarrow Item \Rightarrow relatedto \Rightarrow Wikipediapage \Rightarrow has \Rightarrow viewsnumber$

3.3.3.3 Evaluation of two methods

We evaluated this two methods on 3000 texts include X mentions from KWDW (Kensho Derived Wikimedia Data) dataset.

These tables shows the results:

	precision	**Recall**	**f1 measure**	**Accuracy**
page views with search engine method	0.44	0.47	0.46	0.45

Table 3.3: evaluation of first method

	precision	**Recall**	**f1 measure**	**Accuracy**
popularity with dictionary method	0.56	0.57	0.55	0.53

Table 3.4: evaluation of second method

This results did not satisfy our needs and did not match with our goals. So we thought to create a new system with other approach which is what it was. And we will get across it in the next sections.

3.4 Second Experiment

Finally, we got to this section, where we will talk about our efficient system using some of machine learning methods such logistic regression and random forest.

It is important to know the data you work on. So we have started with this essential process where we dived into the data and explored it with a good manner. And In order to understand our data, we looked at each feature (column) and have tried to understand their meaning and relevance to the class column.

3.4.1 Dataset Correcting

From those charts below, we can conclude that there are many feature aren't configured yet to depend on them and train our model. We can see that the 'spacy sim' feature does match our perspective because it should create a graph similar to logistic graph. It is the same thing with 'CDCM sim' and 'CWCM sim' features.

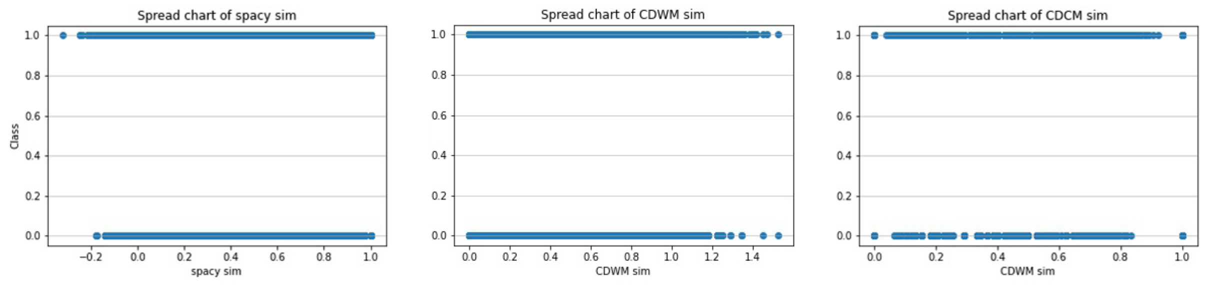


Figure 3.5: spread chart for three features

Due to a potential error with the similarity functions, a false values were calculated and added to Spacy sim column cells and to other also. When we plot their charts we notice that lower values which means that the mention and the candidate are not similar to each ether are in the same row with a value of '1' in the column named 'class'. Instead of that it is supposed to be a value of 0 in that cell. So we did a filtering operation, and saved just the rows that satisfies these conditions:

A cell value in Spacy sim column greater than 0.5 and Cell value in class Columne qual to one.

Cell value in Spacy sim column smaller than 0.5 and Cell value in class Column equal to zero.

we did the same filtering to the other two similarities.

The following figure 3.6 shows how the data in that column become much better, and serves our task with a higher quality.

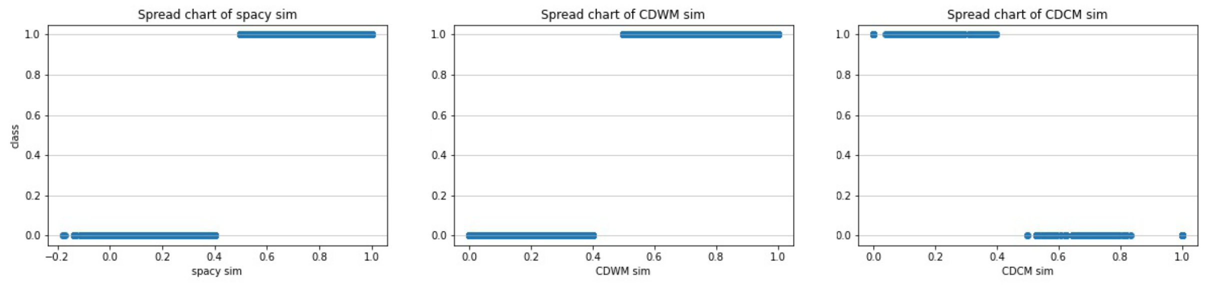


Figure 3.6: spread chart for three features edited

3.4.2 Dataset Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. we started exploring the data by plotting the class count shown on figure 3.7 and understanding it can help us a lot for training an effective model.

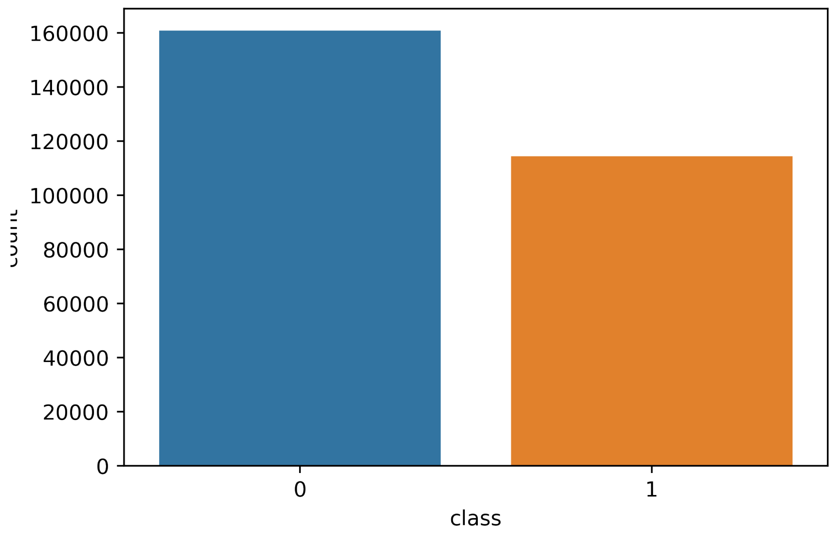


Figure 3.7: class count plot

In this figure 3.7 we can conclude that the data is unbalanced where we can see that 65 % percent are zeros and 35 % are ones. We will deal with this small problem later.

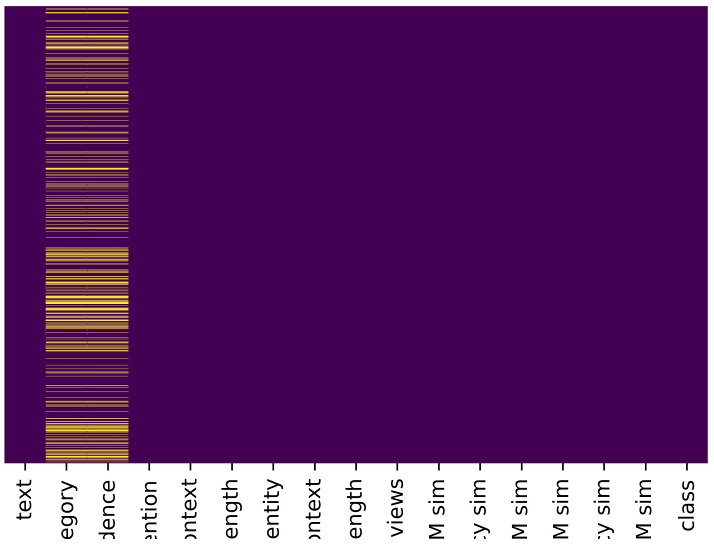


Figure 3.8: null count

From this one, we notice that a lot of rows has a null cells in the category column. On the contrary there are a little of nulls in other columns. To tackle this problem, we fill all the null cells in the category column with the category ‘Other’. So we avoid losing a large number of rows containing a significant information.

From a different angle, we can see a problem. If we change the null cells in category column by ’Other’, it maybe lead to error because of the fact that the category cell was filled by null cause of a connection error or accidental error from Google API. but in real, the text has a category(was Not 'Other'). so we prefer to drop those lines, sacrificing that significant information.

For the other columns we fill them by the mean of each column.the mean is given by the total of the values of the colum divided by the number of rows. the mean is equal to:

$$x = \frac{1}{n}\sum_{1}^{n} x_i \tag{3.2}$$

After completing this operation we plot the chart again, and here it is figure 3.9 the result, no cell is null!

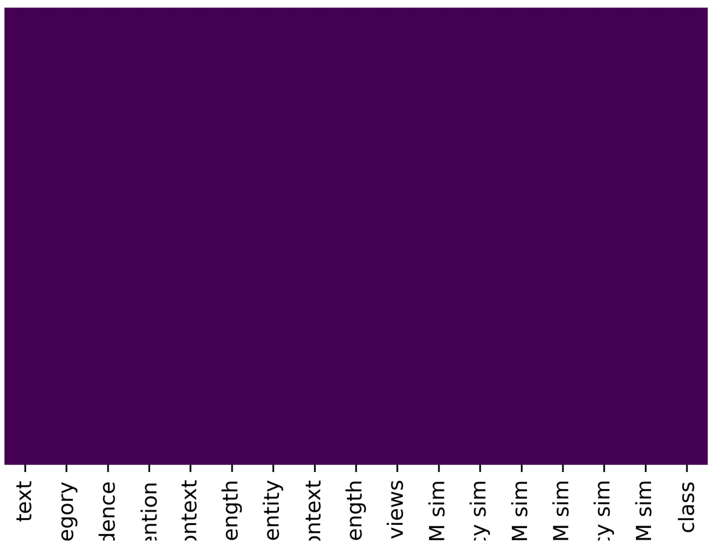


Figure 3.9: null count

3.4.3 Feature Selection

In order to select the features that make our models learn in an efficient way, we did an evaluation test on TD dataset to each feature individually, using the three classifiers LR, DT and RF. Then we select features with an accuracy of more than 80 % percent and drop the others. We don't know if this selecting method do the trick or not, but the results of models trained on those features can prove its effectiveness. The table 3.5 shows 6 features that reach the threshold which are: Category, Mention length, Mention context, Item length, Item context, Spacy sim.

	Logistic Regression	Decision Tree	Random Forest
Category	0.83	0.85	0.86
Confidence	0.69	0.71	0.70
Mention length	0.81	0.83	0.84
Mention context	0.85	0.86	0.86
Item length	0.84	0.84	0.85
Item context	0.86	0.86	0.86
Page views	0.58	0.65	0.67
CDCM sim	0.70	0.72	0.71
Spacy sim	0.80	0.82	0.86
CDWM sim	0.69	0.70	0.71
CTX CDCM sim	0.65	0.69	0.72
CTX spacy sim	0.69	0.70	0.70
CTX CDWM sim	0.68	0.69	0.69

Table 3.5: features evaluation

3.4.4 Training Models

The experiment was carried out using three classifier models, namely: logistic regression, decision tree, Random Forests.

Logistic Regression [10] is a classifier, based on a logistic regression function also called a sigmoid function. The decision trees method [15] classifies data into branch segments including a root node, internal nodes and leaf nodes. Random Forests (RF) [2] classification works by joining several decision trees together to correct over-fitting to the training set of the decision trees.

We have trained our models on a splitted part represent 70 % from the dataset. As we know the trainer accept only numerical values and does not accept string values for that we convert columns that contains values of type string or Boolean or the categorical columns to became numerical values. We utilized for this step the TF-IDF Vectorizer for SKLearn library. The snippet of code below shows how we convert it.

```python
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop words='english', max features=1500)
x1_train_pre_tfidf = vectorizer.fit(x1_train)
x1_train_tfidf = x1_train_pre_tfidf.transform(x1_train)
x1_test_tfidf = x1_train_pre_tfidf.transform(x1_test)
```

Listing 3.6: TF-IDF vectorizer from SKLearn library

We convert each categorical or textual column separately, and concate them together in one pandas DataFrame. After this the data was ready to enter the trainer as input.

For each method we import the class that is responsible to do the work, and then create an instance from that class. The instance has a method called 'fit' that accept data as input. The next few lines we put in your hands the implementation of each one of them.

Using logistic Regression

```python
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(solver='lbfgs',max_iter=2000,penalty='l2', C
    =3.1)
lr_model.fit(pd_x6_train, y_train)
```

Listing 3.7: logistic regression code

Using decision tree

```python
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,
    random_state=0)
clf.fit(train, y_train)
```

Listing 3.8: decision tree code

Using random forest

```python
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, max_depth=None,
    min_samples_split=2, random_state=0)
clf.fit(train, y_train)
```

Listing 3.9: random forest code

3.4.5 Testing and evaluation

We have tested our models on a splitted part represent 30% from the datasets(TD and KWDW_R). And the result of the experiment is presented in this section.

TD dataset convolution matrixes shown in figure 3.10 shows that logistic regression succeed on 89.31 % of predictions it made and fail on 10.69 %. And decision tree classifier succeed on 94.27 % of predictions it made and fail on 5.73 %. And random forest succeed on 92.38 % of predictions it made and fail on 7.63 %.

KWDW_R dataset convolution matrixes shown in figure 3.11 shows that logistic regression succeed on 83.71 % of predictions it made and fail on 16.29 %. And decision tree classifier succeed on 84.7 % of predictions it made and fail on 15.3 %. And random forest succeed on 97.88 % of predictions it made and fail on 2.12 %.

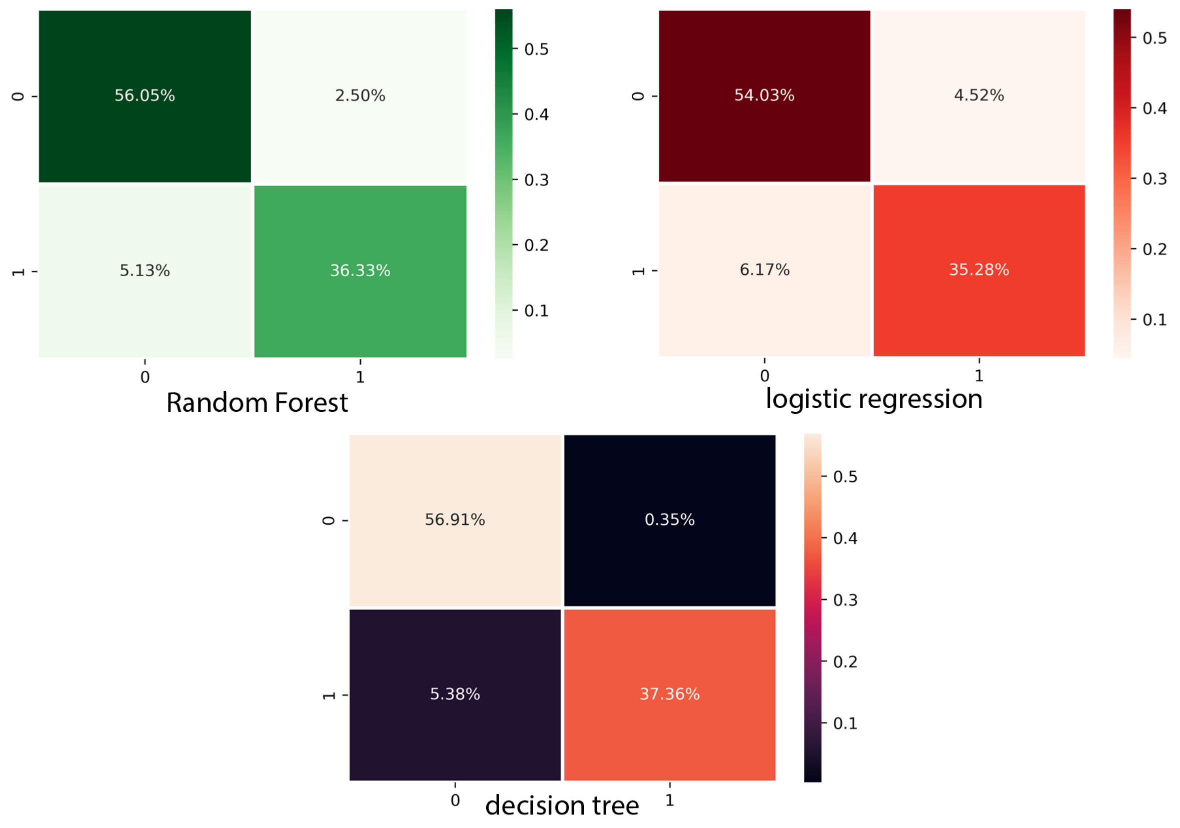


Figure 3.10: TD dataset's convolution matrixes

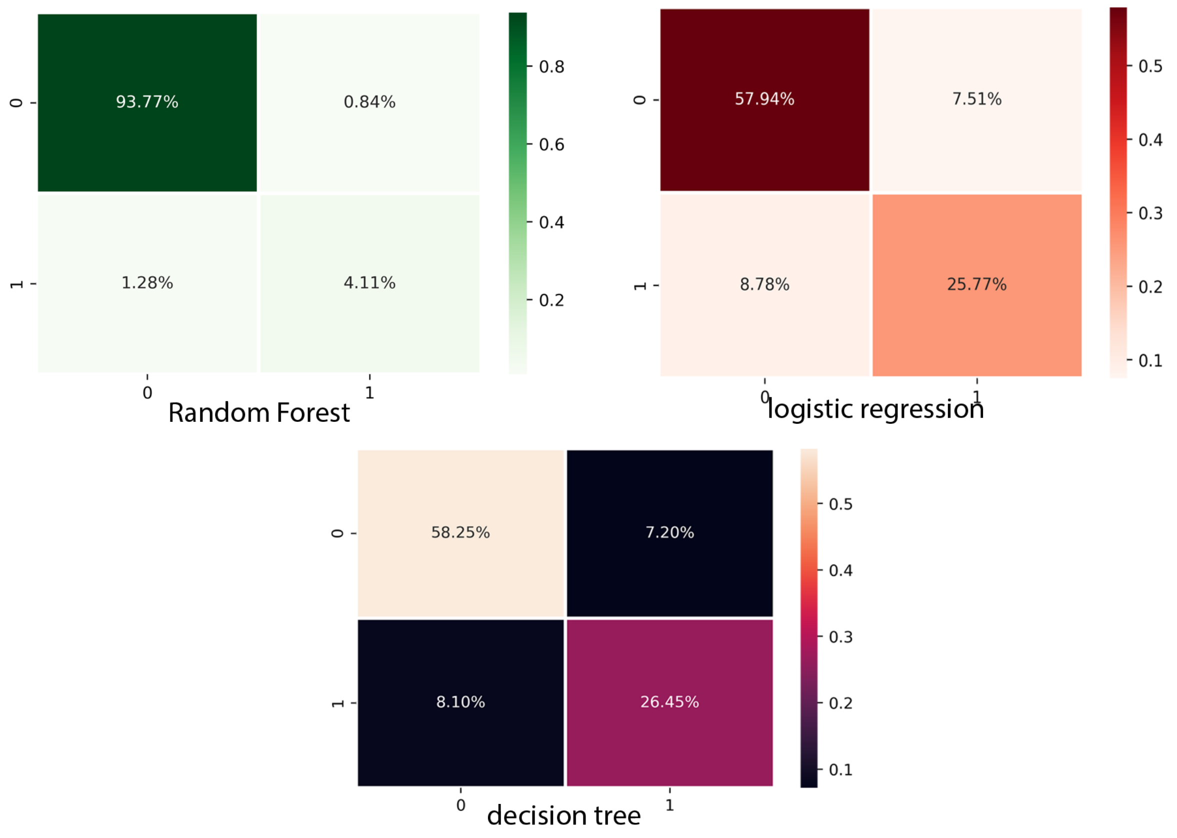


Figure 3.11: KWDW_R dataset's convolution matrixes

From the following tables we can notice that the random forest classifier achieved abetter result than the rest, reaching ninety-two percent on the accuracy metric. The logistic classifier obtained eighty-nine percent on the same metric and ranked last. The tree decision ranked second and rich a score of ninety-one percent on the accuracy measure.

Table 3.6: TD dataset evaluation

(A) Decision Tree

	precision	recall	f1 score	accuracy
0	0.91	0.94	0.93	0.91
1	0.92	0.87	0.89	

(B) Logistic Regression

	precision	recall	f1 score	accuracy
0	0.90	0.92	0.91	0.89
1	0.89	0.85	0.87	

(C) Random Forest

	precision	recall	f1 score	accuracy
0	0.92	0.96	0.94	0.92
1	0.94	0.88	0.91	

Table 3.7: KWDW_R dataset evaluation

(A) Decision Tree

	precision	recall	f1 score	accuracy
0	0.88	0.89	0.88	0.85
1	0.79	0.77	0.78	

(B) Logistic Regression

	precision	recall	f1 score	accuracy
0	0.87	0.89	0.88	0.84
1	0.77	0.75	0.76	

(C) Random Forest

	precision	recall	f1 score	accuracy
0	0.88	0.92	0.90	0.86
1	0.83	0.76	0.80	

3.4.6 improve dataset

Before we close this section, an enhancement is still available to do. Based on the fact that the database is not balanced and it is important thing for the trainer to have a balanced data in the aim of achieving a better result. Returning to the figure 3.7 that shows the number of samples with class 1 and the samples with class 0 in TD dataset.

We extract some rows from KWDW_R dataset that has the value '1' in the 'class' column. And add them to TD dataset, In order to get a new balanced dataset.

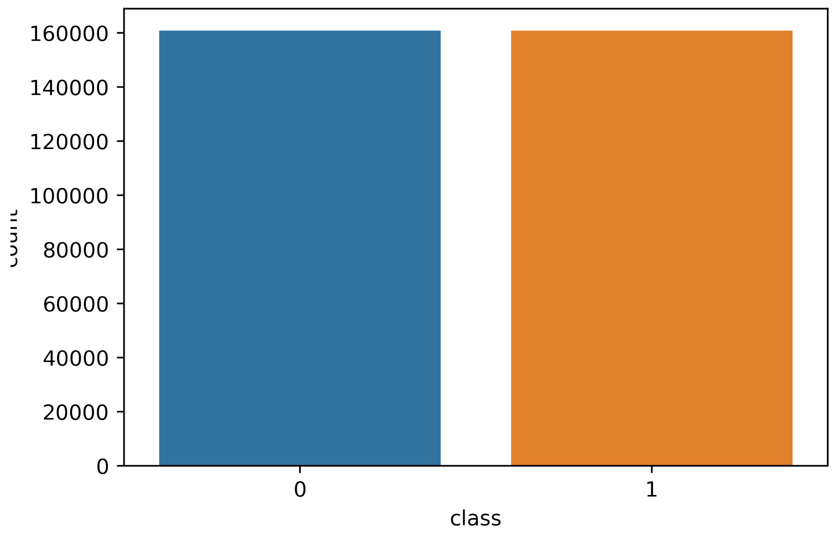


Figure 3.12: class count plot

We did the same experiment on this dataset using the three methods, applying the same function of filtering and cleaning. After this change, we observed an increase of 1 percent for all three method on the accuracy measure. And the tables 3.8 bellow shows the result. It confirms the importance of balancing the dataset.

The convolution matrixes shown in figure 3.13 shows that logistic regression succeed on 91.56 % of predictions it made and fail on 8.44 %. And decision tree classifier succeed on 94.27 % of predictions it made and fail on 5.73 %. And random forest succeed on 93.73 % of predictions it made and fail on 6.27 %.

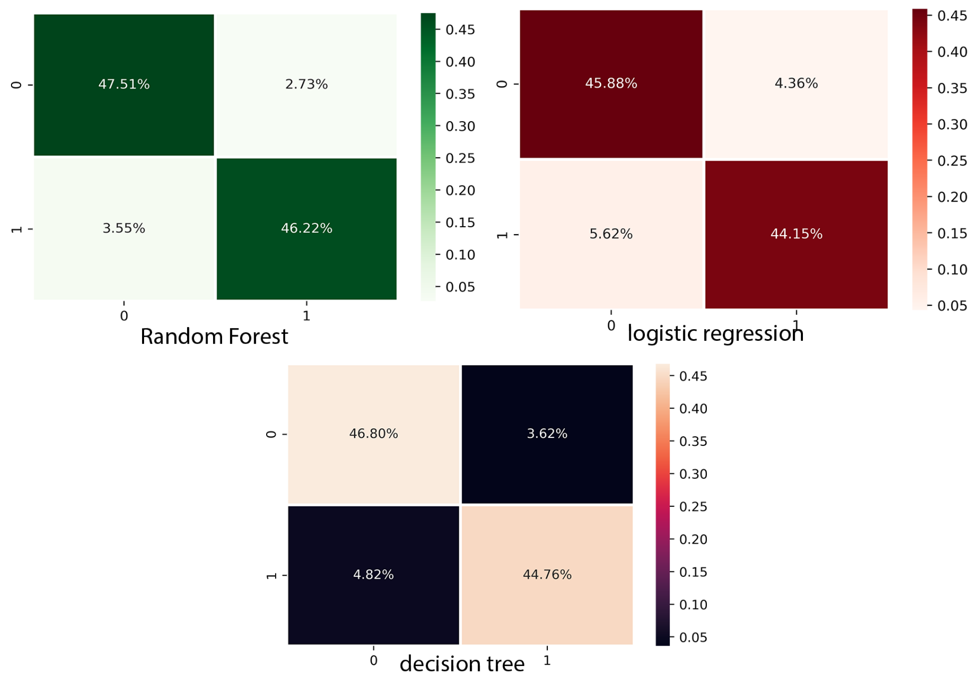


Figure 3.13: TD dataset's convolution matrixes

Table 3.8: balanced dataset evaluation

(A) Decision Tree

	precision	recall	f1 score	accuracy
0	0.91	0.93	0.92	0.92
1	0.93	0.90	0.92	

(B) Logistic Regression

	precision	recall	f1 score	accuracy
0	0.89	0.91	0.90	0.90
1	0.91	0.89	0.90	

(C) Random Forest

	precision	recall	f1 score	accuracy
0	0.93	0.95	0.94	0.94
1	0.94	0.93	0.94	

3.5 Conclusion

In this chapter we have presented all experiments that we tried and tested such as the first system which didn't live up to the hopes. Then we presented our datasets where we applied three machine learning methods on them. For our task we notice that Random Forest method works well and achieve higher scores on different datasets.

General Conclusion

In this thesis, we presented two approaches used to solve named entity linking problem. The first one depends on probabilities and the second depends on machine learning algorithms.

In chapter one, we have the entity linking problem on general and we talked about what is it, and what features are exploited to create linking systems. We particularized a part of the talk about familiar approaches that followed to handle the problem. In chapter two we presented four systems such as Cucerzan, UKP-UBC system. We discussed them and show their perspectives and results. In chapter three we build two searchable alias dictionary to generate referent Wikipedia articles for the given mentions. Each one has its structure, but they're not very different from each other. We generated two datasets based on Wikipedia and KWDW dataset each one related with one dictionary. We utilized these two datasets separately and combined to train our models using logistic regression, decision tree and random forest.

Our models achieve 94 % and 92 % on accuracy metric using the combined dataset when we add rows from KDWD dataset to our dataset that we build depending on Wikipedia article and Wikidata site. Those rows are from class '1', and after adding them the dataset become balanced which made it easy to achieve this result.

As a future work, we plan to apply more machine learning methods on our dataset and may achieve a higher result, furthermore we can dive into deep learning method such RNN. And we thought about adding more feature to the dataset which could constitute a high value feature and make the models more efficient. Because our native language (we're talking about Arabic language) we plan to construct a similar dataset from Arabic Wikipedia Articles and apply those ML methods on them and monitor its effectiveness.

References

[1] Adio Akinwale and Adam Niewiadomski. “Efficient Similarity Measures for Texts Matching”. en. In: (2015), pp. 7–28. URL: https://core.ac.uk/download/pdf/53096864.pdf (visited on 05/13/2021).

[2] L. Breiman. “Random Forests”. en. In: (2001), pp. 5–32. (Visited on 07/15/2021).

[3] Razvan Bunescu and Marius Pasca. “Using Encyclopedic Knowledge for Named Entity Disambiguation”. en. In: (Apr. 2006), pp. 9–16. URL: https://aclanthology.org/E06-1002.pdf (visited on 05/20/2021).

[4] Silviu Cucerzan. “Large-Scale Named Entity Disambiguation Based on Wikipedia Data”. en. In: (June 2007), pp. 708–716. URL: https://aclanthology.org/D07-1074.pdf (visited on 05/20/2021).

[5] Elizabeth D.Liddy. *Natural Language Processing*. 2001.

[6] Mark Dredze et al. “Entity Disambiguation for Knowledge Base Population”. en. In: (Aug. 2010), pp. 277–285. URL: https://aclanthology.org/C10-1032.pdf (visited on 05/06/2021).

[7] Nicolai Erbs et al. “UKP-UBC Entity Linking at TAC-KBP”. en. In: (2012), pp. 1–6. URL: https://www.ixa.eus/sites/default/files/dokumentuak/3450/KBP_UKP_UBC.pdf (visited on 06/05/2021).

[8] Toni Gruetzea et al. “CohEEL: Coherent and Efficient Named Entity Linking through Random Walks”. en. In: (June 2018), pp. 1–25. URL: https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/publications/2016/coheel_jwskg.pdf (visited on 05/10/2021).

[9] Ben Hachey et al. “Evaluating Entity Linking with Wikipedia”. en. In: (May 2012), pp. 1–48. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.1494&rep=rep1&type=pdf (visited on 05/16/2021).

[10] S. L. C. Houwelingen and J. C. Van. “Ridge Estimators in Logistic Regression”. en. In: (1992), pp. 95–108.

[11] Diksha Khurana et al. “Natural Language Processing: State of The Art, Current Trends and Challenges”. en. In: (Aug. 2017), pp. 1–25. URL: https://arxiv.org/ftp/arxiv/papers/1708/1708.05148.pdf (visited on 04/02/2021).

[12] Tie-Yan Liu. “Learning to Rank for Information Retrieval”. en. In: (2009), pp. 225–331. URL: https://shortest.link/12WL (visited on 05/16/2021).

[13] GONGQING WU(IEEE Member), YING HE, and XUEGANG HU. “Entity Linking: An Issue to Extract Corresponding Entity With Knowledge Base”. en. In: (Mar. 2018), pp. 6220–6231. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8246707.

[14] Aasish Pappu, Roi Blanco, and Yashar Mehdad. “Lightweight Multilingual Entity Extraction and Linking”. en. In: (Feb. 2017), pp. 1–10. URL: http://www1.cs.columbia.edu/~kapil/documents/wsdm17nel.pdf (visited on 06/17/2021).

[15] J. R. Quinlan. “Induction of decision trees”. en. In: (1986), pp. 81–106.

[16] E.F. Tjong Kim Sang and F. De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”. en. In: (2003), pp. 142–147. URL: https://link.springer.com/content/pdf/10.1007/978-3-319-11964-9_33.pdf (visited on 07/15/2021).

[17] Mark Schmidt. “Argmax and Max Calculus”. en. In: (Jan. 2016), pp. 1–3. URL: https://www.cs.ubc.ca/~schmidtm/Documents/2016_540_Argmax.pdf (visited on 05/26/2021).

[18] Wei Shena, Jianyong Wang(Senior Member IEEE), and IEEE) Jiawei Han(Fellow. “Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions”. en. In: (May 2014), pp. 1–20. URL: http://www.projectsgoal.com/download_projects/cloud-computing/cloud-computing-projects-GCC00111.pdf (visited on 05/05/2021).

[19] Sonit Singh. “Natural Language Processing for Information Extraction”. en. In: (July 2018), pp. 1–24. URL: https://arxiv.org/pdf/1807.02383.pdf.

[20] Rene Speck and Axel-Cyrille Ngonga Ngomo. “Ensemble Learning for Named Entity Recognition”. en. In: (July 2020), pp. 520–534. URL: https://link.springer.com/content/pdf/10.1007/978-3-319-11964-9_33.pdf (visited on 07/06/2021).

[21] Ricardo Usbeck et al. “AGDISTIS - Graph-Based Disambiguation of Named Entities using Linked Data”. en. In: (Oct. 2014), pp. 1–15. URL: https://link.springer.com/content/pdf/10.1007/978-3-319-11964-9_29.pdf (visited on 07/03/2021).