Big Data Research Project

Andrea Armani
Dimitrios Tsesmelis

Discovering Missing Internal-links in Wikipedia using Semantic-based approaches

prepared at CentraleSupélec

Defended on February 2021

Tutor:	Nacéra Seghouani	-	CentraleSupelec	nacera.seghouani@centralesupelec.fr
Supervisor:	Adnan El Moussawi	-	CentraleSupelec	adnan.el_moussawi@lri.fr

0.1 Acknowledgments

We would like to express our gratitude to Prof Nacéra SEGHOUANI and Mr. Adnan EL MOUSSAWI, the tutor and the supervisor of our project respectively, for providing us with knowledge, help and the required guidance in the research and development of our idea.
As our work is an extension of an outstanding thesis, we would like to thanks his creator Mr. Jose Carlos Badillo. His impressive work and his help throughout the whole process were fundamental for us and, without his support, we would have never been able to carry out this project.

Contents

Chapter 1

Introduction

Wikipedia is the most extensive and most accessible compilation of knowledge existing in our times. The content of Wikipedia is free under the GNU Free Documentation License [1], thus anyone can modify the information of an article without following any review or approval process before displaying the update. Although this makes the site more susceptible to unreliable sources, it is expected that if someone observes something false, biased, or defective, he/she can quickly correct the text so that articles can be improved, becoming more trusted as they get enriched throughout time. Moreover, parallel editions in different languages have been raised to create a global knowledge ecosystem. It is important to underline that articles between languages are not translations. Instead, they are articles with their own particular content that can be compared with the ones in other editions.

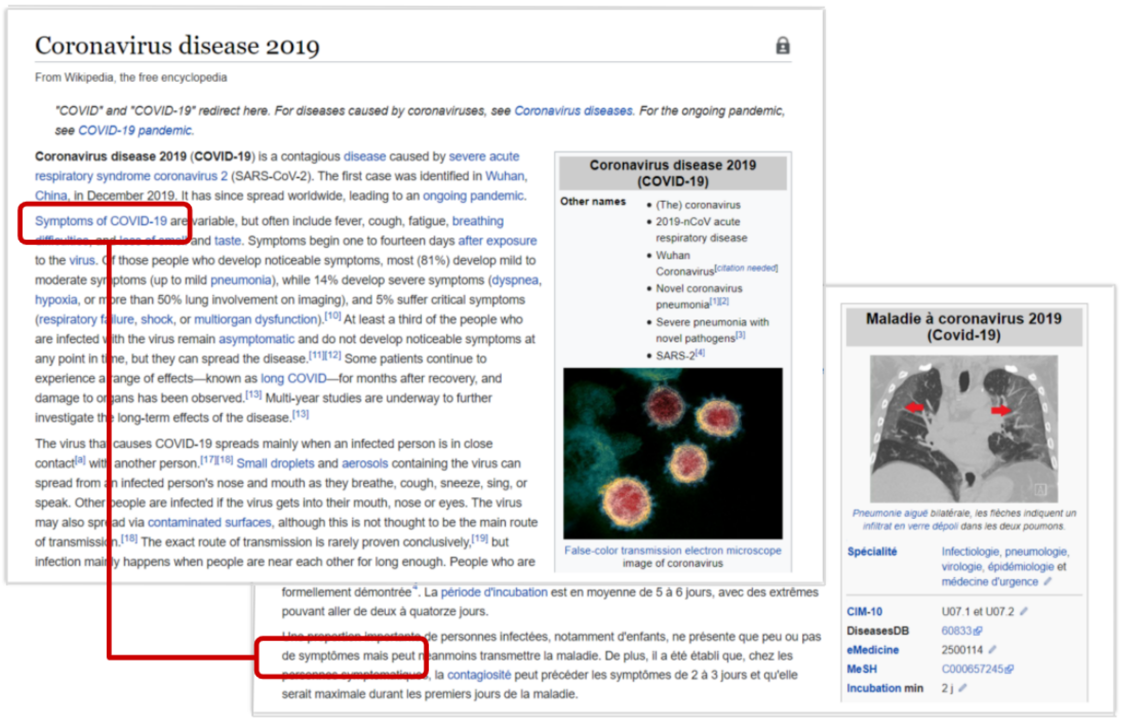
As Wikipedia is arranged as a network of pages interconnected through hyperlinks, it can naturally be modeled by a directed graph, taking the article pages as vertices and the hyperlinks as edges. It is also interesting to note that Wikipedia has proven to be a big data environment with a high volume of articles growing exponentially over the time [Buriol 2006], and keeping a high velocity of changes with a pace of more than 1,200 edits per minute [Wikipulse 2021]. The latter, combined with the fact that hyperlink creation is a human responsibility and the network can evolve without supervision, can lead to omitting relevant links or constant generation of errors that may harm the knowledge discovery and the navigation over the graph. In such an environment, the linking process (creation, edition, and deletion) demands to be supported by recommendation systems that helps with the proper detection and assessment of valuable hyperlinks to mitigate the weaknesses coming from the human factor.

1.1 Objective and Goals

When processing Wikipedia as a graph, we can consider the articles to be vertices and the links to be edges of the graph. Each link can be either an *internal-link* or a *cross-link*. The difference is that internal links are used to relate articles across the same language, while cross-links are used to perform multilingual exploration. Fig. 1.1 is an example of an *internal missing-link*: in the english version of the *Coronavirus disease 2019* webpage there is an internal link to *Symptoms of COVID-19* webpage, which is a useful way to correlate the disease with the symptoms that are originated from it. By exploiting a crossing-link we can move from the english version of *Coronavirus disease 2019* to the corresponding french version, *Maladie à coronavirus 2019*. As it is depicted in Fig. 1.1, the french version of the article skips the internal link to the symptom page.

[1] GNU Free Documentation License `https://www.gnu.org/licenses/fdl-1.3.en.html`

Figure 1.1: Cross-links example



Coronavirus disease 2019

From Wikipedia, the free encyclopedia

"COVID" and "COVID-19" redirect here. For diseases caused by coronaviruses, see Coronavirus diseases. For the ongoing pandemic, see COVID-19 pandemic.

Coronavirus disease 2019 (**COVID-19**) is a contagious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The first case was identified in Wuhan, China, in December 2019. It has since spread worldwide, leading to an ongoing pandemic.

Symptoms of COVID-19 are variable, but often include fever, cough, fatigue, breathing difficulties, and loss of smell and taste. Symptoms begin one to fourteen days after exposure to the virus. Of those people who develop noticeable symptoms, most (81%) develop mild to moderate symptoms (up to mild pneumonia), while 14% develop severe symptoms (dyspnea, hypoxia, or more than 50% lung involvement on imaging), and 5% suffer critical symptoms (respiratory failure, shock, or multiorgan dysfunction).[10] At least a third of the people who are infected with the virus remain asymptomatic and do not develop noticeable symptoms at any point in time, but they can spread the disease.[11][12] Some patients continue to experience a range of effects—known as long COVID—for months after recovery, and damage to organs has been observed.[13] Multi-year studies are underway to further investigate the long-term effects of the disease.[13]

The virus that causes COVID-19 spreads mainly when an infected person is in close contact[a] with another person.[17][18] Small droplets and aerosols containing the virus can spread from an infected person's nose and mouth as they breathe, cough, sneeze, sing, or speak. Other people are infected if the virus gets into their mouth, nose or eyes. The virus may also spread via contaminated surfaces, although this is not thought to be the main route of transmission.[18] The exact route of transmission is rarely proven conclusively,[19] but infection mainly happens when people are near each other for long enough. People who are

Coronavirus disease 2019 (COVID-19)	
Other names	• (The) coronavirus • 2019-nCoV acute respiratory disease • Wuhan Coronavirus[citation needed] • Novel coronavirus pneumonia[1][2] • Severe pneumonia with novel pathogens[3] • SARS-2[4]

False-color transmission electron microscope image of coronavirus

formellement démontrée[4]. La période d'incubation est en moyenne de 5 à 6 jours, avec des extrêmes pouvant aller de deux à quatorze jours.

Une proportion importante de personnes infectées, notamment d'enfants, ne présente que peu ou pas de symptômes mais peut néanmoins transmettre la maladie. De plus, il a été établi que, chez les personnes symptomatiques, la contagiosité peut précéder les symptômes de 2 à 3 jours et qu'elle serait maximale durant les premiers jours de la maladie.

Maladie à coronavirus 2019 (Covid-19)	

Pneumonie aiguë bilatérale, les flèches indiquent un infiltrat en verre dépoli dans les deux poumons.

Spécialité	Infectiologie, pneumologie, virologie, épidémiologie et médecine d'urgence
CIM-10	U07.1 et U07.2
DiseasesDB	60833
eMedicine	2500114
MeSH	C000657245
Incubation min	2 j

In his master thesis [Badillo 2020], the author proposes to exploit the topology of Wikipedia's graph structure with the goal of providing an efficient way to discover missing internal links among Wikipedia articles regarding the same topic, across different languages. For an article, also defined as target, the first step of the proposed algorithm is to provide a list of candidates (Candidate Set). Such candidates are found as internal links of the different language versions of the target. The goal of this step is to find potential suggestions that could improve the hyperlink structure of the aforementioned article. Following this, a second task represents each target article as a *representative vector*. This intermediate operation is necessary to prepare the data for the the third and final step, in which the suggestions are ranked using *Extended Jaccard with Reciprocal Centrality* using Page-Rank as centrality measure. The way in which this approach is implemented, however, contains a bottleneck that centralizes the computation of the representative vectors, which results in a slowdown in performance. Moreover, when the thesis was written, there was no other ranking algorithms to compare the results, hence it was not possible to include an evaluation task based on the comparison of multiple approaches.

The main goal of our work is to extend the aforementioned master's thesis by adding a different way to provide suggestions to enrich a Wikipedia page with internal missing links. To do so, we are leveraging the benefits of a semantic-based approach that aims to find relevant pages, not included yet in the article as internal links, based on the textual information they hold. The computation of the semantic relatedness will be performed by using Semantic-based analysis, meaning that the different Wikipedia articles will be compared based on the text that they include. The information needed as input for the algorithm is in the form of plain text, so it is interesting to try to include a different amount of information and evaluate the outcome of the algorithm. That is why we implemented two different versions:

1. **Title-based analysis**: The title of the articles are used as a similarity measure;
2. **Abstract-based analysis**: The abstract part of the articles is used as similarity measure.

Once the results are provided, we further extended our work with a section to evaluate them considering what we achieved in the two different versions. The aim of this part of the work is to collect the different suggestions obtained along with their weights and finally produce an overall ranking that benefits from the different approaches. As in the future it is reasonable to think that further approaches will be developed, we propose a framework that can adapt to different inputs.

1.2 Contributions

The contributions with this work serve fundamentally to the purpose of adding new methods of proposing missing internal links to Wikipedia articles. Specifically, it exploits both the structure of Wikipedia as a graph as well as the semantics of the corpus of its articles to provide a meaningful and more performant way of ranking the candidates. In addition, it combines several recommendation systems to create a weighted voting recommendation system that is able to suggest missing internal links, with higher accuracy.

The rest of this work is structured as follows. Chapter 2 details the proposed architecture to perform the candidate ranking using semantic-based approach and a rigorous methodology to compare rankings provided by different algorithms. Chapter 3 elaborates on details about the used dataset as well as the resources that were used to perform our experiments. Chapter 4 presents several aspects of our experiments, the results of the algorithms implemented by this work and a comparison with different approaches. Finally, Chapter 5 concludes this work and discusses future aspects of this research.

1.3 Related Works

Several text-based approaches have been proposed over the years that enable the comparison of text corpus. These techniques are well known and placed under the umbrella of Natural Language Processing field (NLP). However, in our work we need to leverage of the specific content of Wikipedia dataset, which consists of many articles, coming from different fields and varying from science to encyclopedic articles. Gabrilovich and Markovitch [Evgeniy Gabrilovich 2007] proposed Wikipedia-based Explicit Semantic Analysis (ESA), which is a way to compute semantic similarity of natural language text by representing the meaning of the text to a high-dimensional vector space of concepts. In this work, we are using this technique to efficiently rank a set of articles (Candidate Set) exploiting the semantics of their text. As we previously explained, the Candidate Set results from the work of Badillo [Badillo 2020] by using the structure of Wikipedia as a graph and by combining the available information in the different editions of Wikipedia.

Chapter 2

Discovering missing internal-links algorithms

2.1 Explicit Semantic Analysis

As we already discussed in the previous chapter, the current method of detecting missing internal links in Wikipedia is a complex problem. The work of [Badillo 2020] splits the process in three different steps, namely *fetching the data* using the available Wikipedia dumps, *creating a Candidate Set* containing many potential articles that can be missing links for the target article and finally *ranking these candidates* in order to show the articles that are highly likely missing links. The goal of our work is to compute the Candidate Ranking in a different way, rather that exploiting the structure of the graph.

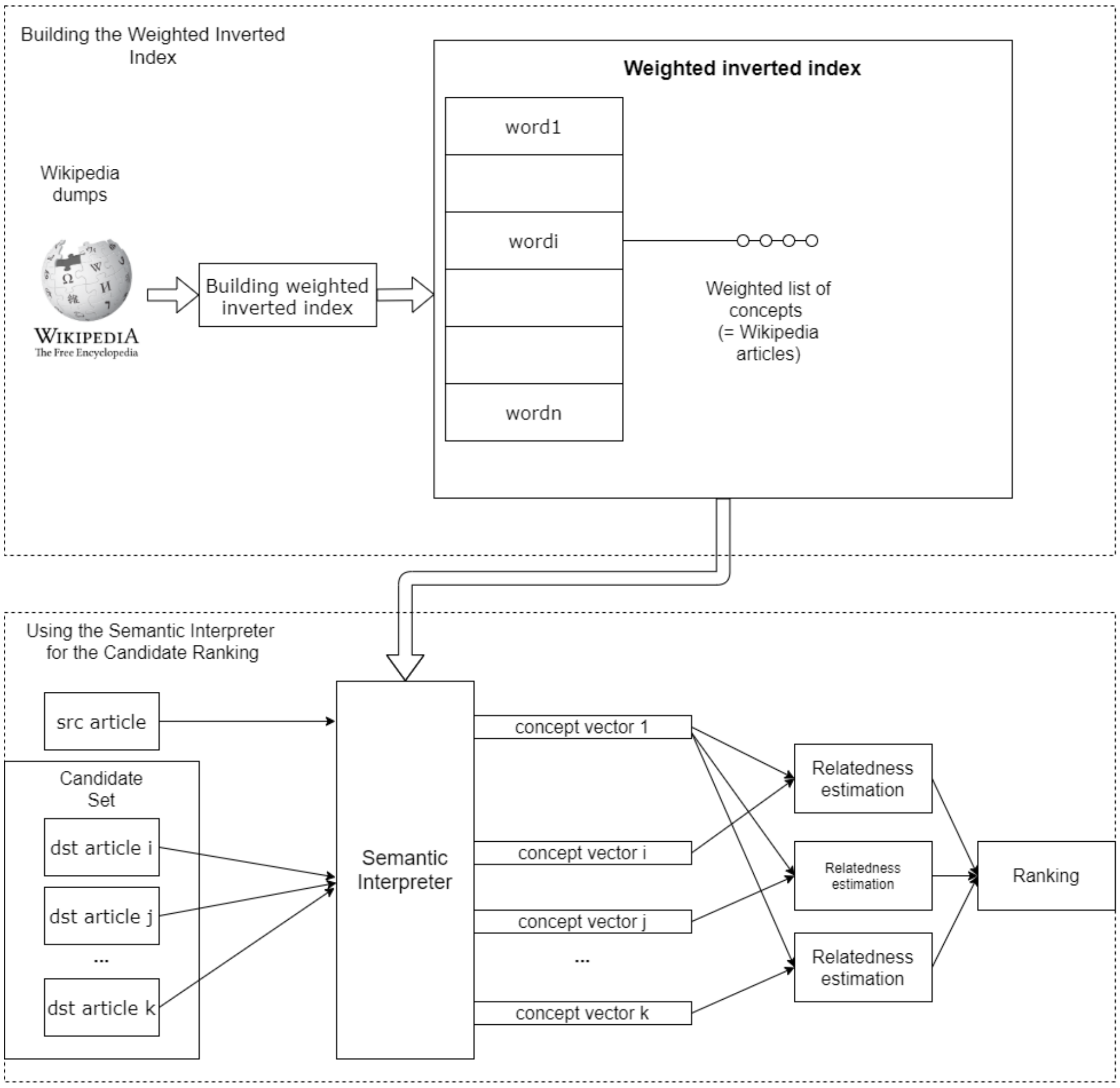
Wikipedia is known to be a good source of information for people that are seeking answers for different questions. These questions can vary and they strongly depend on the individual field of study. For instance, someone may look for information related to art, another one may search Wikipedia for scientific purposes and another one for encyclopedic reasons. Although these reasons are very different, they all share one common element: their answers will be in the **form of text**. This characteristic inspired us to develop a corpus-based, and more specifically a semantic-based approach, that will assist on ranking the candidate articles. In our approach, this process consists of the following two steps:

1. **Building the Weighted Inverted Index:** The first step is to build a **Weighted inverted index** [Evgeniy Gabrilovich 2007] that will be later used to map the Wikipedia articles from the text space to the vector space. It is important to note that this process has a high, **one time** calculation cost, as it will be created in the beginning of the program once, and then it will be used in all the subsequent executions. To build such an index, the English Wikipedia dumps [1], containing all the articles of Wikipedia, are used.

2. **Candidate Ranking:** Such an index can be used to create a Semantic Interpreter, the main goal of which is to map the input texts to the vector space. More specifically, from the previously computed Candidate Set, that contains several articles that are potential candidate links for the target article, we need to define a similarity measure based on the text of the articles. By providing the articles to the Semantic Interpreter, we will end up with vectors (one per article) that represent the articles

[1]**enwiki-YYYYMMDD-pages-articles-multistream.xml.bz2** file available at `https://dumps.wikimedia.org/enwiki/YYYYMMDD/`, where YYYYMMDD should be replaced by the date of the desired Wikipedia snapshot

Figure 2.1: Proposed Architecture Diagram of the Candidate Ranking



in the vector space. Finally, the ranking can be performed by estimating the relatedness between the vector of the input article and the rest of the vectors (e.g. by using cosine similarity).

With our approach, we are able to tackle the bottleneck of the Candidate Ranking in [Badillo 2020] as the computation of the vectors can be performed much faster. It is clear from our results in Chapter 4 that our process of ranking the candidates, produced by the previous process, outperforms the one exploiting the structure of the graph, as its calculation cost varies from seconds to few minutes, depending on the used method. The same process in this work [Badillo 2020] took more than 15 hours! The previously described process is illustrated in Figure 2.1

2.1.1 Building the Weighted Inverted Index

As described in [Evgeniy Gabrilovich 2007], to speed up the Semantic Interpretation, we first need to build an Inverted Index that maps each concept of Wikipedia to a number of words. In our work, which consists an extension of the Wikipedia-based Explicit Semantic Analysis (ESA) [2], Apache Lucene library [3] has been used in two steps. In the first step, the full text of each Wikipedia article is used and each word is mapped to a term-to-document index, where document is meant to be a concept. In the second step, the previously created index is inverted by mapping each concept to all the terms that are important for that concept.

2.1.2 Using the Semantic Interpreter

Once the inverted index is built, it is possible to exploit the Semantic Interpreter. To do so, following [Evgeniy Gabrilovich 2007], the input text is represented with a vector using the TFIDF scheme. Then, the Semantic Interpreter iterates over each text word, retrieving corresponding entries (concepts) from the inverted index. Finally, it merges them into a weighted vector of concepts that represents the given text in the vector space. In order to compute the similarity of two input texts, we calculate the dot product between the two vectors. Consequently, the more concepts two vectors share, the more similar they will be.

2.1.3 Ranking Methods

As already mentioned in a previous chapter, the current methods of proposing internal missing links in Wikipedia lack of a proper evaluation method. Hence, the more recommendation systems exist, the more likely it is to propose common articles as missing links. With our approach, we propose two different semantic-based recommendation systems.

1. In our first solution, we propose a system that uses the previously explained Semantic Interpreter that receives as input the **titles** of the articles to perform the Candidate Ranking. The Algorithm 1 summarizes the process of performing the Candidate Ranking using the titles of the articles.

Algorithm 1 Candidate Ranking using article titles

```
Result: A list containing the ranked candidate articles
$t1 \leftarrow$ Extract the target title from the input file
for each candidate article do
    $t2 \leftarrow$ Extract the article title from the input file
    $ct1 \leftarrow$ Compute the concept vector of t1
    $ct2 \leftarrow$ Compute the concept vector of t2
    $score \leftarrow$ Compute the dot product of ct1 and ct2
    Save the computed score to the output file
end
```

[2]ESA source code https://github.com/pvoosten/explicit-semantic-analysis
[3]Apache Lucene https://lucene.apache.org/

2. In our second solution, we propose a system that uses the Semantic Interpreter and the **abstract** part of the articles to perform the ranking. To extract the abstract of an article, we use the MediaWiki [4], which is an API provided by Wikipedia to easily extract knowledge from it. With the provided structure of the URL [5], it is possible to extract the abstract of a Wikipedia article using its API. The Algorithm 2 summarizes the process of performing the Candidate Ranking using the abstracts of the articles.

Algorithm 2 Candidate Ranking using article abstracts

```
Result: A list containing the ranked candidate articles
$t1 \leftarrow$ Extract the target title from the input file
$a1 \leftarrow$ Call Algorithm 3 with t1 as input
for each candidate article do
    $t2 \leftarrow$ Extract the article title from the input file
    $a2 \leftarrow$ Call Algorithm 3 with t2 as input
    $ca1 \leftarrow$ Compute the concept vector of a1
    $ca2 \leftarrow$ Compute the concept vector of a2
    $score \leftarrow$ Compute the dot product of ca1 and ca2
    Save the computed score to the output file
end
```

Algorithm 3 Extract the Wikipedia article's title using MediaWiki

```
Input: Target title
Output: The abstract of the input Wikipedia article
Form the required URL to query the API
Get the result from the API (in .json)
Extract the field corresponding to the abstract
```

2.2 Weighted Voting Recommendation System

As mentioned in the introductory chapter, we further extended [Badillo 2020] with the addition of a task to compare the results obtained by different ranking algorithms. As we consider likely and desirable to have different approaches to rank candidates, we developed a framework capable of ingesting an arbitrary number of rankings and find, among all the suggestions, which articles are the most relevant.
In order to be able to asses the order of the suggested articles, each ranking should not only contain all the suggestions, but also a vector representing their importance weights. In case such vector is not provided, each suggestion of an algorithm will have the same weight. Below is a formal definition of the problem. Let:

- K be the number of ranking algorithms;

[4]MediaWiki API https://www.mediawiki.org/wiki/MediaWiki
[5]Coronavirus disease 2019's abstract https://en.wikipedia.org/w/api.php?action=query&format=json&prop=extracts&titles=Coronavirus_disease_2019&redirects=1&exintro=1&explaintext=1

- M be the number of ranked pages for each ranking $\{m_i | m >= 0, i <= K\}$
- R be the set of K rankings $\{r_i | r \in R, i <= K\}$
- For every $r_i \in R$ let:
  - C be the list of candidates
  - W be the list of weights associated to C

However, as the weights are obtained as results of different algorithms, they could potentially have different scales as well, hence it would not be possible to compare the algorithms as is. To tackle this problem, we scaled the results of each algorithm using the *MinMax Scaling* in order to obtain a new vector w', in which every weight is a continuous value between the range [0,1]:

$$\{\forall c_{ij} \exists w'_{ij} | c_i \in C \land C \in R \land 0 \leqslant i \leqslant K \land w'_{ij} \in [0,1] \land 0 \leqslant j \leqslant M_i\} \tag{2.1}$$

Now all the weights have an homogeneous scale, so it is finally possible to compare them for every suggestion.
Let:

- X be the set of the possible suggestions from all the articles.

Each $x \in X$ is an article associated to an array that contains up to K scores. The final step of the evaluation is, for every suggestion $x \in X$, to calculate the *Overall Ranking Score* ORS_x as such:

$$ORS_x = AVG(x)\mathrm{e}^{\frac{k_x}{K}} \tag{2.2}$$

The mathematical formulation sets the bounds of the formula between [0,e^1]. It aims to represent two different aspects: in the first part of the formula, for every suggestion we keep the average of the results provided by the algorithms. However, this is not sufficient: rather than using the simple average, we tried to value more the suggestions that were produced by several algorithms. That is why, in the second operator, instead of applying a linear growth we exploited the exponential. In such a way, we implicitly represented the idea that suggestions received many times, possibly without a great average, are more important than the ones that obtained a good score over few instances. However, with a small number of algorithms applied, the exponential does not significantly influence the results and the contribution of the second part of the formula is quasi-linear. Nevertheless, as some new approaches will be developed in the future, the leverage effect of this term will be shown.

Algorithm 4 Weighted Voting Recommendation System

Result: An ordered list containing the best suggestions from a set of suggestion arrays

```
$s1 \leftarrow$ Set of suggestion arrays
$S2 \leftarrow$ Empty set
$K \leftarrow$ number of suggestion arrays
for each array in $s1$ do
    $C \leftarrow$ Articles in the array
    $W \leftarrow$ Weights associated to $C$
    $W' \leftarrow$ MinMax scaling of $W$
    $S2 \leftarrow$ Append to S2 the candidate C along with W'
end
$X \leftarrow$ Set of articles in S2
$res \leftarrow$ Empty Set
for each article $x$ in $X$ do
    $avg_x \leftarrow$ average score of W' for article x
    $k_x \leftarrow$ occurrences of x
    $ORS \leftarrow avg_x \mathrm{e}^{\frac{k_x}{K}}$
    $res \leftarrow$ Append x and its ORS to res
end
Return $res$
```

Chapter 3

Resources & Performance

3.1 Wikipedia dataset

Wikipedia freely distributes its data to the public in several ways. For the purposes of this work, the database dump files is the best performing way to process the Wikipedia's data. They can be downloaded from the Wikipedia Foundation website [1]. Their repository contains the most recent snapshots of the data with a variety of formats like SQL, JSON, or XML for each one of the language editions. To construct the graph data structure relevant for this study as well as the inverted index for the semantic interpreter, it is enough to download the files listed in Table 3.1. The database schema can be found in MediaWiki documentation [2] and it is the easiest way to understand the attributes included in these files.

File Name	**Content**	**Size**
multistream.xml.bz2	Full text of each page (article)	19 GB
page.sql.gz	Base information per page	1.7 GB
redirect.sql.gz	Redirection list	129.4 MB
category.sql.gz	Category information	24.9 MB
categorylinks.sql.gz	Category membership links records	2.6 GB
pagelinks.sql.gz	Page-to-page link records	6.3 GB
langlinks.sql.gz	Interlanguage link records	366.3 MB

Table 3.1: List of dump files.

In particular, for this work the dumps of English and French editions where downloaded.

3.2 Resources

Taking into account the volume of the data and the type of processing that had to be done, it is clear that a personal computer would not be enough to efficiently execute the jobs. More specifically, the snapshot of 01/12/2020 of Wikipedia (only the used files) consists of around 30GB of zipped data and we are processing such data by exploiting the structure of the graph to compute the Candidate Set.
We decided to migrate our experiments to the cloud in order to have the required computational power. More specifically, we used a big enough Virtual Machine (VM), the

[1] Wikipedia dumps `https://dumps.wikimedia.org/backup-index.html`

[2] `https://www.mediawiki.org/w/index.php?title=Manual:Database_layout/diagram&action=render`

power of which is equal to that of a cluster, deployed in Amazon Web Services (AWS). The detailed specification of the VMs can be found in Table 3.2. We experimented with several types of machines in order to find the ideal one for the generation of the Candidate Set, which is a task that requires a big amount of RAM. The results of our experiments are presented in the Section 4.2.

AWS VMs List			
VM Name	Memory (GB)	# VCPUs	SSD Disk (GB)
VM16-128	128	16	200
VM32-256	256	32	200
VM64-512	512	64	200

Table 3.2: AWS VMs.

Chapter 4

Experiments & Results

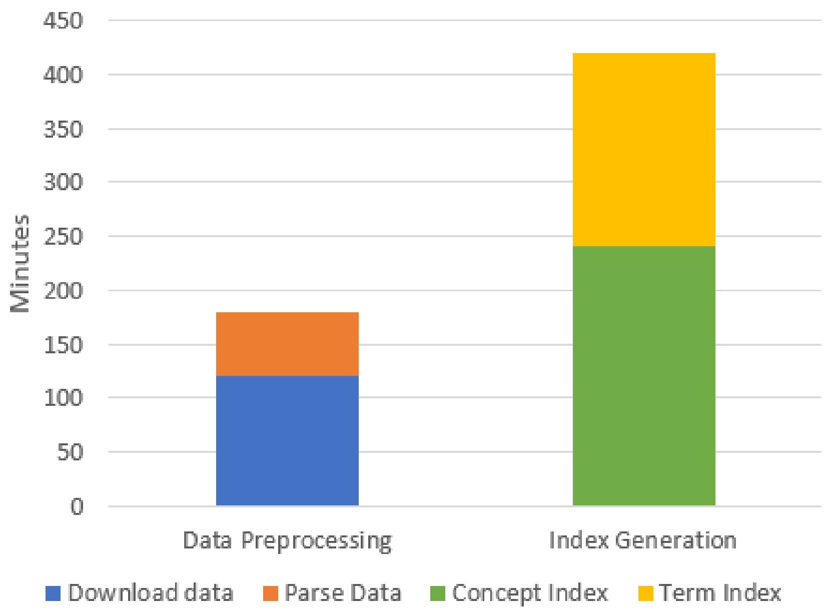
This chapter focuses on reporting the results of the experiments that we performed. In Section 4.1 we present the performance of several jobs, in terms of calculation cost. In Section 4.2 we illustrate several target articles along with the internal-missing link recommendations that we obtain with our solution.

4.1 Performance

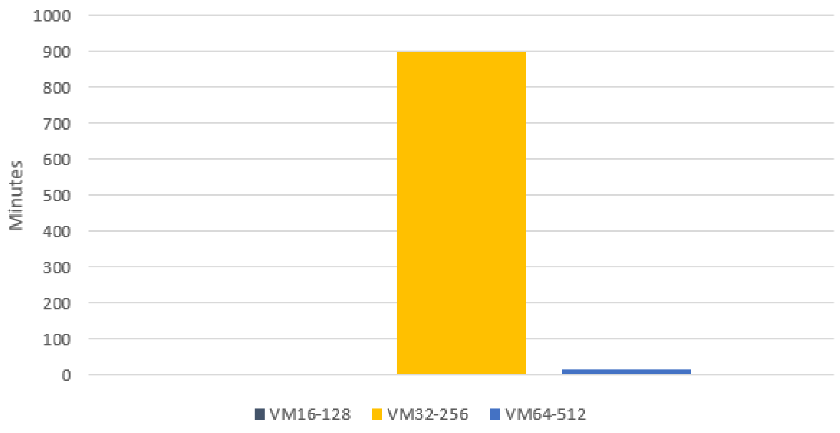
In this section we are going to investigate the performance of several jobs. Before starting the real missing links detection on Wikipedia, we first need to execute several preprocessing jobs that are listed below:

1. Data Preprocessing Jobs.
   - Download data and zip them to bzip2 format.
   - Parse data to be loaded as graph in GraphX.
2. Index Generation.
   - Term Index Generation.
   - Concept Index Generation.

The execution of the previous jobs has one time cost, meaning that we execute them once and we do not need to run them again for our experiments. We need to mention here that between the Data Preprocessing and the Index Generation jobs there are no dependencies, and hence such jobs can be **run in parallel**. Figure 4.1a depicts the performance of all the preprocessing jobs.



(a) Execution times of preprocessing (1 time executed) jobs



(b) Candidate set execution times on different Virtual Machines

Figure 4.1: Measured Performance

As we see in Figure 4.1a, the execution time is high, especially for the generation of the index. We remind here that this index is created by using the whole body of the text of each Wikipedia article and hence the volume is very high. For this job, the **enwiki-20201201-pages-articles-multistream.xml.bz2** file is used the total size of which is equal to 19GB (zipped) and 75GB (unzipped).

Moving on to the jobs that are related with the analysis on the Wikipedia pages, we can sum them up to the following list:

1. Analysis Jobs.
    - Candidate Set Generation.
    - Candidates Ranking.

In Figure 4.1b we can see the performance of Candidate Set Generation on different VMs hosted in AWS. Such a job has very high memory requirements, which make it impossible to run in machine with less than 256GB of RAM (e.g. VM16-128 with 128GB of RAM cannot run this task). Although VM32-256 was able to successfully execute the Candidate Set generation, we can see from the chart that the execution time was 15 hours long, which is too high. To tackle this problem, we decided to switch to a bigger VM, namely VM64-512, the capabilities of which allowed us to run this task in only 15 minutes!

Regarding the Candidates Ranking, Figure 4.2 shows that ranking the candidates using the **Semantic Interpreter** and a part of the text of the articles as input, is very time efficient. More specifically, we can see that the execution time when using the titles of the articles requires only **10 seconds** while when using the abstracts of the articles, it takes around **23 minutes**. These execution times can be further improved if we parallelize the computation of the vectors. Such improvements are discussed in Section 5.2. From these results, it is clear that our approach outperforms the method of Candidate Ranking of [Badillo 2020], where the required time for the same volume of data is around **15 hours**.

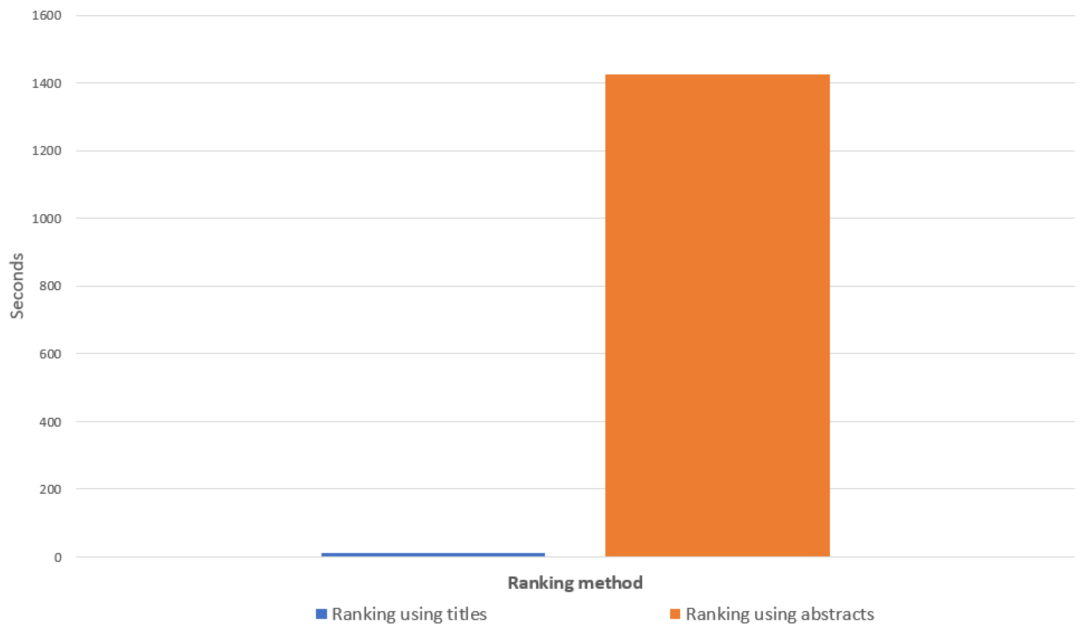


Figure 4.2: Candidate ranking execution time using different methods

4.2 Results

In this chapter we are presenting the results of our experiments. Firstly, we try to explain how exactly ESA works, by showing the concept vectors of a specific sample article, namely *Coronavirus Disease 2019*. Moreover, in the next section we are comparing the results obtained from our work and from [Badillo 2020] thesis. Finally, we illustrate the capabilities of our recommendation system by applying it to several target articles and by interpreting the produced results.

4.2.1 Concept Vector Example

In a previous chapter we explained that ESA works by mapping the given articles from the text to the vector space using Wikipedia concepts. Table 4.1 shows which concepts are related to the page *Coronavirus Disease 2019*. Each Wikipedia concept is represented as an attribute vector of words that occur in the corresponding article. Entries of these vectors are assigned weights using TFIDF scheme. Moreover, every page has a weight associated which quantifies the strength of association between the concept and the words.

Title	Concept	Weight
Coronavirus Disease 2019	Severe acute respiratory ...	0.36845145
	Betacoronavirus	0.069145076
	Metapneumovirus	0.016899912
	Pasteur Institute	0.015240104
	Ageusia	0.013256758
	Crimean–Congo hemorrhagic fever	0.012987377
	Infection control	0.01211222
	Lassa fever	0.011693488
	Parovirus	0.011373511
	International Committee on ...	0.010847493
	...	...

Table 4.1: Concept Vector Example

It is also possible to further inspect the results. For instance, let's observe the relation between the concept *Betacoronavirus* and *Coronavirus Disease 2019* in Table 4.2.

Betacoronavirus concepts	Coronavirus Disease 2019 concepts
Murine Coronavirus	Coronavidae
Embecovirus	Coronavirus Act 2020
Human coronavirus HKU1	Human coronavirus HKU1
List of virus genera	Murine Coronavirus
1889–1890 pandemic	COVID-19 pandemic in Antarctica
Human coronavirus 229E	Wikipedia's response to the COVID-19 pandemic
Coronavirus diseases	DA2PPC vaccine
MERS coronavirus EMC	Human coronavirus 229E
Coronaviridae	Coronavirus diseases
...	...

Table 4.2: Concepts related to pages Betacoronavirus and Coronavirus Disease 2019

We can see that the two articles share the some concepts that are marked by the same colors. It also worths to mention that all the concepts that are linked to the page are easily identifiable as related to the topic.

4.2.2 Results Comparison

As the intention of the work is to extend what was previously done in the master's thesis [Badillo 2020], it necessarily needs to compare the results that the Semantic Interpreter achieves with the ones obtained with the previous methodology. Table 4.3 serves this goal by analyzing the suggestions for *Coronavirus Disease 2019*. For the sake of comparison and only for this target article, both the methodologies used the same dump of Wikipedia, namely snapshot of 2020/04/01. For the rest of the experiments, the snapshot of 2020/12/01 was used. The first column of the results shows the top 10 candidate articles, coming from our approach (Semantic Interpreter) and using the titles of the articles, the second column shows the results using the abstracts of the articles and the third column the top 5 candidates from the master's thesis (Topological Research):

Top 10 using **Titles**	Top 10 using **Abstracts**	Top 5 **Topological Research**
Severe acute respiratory syndrome-related coronavirus	Hand foot and mouth disease	Betacoronavirus
Betacoronavirus	Nipah virus infection	Severe acute respiratory syndrome
Metapneumovirus	Chickenpox	Viral hemorrhagic fever
Pasteur Institute	Rift Valley fever	Bornholm disease
Ageusia	West Nile fever	Indiana vesiculovirus
Crimean–Congo hemorrhagic fever	Measles	...
Infection control	Yellow fever	...
Lassa fever	Lassa fever	...
Parovirus	Monkeypox	...
International Committee on Taxonomy of Viruses	Crimean–Congo hemorrhagic fever	...

Table 4.3: Ranking comparison between different approaches for Coronavirus Disease 2019 article

It is evident how all methods returns meaningful suggestions. Moreover, two of the methods were able to suggest the same page (*Betacoronavirus*) in the top 10 recommendation while the third (abstracts) was able to rank it in the 75th position, which is remarkable given the diversity of the approaches applied.

4.2.3 Additional Experiments

In this sections we present several results, obtained from our work. More specifically, we are trying to propose internal missing links for Wikipedia articles that belong to different categories, namely art, science, industry and history.

At first glance, we can clearly see from the following results that in general, proposing missing internal links using ESA with abstracts seems to provide more accurate recommendation, as the the weights associated with the proposed articles are higher in most of the cases. At this point, we remind that the weight shows how similar is the source with the destination articles, meaning that the higher the weight, the more similar the articles are. Although the use of titles seems to be more sensitive and less accurate, we will see later that this is not always true, as in some cases, the simplest solution (titles) seems to "tell the truth", even if it seems to be wrong at first glance!
We should not forget that there is a trade-off between using ESA with abstracts and titles, as the first one tends to be more accurate but significantly slower.

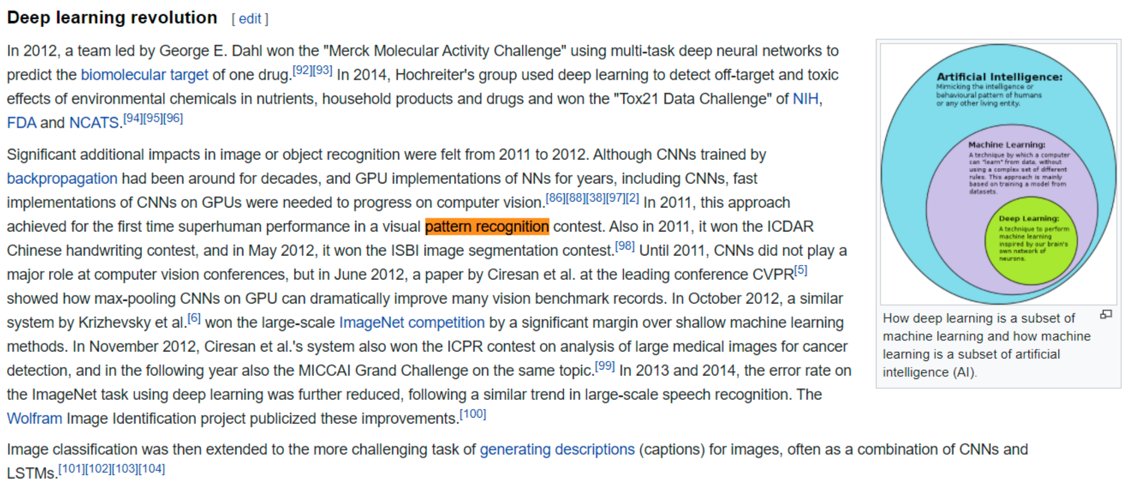
- Internal Link Recommendations for ***Deep Learning***: Table 4.4 shows the top 10 missing links for the target article *Deep Learning*. By observing the results, we can see that both methods propose 6 same articles in the top 10 recommendations, some of them with very high weight. Out of the 6 common articles, there are 3 that refer to well-known Deep Learning frameworks, namely *PyTorch*, *TensorFlow* and *Keras*. These suggestions are very meaningful and they should be included in Wikipedia's Deep Learning article as internal links.
  Extreme learning machine [1] is the first article, suggested by both methods. It seems that it is a quite meaningful result as it refers to the topic of Feedforward neural networks, which is basically a meaning that encloses Deep Learning, and hence it should be related to it.
  Another interesting recommended article from ESA with abstracts is *Pattern recognition*. As we see in Fig. 4.3, this key word exists many times in the corpus of *Deep Learning* article [2] but a link to the article [3] is missing.

[1]Extreme learning machine `https://en.wikipedia.org/wiki/Extreme_learning_machine`
[2]Deep Learning `https://en.wikipedia.org/wiki/Deep_learning`
[3]Pattern recognition `https://en.wikipedia.org/wiki/Pattern_recognition`

Top 10 using **Titles**	Weight	Top 10 using **Abstracts**	Weight
Extreme learning machine	0.23	Extreme learning machine	0.60
PyTorch	0.19	Neural Designer	0.50
TensorFlow	0.14	Pattern recognition	0.40
Keras	0.13	TensorFlow	0.27
Learning	0.12	Keras	0.24
Apache_MXNet	0.12	Emergent algorithm	0.23
Open Neural Network Exchange	0.11	OpenNN	0.18
Theano(software)	0.11	PyTorch	0.15
Neural Designer	0.09	Microsoft Cognitive Toolkit	0.14
OpenNN	0.08	Text mining	0.14

Table 4.4: Internal missing links for Deep Learning

Figure 4.3: Detected missing links for *Deep Learning* article



Deep learning revolution [edit]

In 2012, a team led by George E. Dahl won the "Merck Molecular Activity Challenge" using multi-task deep neural networks to predict the biomolecular target of one drug.[92][93] In 2014, Hochreiter's group used deep learning to detect off-target and toxic effects of environmental chemicals in nutrients, household products and drugs and won the "Tox21 Data Challenge" of NIH, FDA and NCATS.[94][95][96]

Significant additional impacts in image or object recognition were felt from 2011 to 2012. Although CNNs trained by backpropagation had been around for decades, and GPU implementations of NNs for years, including CNNs, fast implementations of CNNs on GPUs were needed to progress on computer vision.[86][88][38][97][2] In 2011, this approach achieved for the first time superhuman performance in a visual pattern recognition contest. Also in 2011, it won the ICDAR Chinese handwriting contest, and in May 2012, it won the ISBI image segmentation contest.[98] Until 2011, CNNs did not play a major role at computer vision conferences, but in June 2012, a paper by Ciresan et al. at the leading conference CVPR[5] showed how max-pooling CNNs on GPU can dramatically improve many vision benchmark records. In October 2012, a similar system by Krizhevsky et al.[6] won the large-scale ImageNet competition by a significant margin over shallow machine learning methods. In November 2012, Ciresan et al.'s system also won the ICPR contest on analysis of large medical images for cancer detection, and in the following year also the MICCAI Grand Challenge on the same topic.[99] In 2013 and 2014, the error rate on the ImageNet task using deep learning was further reduced, following a similar trend in large-scale speech recognition. The Wolfram Image Identification project publicized these improvements.[100]

Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.

How deep learning is a subset of machine learning and how machine learning is a subset of artificial intelligence (AI).

Image classification was then extended to the more challenging task of generating descriptions (captions) for images, often as a combination of CNNs and LSTMs.[101][102][103][104]

- <u>Internal Link Recommendations for ***Elon Musk***</u>: Table 4.5 shows the top 10 missing links for the target article *Elon Musk*. By observing the results, we can confidently conclude that the scores are not biased by the topic of a candidate article. As our approach is corpus-based, it could face the risk that pages related to a similar topic could be very similar. If this was the case, the suggestions we would obtain would end up being limited to few topics. For some articles, for instance *Deep Learning*, it is very hard to tell if this is the case, as the topic itself is very specific. That is the reason why we also included *Elon Musk* as a target page: mister Musk has several businesses and interests in fields that are completely different one to another. What we obtained are suggestions that vary from TV Series to Automotive, and from topic related from Mars to weapons. Thus, we can confidently conclude that the field of an article does not bias ESA ranking.

Top 10 using **Titles**	Weight	Top 10 using **Abstracts**	Weight
Colonization of Mars	0.026	Electric car	0.0626
Rocket	0.010	Launch vehicle	0.0560
AltaVista	0.026	Colonization of Mars	0.0420
Lethal autonomous weapon	0.0095	Rocket	0.0115
Electric car	0.0077	Mars (2016 TV series)	0.0108
Mars (2016 TV series)	0.0075	Lethal autonomous weapon	0.0085
Open letter	0.0073	The Planetary Society	0.0080
Flamethrower	0.0054	California	0.0079
...	...	Dianne Feinstein	0.0070
...	...	Sport utility vehicle	0.0054

Table 4.5: Internal missing links for Elon Musk

- Internal Link Recommendations for ***French Revolution***: Table 4.6 shows the top 10 missing links for the target article *French Revolution*. Both approaches show that *Trial of Louis XVI* should be included as an internal link, which is safe to say that is correct. As for the rest of the suggestions provided, most of them are about people that lived or facts that happened during the French Revolution. Some historical figures are, for instance, *Louis Marie Turreau*, a french general, or *Martial Herman*, a judge during the Reign of Terror. As for the facts, some examples are the *Federalist revolts* or the *Proclamation of the abolition of the monarchy*.
  What is interesting here is to note that all these suggestions are about people and facts that rarely are remembered outside France. However, as the french version of Wikipedia dumps is used, the english page will benefits from this as it will be able to include more fine-grained details that, although being very relevant, are usually forgotten by most of the editors.

Top 10 using **Titles**	Weight	Top 10 using **Abstracts**	Weight
Mona Ozouf	0.08739	Federalist revolts	0.58
Louis Marie Turreau	0.05707	Demonstration of 20 June 1792	0.51
Michel Vovelle	0.0531	Revolt of 1 Prairial Year III	0.48
Trial of Louis XVI	0.0528	Muscadin	0.40
Prairial	0.0424	First Restoration	0.39
Albert Mathiez	0.0396	Trial of Louis XVI	0.39
Proclamation of the abolition of the monarchy	0.03588	Martial Herman	0.36
Claude Basire	0.0346	Le Vieux Cordelier	0.33
Christophe Antoine Merlin	0.0333	Biens nationaux	0.32
Revolutionary sections of Paris	0.0275	Maurice Duplay	0.29

Table 4.6: Internal missing links for French Revolution

- Internal Link Recommendations for ***Peugeot***: Table 4.7 shows the top 10 missing links for the target article *Peugeot*. From the first column, we have removed the first 52 recommendations because they have **Weight** = 1. All the 52 removed articles have similar titles: *Peugeot Type Number*, where Number is always different for each line. By observing such behavior, someone can state that using ESA with titles is sensitive to titles that share many common words (in this case "Peugeot" and "Peugeot Type 2" share the word "Peugeot") as ESA first splits the input text into word using TFIDF scheme and then it relates each word with some concepts. Hence, if two input texts have many words in common, the result will be very close to the maximum value (1). However, by observing the results given by the ESA with abstracts, we see that the top 10 articles include the ones that we removed before. In this case, we cannot state that ESA has the same problem as before, as the input text is much longer (hence less common words) and the **Weight** <> 1. Consequently, this is a proof that in many cases, the naivest solution is probably the best (or very close to it).

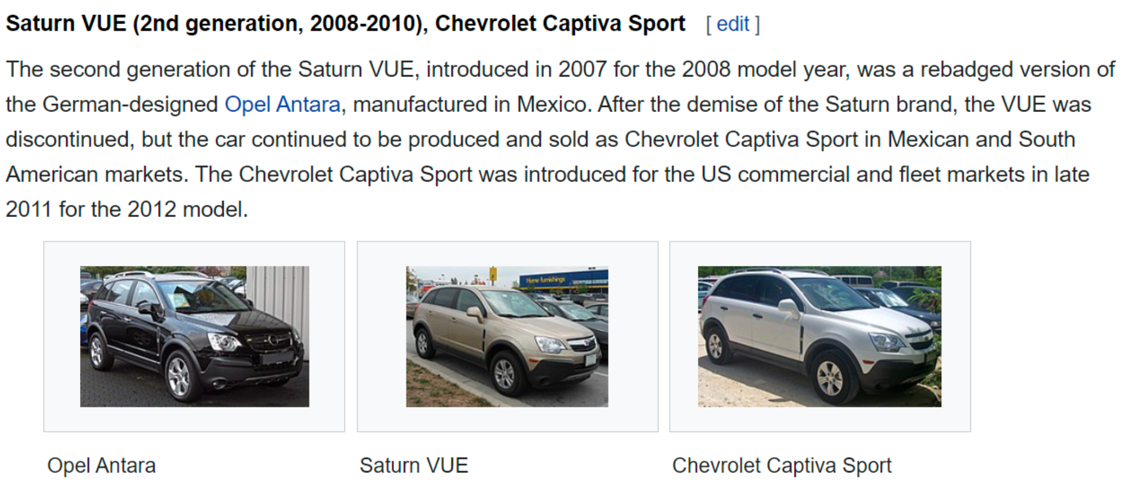
  Moving on to the next articles, there are several recommendations that are related to PSA Group. PSA Group is a multinational manufacturer of automobiles and motorcycles that owns several well-known brands, like Peugeot, Citroën, DS, Opel and others. Inspired by these links, in the next example, we experimented with one of these brands, namely *Opel*, to find out whether our system will recommend some common articles for the two brands that are part of the PSA Group.

Top 10 using **Titles**	Weight	Top 10 using **Abstracts**	Weight
Peugeot Quadrilette	0.83	Peugeot Type 159	0.62
Guangzhou Peugeot Automobile Company	0.72	Peugeot Type 2	0.61
Lion-Peugeot Type VA	0.71	Peugeot Type 56	0.61
PSA HDi engine	0.14	Peugeot Type 6	0.60
Mandeure	0.052	Peugeot Type 5	0.60
PSA Mangualde Plant	0.051	Peugeot Type 14	0.59
PSA Kenitra plant	0.050	Peugeot Type 105	0.59
2010 Paris Motor Show	0.037	Peugeot Type 16	0.57
2014 Paris Motor Show	0.037	Peugeot Type 9	0.57
2012 Paris Motor Show	0.037	Peugeot Type 10	0.56

Table 4.7: Internal missing links for Peugeot

- Internal Link Recommendations for ***Opel***: Table 4.8 shows the top 10 missing links for the target article *Opel*. It seems that there are not recommended articles that are common for *Opel* and *Peugeot*. However, the obtained results are quite interesting, as we see that *Opel Vivaro* is a missing link, proposed by both methods and with high weight. *Opel Vivaro* refers to a light commercial vehicle, manufactured by Opel. It is highly likely that this link is a real missing link, as *Opel* article includes many models of the company but not *Opel Vivaro*. Moreover, other models are proposed by ESA with abstracts, namely *Saturn Vue*, *Chevrolet Captiva* and *Suzuki Splash*,

Figure 4.4: Detected missing links for *Opel* article



Saturn VUE (2nd generation, 2008-2010), Chevrolet Captiva Sport [edit]

The second generation of the Saturn VUE, introduced in 2007 for the 2008 model year, was a rebadged version of the German-designed Opel Antara, manufactured in Mexico. After the demise of the Saturn brand, the VUE was discontinued, but the car continued to be produced and sold as Chevrolet Captiva Sport in Mexican and South American markets. The Chevrolet Captiva Sport was introduced for the US commercial and fleet markets in late 2011 for the 2012 model.

Opel Antara — Saturn VUE — Chevrolet Captiva Sport

with **Weight** > 0.20. As we see in Fig. 4.4, these recommendations are meaningful, as these models are included in *Opel* article [4] but the links to them are missing.

Top 10 using **Titles**	Weight	Top 10 using **Abstracts**	Weight
Opel Vivaro	0.64	Opel Vivaro	0.58
Astra (company)	0.050	Saturn Vue	0.33
Dudenhofen	0.033	Chevrolet Captiva	0.22
Saturn Vue	0.025	Suzuki Splash	0.21
Chevrolet Captiva	0.016	Astra (company)	0.12
Chevrolet Venture	0.014	Chevrolet Venture	0.11
Sandouville Renault Factory	0.013	Carlos Tavares	0.07
Suzuki Splash	0.012	Mary Barra	0.06
Car body style	0.011	Chevrolet Chevelle	0.04
Cerizay	0.008	Suzuki	0.031

Table 4.8: Internal missing links for Opel

4.2.4 Voting Recommendation System Example

Table 4.9 shows the ranking provided by ORS calculation. Following the algorithm, in column *W_title* and *W_abstract* the weights are scaled. Then, based on these scaled data, ORS is calculated. It is interesting to note that *Extreme learning machine* is first suggestion, like both the articles suggested. Moreover, *Pattern recognition* is the only article selected that does not have two score bigger than zero. That is because the number of algorithm K applied is not big, hence the formula cannot fully benefits from the exponential part. However, in the future more algorithms will be provided and K will increase,

[4]Opel `https://en.wikipedia.org/wiki/Opel`

hence the exponential will influentiate more the ORS score. Finally, it is interesting to note that suggestion that were in the top-10 of the algorithms, like *Learning* or *Theano (Software)*, don't appear in the ORS ranking, as their weights were small enough to not be finally considered.

Top 10 using Voting System	W_title	W_abstract	ORS_Weight
Extreme learning machine	1	1	2.71828
Neural Designer	0.38122	0.84366	1.66477
PyTorch	0.83625	0.28842	1.52869
TensorFlow	0.58853	0.44192	1.40053
Keras	0.56892	0.39316	1.30761
Pattern recognition	0	0.67649	1.11535
OpenNN	0.36679	0.36376	0.99293
Apache MXNet	0.50468	0.16379	0.90856
Open Neural Network Exchange	0.45923	0.17499	0.86201
Microsoft Cognitive Toolkit	0.33926	0.25008	0.80011
...	...	...	...

Table 4.9: Weighted Voting Recommendations for *DeepLearning*

Chapter 5

Conclusion & Perspectives

5.1 Conclusion

In this work, we studied the missing internal-links detection problem. We exploited the Wikipedia dataset, structured as a graph as well as semantic-based techniques in order to discover and propose missing internal links for several Wikipedia articles.

Our work constitutes a continuation of Badillo's master thesis [Badillo 2020]. Both works together provide a framework that can be summed up in the following three tasks:

1. Collecting, storing and parsing the current snapshot of Wikipedia into a graph.
2. Suggest a candidate set that can improve a target article's hyperlink structure.
3. Rank the generated candidate to provide the topK most relevant missing links. For this task, several parts of the candidate articles have been used, namely titles and abstracts, in order to define a similarity measure between two articles.

The algorithm developed in this work can be applied to any article of the English version of the online encyclopedia and can be used with the latest Wikipedia dumps.

In conclusion, with our work we managed to propose another way of proposing missing links, additional to the one proposed by Badillo. We can safely state that our algorithm is better, in terms of performance, and the obtained results are comparable.

5.2 Future Work

As a future task, the execution time of the candidate ranking using the abstracts of the articles can be further improved by introducing parallelization. More specifically, multi threading techniques can be applied to speed up the performance of the loop of Algorithm 2. In this way, supposedly that the candidate set contains N articles and the used computer has M cores/processors, the execution time can be reduced up to N/M times.

Although our work contributes to the existing one by proposing a new technique to detect missing internal links in Wikipedia and hence, the results are meant to be more accurate, still there are ways to further validate the quality of the obtained results. One method could be by running our program on an **older snapshot** of Wikipedia (e.g. snapshot of April 2020) and then checking which of the top ranked candidates were added as real internal links to a **newer snapshot** (e.g. snapshot of January 2021). Such an approach seems to be effective as Wikipedia is a dynamic environment, guided by humans, and hence, continuously improved.

We proposed two versions of the ESA algorithm, either based on the title of a webpage or on the abstract obtained from it. It is easy to understand that the more information we provide to the algorithm, the more powerful the representation of a webpage in terms of similarity to the original. We could extend this idea up to taking the full-text inside it, however there are two potential harms: the first problem is that more information comes with a cost in terms of computation. The second one is that adding too detailed information could mislead the ESA algorithm. In other words, this future work is about solving two different tradeoffs: one between the volume of information and the computation it requires and another one between the volume of information and the potential loss of meaningfulness it entails.

Finally, a last future work could be done to improve the suggestions that our approaches provide: the Top-10 ranking proposed by the abstract in Table 4.7 has inside only *PeugeotType* cars. A person that would like to modify *Peugeot* webpage, maybe not an expert in the different car models, would not have a benefit from the proposed suggestions. The problem is that such suggestions are very similar one another, so what can be done is to develop a similarity matrix for the pages. Finally, when the ranking is being created it should also check the similarity score to decide wether or not a suggestion should be added, based on the similarity of the previous selected suggestions and the importance assessed by the ESA algorithm.

Appendix A

Appendix

A.1 Resources

A.1.1 Microsoft Azure

We initially tried to deploy our program to a Virtual Machine (VM) hosted in Microsoft Azure by using the Student Free Tier with a total budget of $100. The specifications of the used VM can be found in Table A.1. By using these resources we realized that we were not able to run the process of generating the Candidate Set, as 16GB of RAM was not enough (*java.lang.OutOfMemoryError: Java heap space*). However, after having deployed our code to the VM on the cloud, we had ready all the required scripts to easily re-deploy our code to another machine. Thanks to our tutor, Mrs. Nacéra Seghouani, we were granted with a budget of $600 to use in Amazon Web Services (AWS).

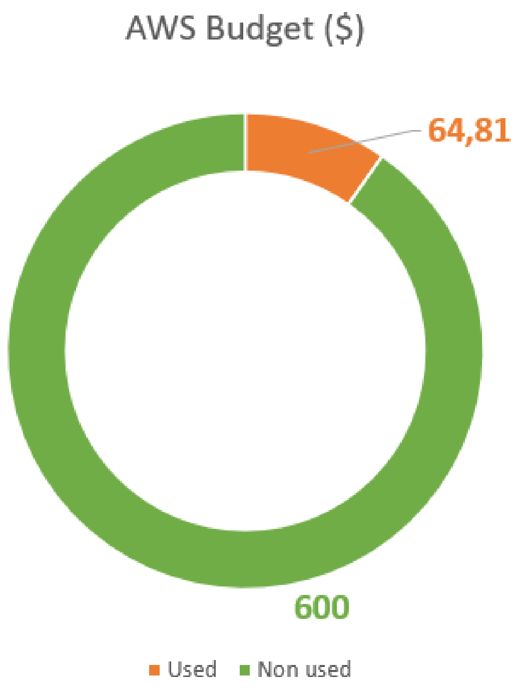
Microsoft Azure VMs List			
VM Name	# VCPUs	Memory (GB)	Disk (GB) - Type
Standard_D2s_v3	2	8	500 - P20 Disks
Standard_D4s_v3	4	16	500 - P20 Disks

Table A.1: Microsoft Azure VMs.

A.1.2 Amazon Web Services Cost

Figure A.1 illustrates the total cost of our executions in AWS. As we see, we managed to keep a low cost of executions, spending less than 11% of the available budget, namely 600$.

Figure A.1: Current AWS Cost



A.2 Source Code

The source code of this work can be found in our GitHub repository [1]. The README.txt file includes guidelines on how to execute the program.

[1]Source Code Repository `https://github.com/JimTsesm/Wikipedia-Internal-Missing-Links-Detection`

Bibliography

[Badillo 2020] Jose Carlos Badillo. Discovering missing internal-links in wikipediausing corpus-based approaches. Master's thesis, CentraleSupélec, CentraleSupélec, 3 Rue Joliot Curie, 91190 Gif-sur-Yvette, France, 2020. (Cited on pages 2, 3, 5, 6, 8, 15, 17 and 25.)

[Buriol 2006] Luciana Buriol, Carlos Castillo, Debora Donato, Stefano Leonardi and Stefano Millozzi. Temporal Analysis of the Wikigraph. 12 2006. (Cited on page 1.)

[Evgeniy Gabrilovich 2007] Shaul Markovitch Evgeniy Gabrilovich. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pages 1606–1611. International Joint Conferences on Artificial Intelligence Organization, 2007. (Cited on pages 3, 5 and 7.)

[Wikipulse 2021] Wikipulse. Edits per minute to wikipedia. vol. [Online; accessed 02-February-2021], 2021. (Cited on page 1.)