

---

# Statistical Models of Semantics using Structured Topics

---

**Statistisches Modellieren von Semantik mit Strukturierten Topics**

Master-Thesis von Simon Dif aus Grenoble, Frankreich

Tag der Einreichung:

1. Gutachten: Dr. Alexander Panchenko
2. Gutachten: Prof. Dr. Chris Biemann



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Fachbereich Informatik  
Language Technology Group

Statistical Models of Semantics using Structured Topics  
Statistisches Modellieren von Semantik mit Strukturierten Topics

Vorgelegte Master-Thesis von Simon Dif aus Grenoble, Frankreich

1. Gutachten: Dr. Alexander Panchenko
2. Gutachten: Prof. Dr. Chris Biemann

Tag der Einreichung:

---

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 20. Juli 2016

---

(S. Dif)

---

---

# Contents

<b>I. Introduction and related work</b>	<b>7</b>
1. Introduction	8
2. Related work	9
<b>II. Background and resources</b>	<b>11</b>
3. Introduction	12
4. Disambiguated Distributional Thesaurus (DDT)	13
4.1. Construction of the thesaurus . . . . .	13
4.2. IS-A relationships . . . . .	17
4.3. DDTs used in our experiments . . . . .	18
<b>III. Extraction of the structured topics</b>	<b>19</b>
5. Presentation of the pipeline	20
6. Preprocessing of the DDTs	21
7. Graph clustering of the DDTs	24
7.1. Introduction . . . . .	24
7.2. Markov Chain Clustering . . . . .	24
7.3. Louvain Method . . . . .	25
7.4. Chinese Whispers . . . . .	26
7.5. Results of the clustering . . . . .	27
7.5.1. Conclusion . . . . .	31
8. Topic annotation with hypernyms	32
8.1. Hypernyms from WordNet . . . . .	32
8.1.1. Searching only the first hypernym in the hierarchy . . . . .	34
8.1.2. Searching all hypernyms in the hierarchy and assigning weights . . . . .	34
8.1.3. Evaluation of WordNet hypernym annotation methods . . . . .	34
8.2. Hypernyms from IS-A relation databases . . . . .	35
8.2.1. Counting hypernyms with and without considering weights . . . . .	35
8.2.2. Adapting TF-IDF scheme to discard noisy hypernyms . . . . .	36



8.2.3. Evaluation of methods to find relevant hypernyms from the IS-A database . 36

**IV. Intrinsic evaluation of the structured topics 37**

**9. Introduction 38**

**10. Interpretability of topics 39**

10.1. Experimental settings . . . . . 39  
10.2. Discussion of results . . . . . 39

**11. Hypernym graph analysis 43**

11.1. Experimental settings . . . . . 43  
11.2. Discussion of results . . . . . 44

**12. Mapping to BabelNet topics 48**

12.1. Experimental settings . . . . . 48  
12.2. Discussion of results . . . . . 50

**13. Conclusion 53**

**V. Application of the structured topics to text categorization 55**

**14. Visualization of topics 56**

14.1. Introduction . . . . . 56  
14.2. Nodes positions . . . . . 56  
    14.2.1. Positioning algorithm . . . . . 57  
    14.2.2. Storing positions . . . . . 57  
14.3. Images . . . . . 57  
    14.3.1. Images sources . . . . . 57  
    14.3.2. Images size . . . . . 58

**15. Topics exploration 59**

15.1. General presentation . . . . . 59  
15.2. Sorting topics by quality . . . . . 59

**16. Text categorization 61**

16.1. Front-end . . . . . 61  
16.2. Back-end: a search engine . . . . . 61  
    16.2.1. Introduction . . . . . 61  
    16.2.2. nDCG . . . . . 63  
    16.2.3. Baseline results with naive random ranking . . . . . 63  
    16.2.4. Simple TF-IDF query . . . . . 64  
    16.2.5. Exact term matching . . . . . 67  
    16.2.6. Combining both methods . . . . . 69



---

16.3. Conclusion . . . . .	72
----------------------------	----

**VI. Conclusion and future work** **73**

<b>17. Conclusion</b>	<b>74</b>
-----------------------	-----------

<b>18. Future work</b>	<b>75</b>
------------------------	-----------

---

# Abstract

Topic modeling is a field of computer science which aims to extract topics out of a document or a corpus of documents. With the exponentially growing World Wide Web and all the literature it contains, topic models are meant to help to relevantly classify these texts and facilitate their exploration.

In this thesis we explore a new methodology to build topics that is different from the conventional unsupervised topic models based such as LDA. We base our approach on a Disambiguated Distributional Thesaurus computed in an unsupervised fashion from texts corpora, on which we apply a clustering method. We explore several parameters for our system and compare the resulting topic models. Additionally, we present a user interface to visualize topics and interact with them.

---

# Acknowledgements

Here I first want to thank Alexander, for advising me all along this thesis, answering all my questions and reviewing my work.

I also thank Chris for the previous advices he gave me, always full of sense.

Finally, I will thank my friends and everybody who supported me during the time of the thesis!



---

# **Part I.**

## **Introduction and related work**

---

---

# 1 Introduction

During the past decades, we have witnessed the emergence of the World Wide Web. Its size has grown exponentially since, reaching an astonishing 50 billion webpages today.<sup>1</sup> In the same time, a lot of text resources have found their place on the web, including global literature with Google Books or scientific papers.

Today, the amount of text information available on the web is such that reading only a small part of it would already take more than a lifetime. The need has emerged for ways to automatically extract information out of texts, in order to save user's time.

Topic models aim to address this issue. They try to automatically find the topics of a text. This allows people to grasp the gist of a text without having to read it. This way, they can read only texts that are relevant with respect to what they are looking for and discard others.

Topic models have been the focus of a lot of work in the last years. Two categories can be found, that answer the following questions:

- How to build topics?
- How to find topics in a text given this text and a set of topics?

Also, several formats of topics have been proposed. A topic can be a distribution of terms of a vocabulary, like in [Blei et al., 2003]. In this case, each topic contains every word of a vocabulary and these words get different probabilities and are sorted differently in each topic. Alternatively, a topic can also be a bag of words. In this case, a topic contains only a division of the whole vocabulary, but on the other hand, the words are not necessarily sorted.

In this master thesis, we explore a new way to build topics. Beyond words, we actually work with word senses: for instance, the word *jaguar* could belong to both a topic about animals and another about cars. Our fully unsupervised system is based on a Disambiguated Distributional Thesaurus, on which we apply a clustering method. We explore several variations in our system parameters and obtain a set of topic models that we compare by conducting three different experiments. We also elaborate methods to annotate topics with hypernyms, that we find either in WordNet [Miller et al., 1990] or in a automatically built IS-A relationships database.

Finally, we introduce a user interface allowing to explore and interact with topics, including their visualization as graphs.

---

<sup>1</sup> <http://worldwidewebsize.com>

---

## 2 Related work

Several directions of work are explored in the field of NLP focusing on topics and plenty of topic models have been proposed. Among them, one of the most famous is the Latent Dirichlet Allocation model (LDA) [Blei et al., 2003]. The general idea behind LDA is the following. Given a text for which we want to find the latent topics, LDA assumes that this text was created following a specific generative model, which contains several parameters. Given this hypothesis, LDA tries to estimate the parameters that were used during the text generation. The topics present in the text are found during this parameter estimation, since one parameter is precisely the distribution over all topics that was used to generate the texts. More precisely, the parameters for the generative model are the number of words in the text and a multinomial distribution over all topics (a Dirichlet distribution, hence the name of LDA). LDA then assumes that each word in the text was generated by first picking a topic, according to the topics distribution and then a word in this topic. In LDA, a topic is defined as a distribution of words over the whole vocabulary. Each vocabulary word has a given probability in each topic. The word is picked in the topic according to the vocabulary words distribution.

LDA therefore presupposes a set of topics, that are distributions over the vocabulary words. The most famous method to compute these distributions is the Gibbs sampling. Gibbs sampling is a relatively old statistical method to approximate distributions [Geman and Geman, 1984] and was first used with LDA in [Griffiths, 2002]. Given a number of topics  $K$ , the Gibbs sampling algorithm tries to correctly assign to each vocabulary word a probability for each topic. Starting from random distributions, it iteratively improves them until reaching a steady state.

Since its introduction, LDA has been used as a basis for new topic models extending it. For instance, a new topic model is proposed in [Wallach, 2006]. Contrary to the original LDA model, where each text is generated as a bag of words, i.e. the generated words are not sorted, in this topic model a new word is generated by additionally taking into account the previous generated word. After taking into account the words order of a document, work has also been done to consider the documents ordering inside a corpus [Blei and Lafferty, 2006]. This model allows to visualize topics predominance over time. Another example of topic model extending LDA is presented in [Rosen-Zvi et al., 2004]. In this work, additional information about authors is included. Each author is associated with a distribution over topics, modeling its interests.

Introduced before LDA, another major topic model is the Latent Semantic Analysis (LSA) [Deerwester et al., 1990] and its probabilistic variant pLSA (probabilistic Latent Semantic Analysis) [Hofmann, 1999]. In LSA, when considering a corpus of  $N$  documents and a vocabulary of  $M$  words, a  $N \times M$  matrix is built based on words occurrences in the documents. This matrix is then factorized using Singular Value Decomposition [Klema and Laub, 1980]. Documents are then represented as vectors whose similarity can be compared by computing their cosine similarity. While topics are orthogonal in LSA, they are distributions over the whole vocabulary in pLSA, like in LDA. The matrix is factorized using non-negative matrix factorization, which implies a non-orthogonal

---

decomposition. Additionally, this decomposition enables the identification of topics by grouping similar words together [Donoho and Stodden, 2004].

In many techniques such as LSA, vectors are used to represent text documents. This is called the Vector Space Model [Salton et al., 1975]. In this space, each dimension corresponds to a corpus term. A term is generally a word, but other definitions are possible, such as a term being a sentence. Documents, as lists of terms, are represented as vectors in the Vector Space. The similarity between two documents can then be computed as the cosine similarity of the two vectors [Gabrilovich and Markovitch, 2007].

Topic models are one way to find topics hidden in a text, text categorization is another one. Contrary to topic models, which are unsupervised, text categorization models involve supervised learning. This means that in text categorization, a model is trained using a set of texts that are already annotated with topics. Only after being trained, a text categorization model can be used to find topics. Several machine learning algorithms have been proposed [Sebastiani, 2002], such as using support vector machines [Ferraresi et al., 2008].

---

---

**Part II.**

**Background and resources**

---

---

## 3 Introduction

In this section, we describe the resources that we use in this thesis. Our system takes as input a Disambiguated Distributional Thesaurus (DDT), presented in [Faralli et al., 2016]. This DDT can be seen as an undirected graph, whose nodes are word senses. For example, a word like *jaguar* will be represented by several nodes in this graph, whether if it means the animal or the car. Two nodes are connected if they have a similar meaning: for example there should be an edge between the node *jaguar* (animal sense) and the node *lion*.

Our system performs a clustering of this graph of word senses, each cluster should contain highly connected nodes. It is therefore likely that these clusters contain highly semantically related words, denoting a topic. For instance, a topic could contain the words *jaguar lion tiger cat* etc. whereas another one could contain the words *Mercedes, Audi, Jaguar, BMW*.

In a next step, we want to be able to easily guess which topic the word senses contained in the cluster denote. In order to enable this, we find accurate hypernyms for each topic. For example, the topic containing the word senses *cat, tiger, lion, jaguar* etc. could have as hypernyms words like *feline* or *animal*.

---

## 4 Disambiguated Distributional Thesaurus (DDT)

---

### 4.1 Construction of the thesaurus

---

The system that we developed to build structured topics takes as input a distributional thesaurus. A thesaurus is a linguistic resource meant to link the natural language of users to the one used in lexical resources and is useful to address potential issues of natural language understanding such as synonymy or homonymy.

A thesaurus is composed of lists of words. Each of these lists represents one concept: that is to say, the words present in each list are similar in terms of meaning. They can be of course synonyms but an entry in the thesaurus is not bounded to a list of synonyms: it is more general than that, such that antonyms can also be part of the list. Words in a list can also be linked in a hierarchical way: there can be hypernyms and hyponyms, meronyms and holonyms.

A hyponym-hypernym relationship between two terms A and B happens when we can say that "A is a B". For instance, with the two words *cat* and *animal*, *animal* is a hypernym of *cat* because a cat is an animal. The other way around, we say that *cat* is a hyponym of *animal*. A meronym-holonym relationship between two terms A and B happens when we can say that "A is a part of B". For example, with the two words *room* and *house*, *house* is a holonym of *room* and *room* is a meronym of *house*, since a house contains rooms.

Thesauri are widely used by linguists but are also available publicly, there are indeed several thesauri on the Web. As an example, Figure 4.1 shows the results for the word *car* in the website thesaurus.com.

There are 33 words returned as *synonyms* of *car*. As we can easily notice, the term *synonym* is not really appropriate here, as only the two first words in the list *auto* and *automobile* are actually synonyms of *car*, but this is not our concern. We can notice that the list contains indeed some synonyms of *car* (*auto*, *automobile*), some words with a similar meaning but not synonyms (*bus*, *truck*), some hypernyms (*machine*), some hyponyms (*limousine*, *touring car*) and some meronyms (*wheels*). Together, these words form a concept.

The thesaurus we use in this thesis is a distributional thesaurus i.e. it is based on distributional semantics. Distributional semantics is a research area aiming at measuring semantic similarity between terms based on their distributions in text in large corpora, following the Distributional hypothesis that linguistic items with similar distributions have similar meanings. Indeed, two similar words should be statistically distributionally similar, that is to say, they should have a similar context. The other way around, by measuring the similarity of contexts for two words, we can elaborate an hypothesis on their semantic similarity. This is precisely the Strong Contextual

## Synonyms for car

Common  Informal 

*noun* vehicle driven on streets

auto	motor	bucket	heap	wheels
automobile	pickup	buggy	jalopy	wreck
bus	ride	compact	junker	clunker
convertible	station wagon	conveyance	motorcar	gas guzzler
jeep	truck	coupe	roadster	touring car
limousine	van	hardtop	sedan	
machine	wagon	hatchback	subcompact	

Figure 4.1.: Results page for the word "car" in thesaurus.com.

Hypothesis introduced in [Miller and Charles, 1991]. The more often two words appear within the same context, the more likely they are to be semantically similar.

More precisely, our distributional thesaurus is an output of the JoBimText project pipeline [Faralli et al., 2016]. The JoBimText project is a distributional semantics project developed and maintained by IBM Research and the Language Technology Group at the TU Darmstadt.<sup>1</sup> It aims to build a framework for computational semantics to address multiple problems encountered in the computational linguistics research area: word sense disambiguation, lexical ambiguity, lexical substitutability, parsing. The JoBimText framework is fully unsupervised, meaning that it can self adapt to multiple amounts of data or different languages. [Glozzo et al., 2013].

The figure 4.2 represents the semantic similarity computation pipeline. It takes raw text as input and outputs semantic similarity values between words in this text.

The first step in the pipeline is the holing operation introduced in [Biemann and Riedl, 2013]. This step takes as input the corpus text and extracts distributional features from structural observations. These observations can be of several types: n-grams, dependency parses, positional co-occurrences etc. Let's consider an example in which the holing operation is used with dependency parses and the toy sentence *I suffered from a cold and took aspirin*, the output of the dependency parsing is represented in Figure 4.3.

For each relationship in output, we perform the holing operation: we transform each relation  $A(B,C)$  into two word-feature pairs:  $B - A(@@,C)$  and  $C - A(B,@@)$ . The @@ represents a "hole" and is used as a generic for any word. With our toy sentence, we get the pairs  $suffered - nsubj(@@, I)$ ,  $I - nsubj(suffered, @@)$ ,  $took - nsubj(@@, I)$ ,  $I - nsubj(took, @@)$  and so on.

The holing operation is performed on the whole corpus and the features are grouped by word and counted. Language elements and context features are also counted individually, the count values are used to adapt each word-feature pair count value. Indeed, words that come often would be advantaged compared to rare ones if only the word-feature pair count value was used. Instead of the raw count, we get a frequency significance measure for each word-feature pair. A pruning

<sup>1</sup> <http://lt.informatik.tu-darmstadt.de/de/software/jobimtext/>



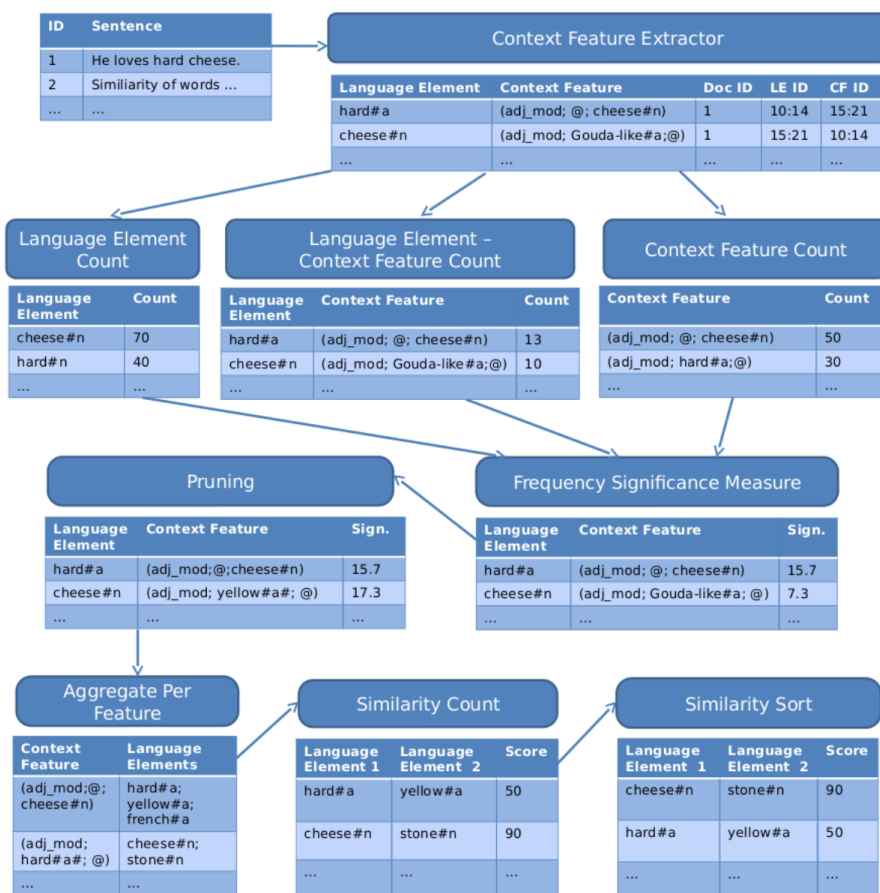


Figure 4.2.: Pipeline for the computation of semantic similarity values. Source: [Biemann and Riedl, 2013]

nsubj(suffered, I)  
 nsubj(took, I)  
 root(ROOT, suffered)  
 det(cold, a)  
 prep\_from(suffered, cold)  
 conj\_and(suffered, took)  
 dobj(took, aspirin)

Figure 4.3.: Dependency parsing output for the sentence *I suffered from a cold and took aspirin.*

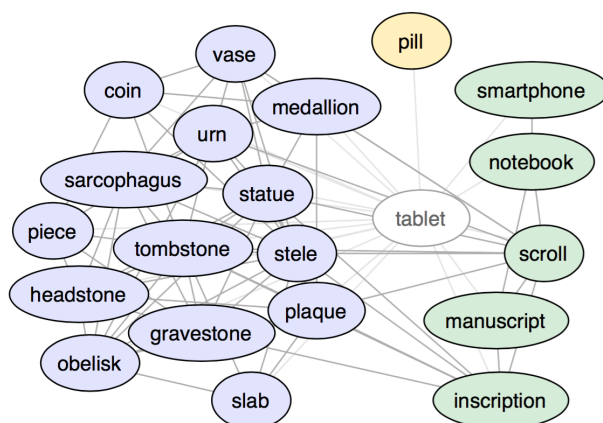
entry	similar terms
mouse:NN	rat:NN, rodent:NN, monkey:NN, animal:NN ...
mouse:NN	keyboard:NN, computer:NN, printer:NN, laptop:NN, device:NN ...
keyboard:NN	piano:NN, synthesizer:NN, organ:NN, instrument:NN ...
keyboard:NN	keypad:NN, mouse:NN, screen:NN, touchpad:NN ...

**Table 4.1.:** Sample from a thesaurus in which none of the words are disambiguated.

operation is then performed to keep only the most significant word-feature pairs. Then, words are grouped by feature: for each feature  $F$ , all words  $W$  having a word-feature pair  $W-F$  are grouped together. In the next step, all pairs of words get a similarity measure which is the number of groups in which they co-occur. The more features they share, the more similar we assume them to be, according to the Strong Contextual Hypothesis. The triplets (word  $w_1$ , word  $w_2$ ,  $\text{sim}(w_1, w_2)$ ) are finally sorted by similarity.

This induces a distributional thesaurus: for each word, we know which words are semantically similar to it. But this is not yet the thesaurus we use as input for our system. The thesaurus we use is a Disambiguated Distributional Thesaurus (DDT), that is to say: it is a distributional thesaurus in which the word senses are disambiguated: for example the word *tablet* should have at least two different senses: the animal and the device used along a computer. To disambiguate words in the distributional thesaurus we have so far, a few more steps are required.

The distributional thesaurus that we get looks like the Table 4.1, it is a list and each item is composed of one word (the entry) and the most similar words to this one. None of these words are disambiguated.



**Figure 4.4.:** Neighborhood graph of the word *tablet*. Source: [Simon, 2015]

First, we find the different senses of the entry word. We do this using its most similar words. In Table 4.1 we have the entry word *mouse* and its most similar words *rat*, *keyboard*, *rodent*, *computer* etc. These words belong either to the animal sense of *mouse*, or to the computer device one. To detect the different senses of the word *mouse*, we will try to split the list of the most similar words into several groups. To do this, we build the graph of the word *mouse* and its most similar words:

the nodes of this graph are words and we draw an edge between two nodes if the two words have their semantic similarity measure above a certain threshold. This graph is represented in Figure 4.4. We then remove the entry node (*tablet*) and all its edges to obtain the neighborhood graph of the entry node. In Figure 4.4, we can clearly see the 3 communities in this graph; we apply a clustering method which automatically finds these communities. The method is unsupervised and determines the number of communities by its own.

Once we identified the different senses that each word in our distributional thesaurus can have, we can split each list of most similar words into several ones according to the senses of the entry word. At this moment, we get the thesaurus represented in Table 4.2. But this thesaurus is not fully disambiguated yet, because the similar words are not.

entry	similar terms
mouse:NN:0	rat:NN, rodent:NN, monkey:NN, animal:NN ...
mouse:NN:1	keyboard:NN, computer:NN, printer:NN, laptop:NN, device:NN ...
keyboard:NN:0	piano:NN, synthesizer:NN, organ:NN, instrument:NN ...
keyboard:NN:1	keypad:NN, mouse:NN, screen:NN, touchpad:NN ...

**Table 4.2.:** Sample from a thesaurus in which only the entry word is disambiguated.

The last step in the building of our Disambiguated Distributional Thesaurus (DDT) is the sense disambiguation of the most similar words for each word entry. We note that, since each word in the thesaurus has its own most similar words, we already know the different senses that each most similar word has, we just need to map each occurrence of this word in a list of most similar words to one of its senses. To achieve this, we use the contexts of this word: the other most similar words and for each of its senses, the list of most similar words to these senses. For each sense, we compute the cosine similarity between the other similar words and the similar words for this word and we assign the sense that gives the highest similarity value. The result is a fully disambiguated distributional thesaurus, such as the example showed in Table 4.3. We notice that among the similar terms of *keyboard:NN:1*, the word *mouse:NN* is successfully disambiguated into the word sense *mouse:NN:1* and it is the same for the word *keyboard:NN* disambiguated into *keyboard:NN:1* when the word appears among the similar terms of *mouse:NN:1*.

entry	similar terms
mouse:NN:0	rat:NN:1, rodent:NN:0, monkey:NN:0, animal:NN:0 ...
mouse:NN:1	keyboard:NN:1, computer:NN:0, printer:NN:2, laptop:NN:1, device:NN:0 ...
keyboard:NN:0	piano:NN:1, synthesizer:NN:1, organ:NN:0, instrument:NN:2 ...
keyboard:NN:1	keypad:NN:2, mouse:NN:1, screen:NN:1, touchpad:NN:0 ...

**Table 4.3.:** Sample from a fully disambiguated thesaurus.

---

## 4.2 IS-A relationships

---

A IS-A relationship between two objects A and B happens when "A is a B". Applied to linguistics, it has the same meaning as the hyponym-hypernym relationship, but can be used more generally than the hyponym-hypernym terms, which are bounded to natural language processing.

---

In parallel with the computation of the DDT, IS-A relationships are also searched in the corpus. To find them, Hearst patterns [Hearst, 1992] are run over the corpus. Hearst patterns are lexico-syntactic patterns to find IS-A relationships in text, some examples (to find that "X is a Y") are "Y, such as X", "X is a Y", "Y, including X", "X and other Y". By running Hearst patterns over the corpus, a IS-A relationships database is built. IS-A databases are used later in this thesis.

---

### 4.3 DDTs used in our experiments

---

In this thesis, four DDTs were used, they were built on two distinct corpora: a news corpus of 100 million sentence news corpus from Gigaword [Graff and Cieri, 2003] and LCC [Biemann et al., 2007] and a wiki corpus with 35 million sentences from Wikipedia. In addition, a parameter for the sense induction algorithm was tuned to give several average sense granularities. Finally, we have two DDTs based on Wikipedia and two based on the news corpus. Their characteristics can be shown in Table 4.4. We notice that the DDT *ddt-wiki-n30* contains way more word senses per word than the other DDTs, due to the sense clustering parameter value  $n$ . There are about 6 word senses per word in average, which seems to be a lot. The other DDTs have a lower average polysemy, around 2.

dataset	# words	# word senses	average polysemy	average # sim. terms
ddt-wiki-n30	258k	1.5M	6.0	16.9
ddt-wiki-n200	206k	368k	1.8	59.3
ddt-news-n50	200k	461k	2.3	44.3
ddt-news-n200	207k	332k	1.6	63.9

**Table 4.4.:** Statistics of the datasets used in our experiments.

---

**Part III.**

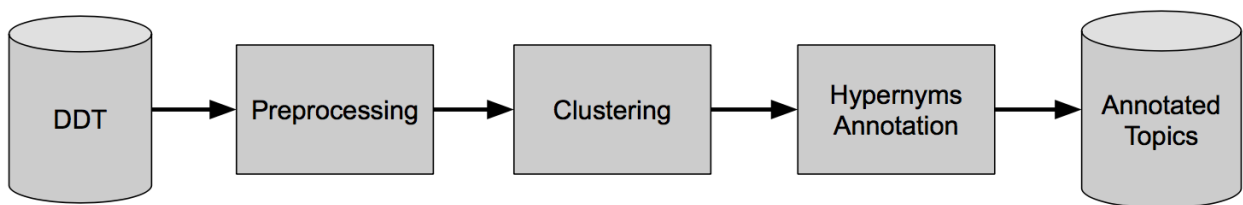
# **Extraction of the structured topics**

---

---

## 5 Presentation of the pipeline

In this part, we describe our system which builds the topics, using a Disambiguated Distributional Thesaurus (DDT) as input. The system can be represented as a pipeline composed of three blocks. The pipeline is represented in Figure 5.1.



**Figure 5.1.:** Dataflow diagram of our system for building topics.

In the first block, the DDT is filtered. We keep only nouns and proper nouns and we also discard very rare words. It is in the second block of the pipeline that we create the topics. We isolate clusters of word senses by applying a clustering method on the filtered DDT. Each cluster is a potential topic. The third step of the pipeline is dedicated to annotate each topic with representative hypernyms. The topics often contain several hundreds of word senses and we try to make it easy to grasp the gist of each of them.

---

---

## 6 Preprocessing of the DDTs

The disambiguated distributional thesauri that we use in this thesis were constructed automatically and contain erroneous word senses that need to be filtered out before we apply a graph clustering method to find the topics. In the following we explore several ways to filter the word senses in the DDT.

First, every word sense in the DDT has a Part Of Speech tag attached to it. The possible tags are the ones of the Penn Treebank. Most of the time, topic modeling deals only with nouns. In our case, we decide to follow this and keep only nouns and proper nouns and we discard all word senses with other tags. In Table 6.1, we show the results of this first filtering on the different datasets. Compared to the original metrics shown in Table 4.4, we can see that by keeping only nouns and proper nouns and discarding other word senses, we remove almost half of them. The second value is the average of the number of similar word senses. This value does not vary much in this case.

dataset	# words	# word senses	average polysemy	average # sim. terms
ddt-wiki-n30	195k	1.2M	5.7	15.5
ddt-wiki-n200	154k	292k	1.9	50.2
ddt-news-n50	125k	310k	2.5	39.6
ddt-news-n200	131k	229k	1.8	57.3

**Table 6.1.:** Datasets metrics after removing all word senses with non noun tags.

Second, word senses in the DDT can be filtered out if these word senses are non frequent. We use a file containing about 8 million words. Each of them has a frequency value. This value was obtained by counting word occurrences within a 100 million words from the LCC corpus [Biemann et al., 2007]. Here we have to note that this file contains words and not word senses. Therefore, for each word sense, the value we use is the global frequency for this word.

Using this file, we can first discard potential misspellings. Indeed, there are word senses that are in the DDT even though they are misspelled or contain extra characters like in *??They???* which is present in the DDT. If a word sense in the DDT is not present in the frequency file, we remove it. Secondly, we can also discard word senses below a certain frequency threshold. The average frequency in the file is 242. Here, it is important to note that the values vary a lot, from 0 to around 10 000 000. We take this into account and try several frequency threshold values: 0, 500, 1200 and 2000 for the different datasets: *ddt-news-n200*, *ddt-news-n50*, *ddt-wiki-n200* and *ddt-wiki-n30*. We filter all word senses according to this threshold: the entry word and the similar terms. If an entry word does not have any similar word after the filtering, we also discard this entry word. The results are respectively shown in Tables 6.2, 6.3, 6.4 and 6.5.

We obtain similar results with the four DDTs. The number of words and of word senses logically decreases when the threshold increases, since we filter out more terms. The average polysemy

slightly increases with the threshold. This is due to the fact that the most frequent words are also the most general ones and thus they generally have more word senses than less frequent words that are more specific. The average number of similar terms does not vary significantly. Based only on these numbers, it is difficult to know which threshold is the best one for our system, we will have to wait until we extract the topics such that we can analyze them and decide which threshold gives the best ones.

frequency threshold	# words	# word senses	avg. polysemy	avg. # sim. terms
0	130891	228375	1.8	57.3
500	47543	111960	2.4	66.6
1200	28115	68927	2.5	66.1
2000	20374	48982	2.4	62.4

**Table 6.2.:** *ddt-news-n200* filtered with different frequency thresholds.

frequency threshold	# words	# word senses	avg. polysemy	avg. # sim. terms
0	110776	272150	2.4	39.6
500	46209	172377	3.7	40.0
1200	27321	110228	4.0	38.0
2000	19799	79816	4.1	35.2

**Table 6.3.:** *ddt-news-n50* filtered with different frequency thresholds.

frequency threshold	# words	# word senses	avg. polysemy	avg. # sim. terms
0	128040	243882	1.9	49.2
500	41623	102378	2.5	50.7
1200	26359	66493	2.6	49.4
2000	19540	49168	2.6	47.5

**Table 6.4.:** *ddt-wiki-n200* filtered with different frequency thresholds.

frequency threshold	# words	# word senses	avg. polysemy	avg. # sim. terms
0	161631	1002166	5.8	14.9
500	44782	426303	8.9	13.1
1200	27631	277236	9.5	12.3
2000	20298	206178	9.8	11.6

**Table 6.5.:** *ddt-wiki-n30* filtered with different frequency thresholds.

Finally, we could think of filtering the DDT based on the semantic similarity measure between a word sense and its similar word senses. The values lie between 0 and 1. By removing the word senses with the lowest semantic similarity, we would keep the strongest links between word senses. This would improve the global quality of our thesaurus. However, removing too many links in our DDT would make it useless because too sparse. It is also difficult to determine which



---

weight threshold would give the best results. Moreover, one could argue that this filtering is not very useful because the weights are taken into account by the clustering methods anyway. Indeed, if two nodes are linked by an edge with a low weight, it is less likely that they end up in the same cluster than if the weight is higher. Given these points, we decide not to use this filtering: we do not discard any link between word senses based on a low semantic similarity.

---

# 7 Graph clustering of the DDTs

---

## 7.1 Introduction

---

The next step in our system pipeline is the graph clustering in order to identify the topics. The Disambiguated Distributional Thesaurus can be seen as a graph of word senses: each word sense is a node of the graph and two nodes are connected by an edge if the two word senses are semantically similar i.e. for two word senses A and B, if B belongs to the list of most similar word senses of A or vice versa. The edges are undirected and their weight is equal to the semantic similarity value between the two word senses A and B.

Graph clustering is the method meant to detect communities of nodes in a graph and split this graph according to these communities. A graph clustering method takes therefore a graph as input and outputs a set of clusters, each cluster being a set of nodes. All the nodes should be present in one and only one cluster: there is no overlapping between nodes clusters.

Clustering the graph enables us to identify groups of nodes which are intensively connected to each other. In our case of a graph of word senses linked regarding their similarity, each cluster should contain words which are closely semantically related to each other. Each cluster could therefore be a topic e.g. a cluster whose nodes are animals names would form a topic about animals.

There exist multiple algorithms to perform graph clustering. Among the most popular ones are the Kerne K-means algorithm [Likas et al., 2003] and several different algorithms based on it or the Markov Chain Clustering (MCL) algorithm. However, the problem with most of the popular methods is that the number of expected clusters has to be given as input, along with the graph. In our case, we do not know how many clusters the method should find, therefore we discard algorithms with this parameter as input and focus our attention on the algorithms in which the number of clusters is determined by the algorithm during the computation. More precisely, we focus on three such methods: MCL, the Louvain Method and Chinese Whispers.

---

## 7.2 Markov Chain Clustering

---

The Markov Chain Clustering method (MCL) was introduced in 2001 in [Van Dongen, 2001]. It is based on the concept of Markov Chains. A Markov Chain is a transition process in a state space. Each transition from one state to another happens according to a certain probability, all the probabilities for the transitions out of each node sum up to 1, there is also a probability to stay in the current state. We can think of the concept of the random walker: a walker goes from state to state and chooses the next state with the transitions probabilities.

The MCL method applies this concept on a graph, the states are the nodes and the probabilities of transitions to other nodes are computed proportionally to the edges weights. The idea of MCL

---

is the one of the random walker: it starts with a node  $N$  and goes from node to node according transition probabilities. If  $N$  is inside a cluster of nodes, the random walker has globally good chances to stay in this cluster after several iterations, because the weights should be higher for edges inside the cluster than for edges going out of the cluster, by definition.

The method uses the adjacency matrix of the graph. Its values are normalized such that for each column, the values sum up to 1. Each value in row  $i$  and column  $j$  then corresponds to the probability that a random walker has to move from the node  $j$  to the node  $i$ . The method is divided into two steps that are iteratively performed: the expansion step and the inflation step. During the expansion step, the matrix is squared. This tends to connect the graph and offer more travel possibilities to the random walker. The inflation step has an opposite consequence: it consists in bringing each value in the matrix to the power  $r$  (with  $r > 1$ , usually  $r = 2$ ). This operation increases the differences between the lowest values and the highest ones. A pruning is then performed, such that each value below a certain threshold is assigned 0. The expansion and the inflation steps are performed until the matrix converges, this means that the matrix  $M$  obtained after the pruning is idempotent:  $M * M = M$  (because then  $M$  is no more modified by the method). Once this state is reached, the clusters found by the method can be determined by analyzing the output matrix: in each column where there is at least one non-zero value, the row index of the non-zero values in this column correspond to the nodes in one cluster.

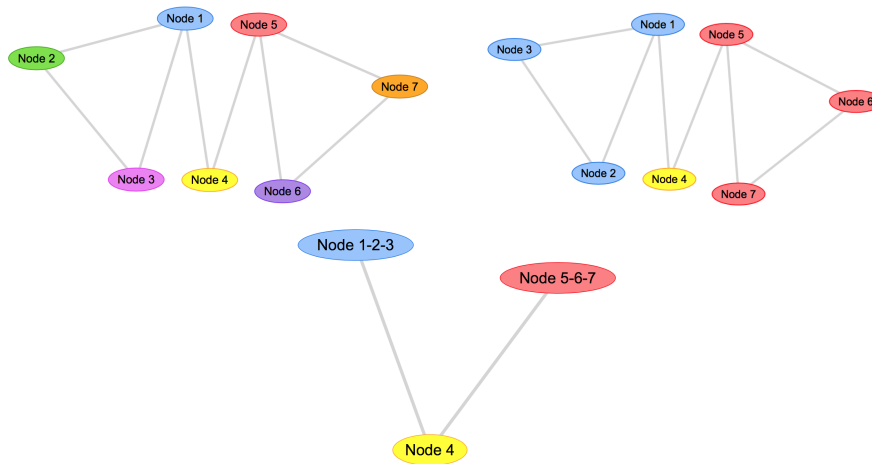
---

### 7.3 Louvain Method

---

The Louvain Method was introduced in 2008 in [Blondel et al., 2008]. It is based on the measure of modularity. The modularity is a measure to evaluate the quality of a graph partition. It comes from the intuitive observation that, given a graph and its clusters, if there are very few edges between nodes in different clusters and most of the edges are between nodes inside a same cluster, then the quality of the clustering of this graph is good. The modularity compares the number of edges inside a cluster to the number of edges between the clusters. Its value lies between -1 and 1: -1 means that all the edges are between clusters and 1 means that all the edges are inside a cluster and as a consequence, that the clusters form disconnected subgraphs of the original graph, in this case the clustering is ideal.

The method is composed of two phases, which can be performed several times if needed. Initially in the first phase, each node of the graph is a community. Then, for each node, values of the modularity are computed whether this node stays in its community or is moved to each of the other communities. Finally, the node is moved to the community for which the modularity has a maximal value. If the global modularity is not improved by moving the node to another community, it stays in its original community. The first phase is complete after the iteration over each node in the graph. In the second phase, each community is collapsed to only one node such that the graph is transformed to a graph of communities. Edges are drawn between communities according to the edges and nodes previously present in these communities. After the second phase, a new smaller graph (the graph whose nodes are the communities) is obtained. The two phases can be applied again on this graph, to build communities of communities. At the end of the method, the result is a hierarchical clustering of the original graph.



**Figure 7.1.:** Example of the two steps of the Louvain Method.

A toy example is shown in Figure 7.1, no weight has been assigned to the edges and it is only meant to demonstrate the two phases of the method. The communities are represented by the different colors: initially each node has its own community. Then, nodes are moved from their community to another one if this operation improves the global modularity of the graph partition. In our example, the nodes 1,2,3 are grouped in the same community, the node 4 stays alone in its initial community and the nodes 5,6,7 are also grouped into one community. In the second phase, the communities are collapsed and a new graph with only three nodes is obtained. It is then possible to apply the first phase again and so on.

---

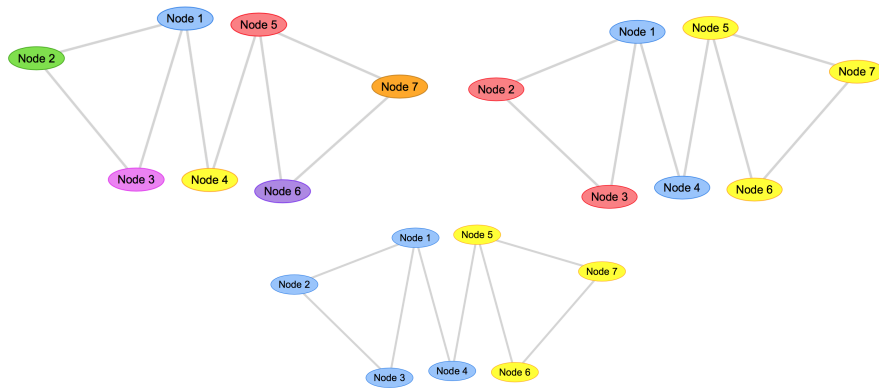
## 7.4 Chinese Whispers

---

Chinese Whispers is a graph clustering method introduced in 2006 in [Biemann, 2006]. Chinese Whispers finds communities of nodes in a bottom-up fashion like the Louvain Method, but does not use the modularity of the graph partition to do it. Instead, it relies on the edges weights.

The method is outlined in Figure 7.2. It starts with an initialization of the graph partition in which, like in the initialization for the Louvain Method, each node gets its own class (or community). Then, all nodes are iteratively processed, in a randomized order: each node inherits the class whose sum of edge weights to this node is maximal. If there are several classes with the same maximal weight, one is chosen randomly. Once all nodes have been processed, one iteration of the method is completed. New iterations are performed while the graph partition has been modified by the last one. Once the partition is stabilized, the method stops and outputs the graph partition.

For instance, given the same graph as for the Louvain Method, the graph partition of each iteration could be the one represented in Figure 7.2, if we assume that the order of the processing of nodes is for example 1, 3, 2, 7, 4, 5, 6.



**Figure 7.2.:** Example of the steps during the Chinese Whispers clustering method.

---

## 7.5 Results of the clustering

---

Each of the three clustering methods introduced before were applied on our filtered DDTs. Although it is not possible to establish for sure which method and which parameters give the best results only with these elements, they already provide some insight about the respective outcome quality.

More precisely, we take a look at 5 different measures for each result: the number of clusters found by the method (# clusters), the average number of word senses per cluster (avg. size clusters), the standard deviation for those cluster sizes (size stand. deviation), the minimum cluster size (min. size) and the maximum cluster size (max. size).

Regarding the number of clusters, there is no target value, as this depends largely on the corpus. Ideally, we should have the same number of clusters with the different methods and also with different frequency threshold values, since increasing the threshold is unlikely to remove all words from a topic but rather to remove some words from each one.

In the case of the cluster size, it is also difficult to say what is the ideal size. Nevertheless, we can state that the topics must not be too small (less than 100) or too big (more than a few thousands) or they might contain not enough or respectively too much information. The standard deviation should be as low as possible, although a few extreme topics (very small or very big) might have a great impact on the standard deviation and since these clusters are meant to be discarded after further evaluation, the value might not be very representative of the global topics quality.

The two last values, the minimum cluster size and the maximum cluster size allow us to grasp the presence of the extreme topics that we evoke before.

In order to have a formal evaluation of the results given by each clustering method, we compute the Davies-Bouldin index for each set of clusters. Introduced in [Davies and Bouldin, 1979], the index is an internal measure of the clusters separation. It involves two measures about the clusters partition of the graph which are:

1) the average distance between nodes within a cluster and the centroid of this cluster. The centroid is the most central node in the cluster. We find it by computing a simple centrality measure of each

---

node within the cluster. There exist a large variety of centrality measures that were developed to quantify importance of nodes within a network. In our case, the measure that we use is defined as follows: for each neighbour  $n'$  of a node  $n$ , we count how many neighbours  $n''$  this neighbour  $n'$  has. The centrality of the node  $n$  is the sum of the number of neighbours  $n''$  of each of its neighbours  $n'$ . The distance is the inverse of the weight of an edge in the graph. For each node in the cluster, the distance to the centroid is the shortest path computed with the Dijkstra algorithm [Dijkstra, 1959].

2) the distance  $d$  between clusters, which is the minimal distance between each couple of clusters.

The Davies-Bouldin metric  $DB$  is computed as follows:

$$DB = \frac{1}{n} \sum_{i=1}^n \left( \max_{i \neq j} \left( \frac{\sigma_i + \sigma_j}{d(i, j)} \right) \right).$$

For each cluster  $i$ , we compute the maximum over all clusters  $j$  of the sum of the average distances of nodes to their centroid for the cluster  $i$  and the cluster  $j$ , divided by the distance between the clusters  $i$   $\sigma_i$  and  $j$   $\sigma_j$ . The result is the average of these maxima.

The numerator in the fraction represents the intra cluster similarity. The highest the value is, the closest the nodes are to their centroid. The denominator represents the inter cluster similarity: the further the two clusters are from each other, the least similar they are. By taking the maximum over the values for each cluster  $j$ , we select the most similar cluster to the cluster  $i$ . We then want this closest and most similar cluster to be the least similar to the cluster  $i$ , leading to a good clusters separation. Therefore, we want the overall Davies-Bouldin index to be as small as possible. We can note that since all distances are strictly positive, the result is also positive.

Additionally, we also discuss the computation time of each method. We indeed aim to keep the total computation time of our system as short as possible.

Results can be seen in Tables 7.1, 7.2, 7.3, 7.4 and 7.5. We observe the same tendencies for Chinese Whispers and Louvain Method: the number of clusters, their average size and the standard deviation for the clusters sizes decrease when we increase the frequency threshold. This can be explained by noticing that when we remove more word senses from the graph by increasing the frequency threshold, first the graph gets of course smaller but it also tends to be more scattered. Clusters obtained with Chinese Whispers are globally bigger than those obtained with the Louvain Method. As a consequence, the Louvain Method produces more clusters than Chinese Whispers.

Regarding the Davies-Bouldin index, we can notice that the value decreases when the frequency threshold is increased, meaning that the quality of the clusters is better. Chinese Whispers and the Louvain Method give globally similar indices, Chinese Whispers gives more often greater indices but the difference is too small to be meaningful.

DDT	frequency	# clusters	avg. cluster size	size stand. deviation	min. size	max .size
news-n200	0	595	382.7	1170.2	1	14054
news-n200	500	359	311.4	975.6	1	9309
news-n200	1200	330	208.5	570.5	1	6452
news-n200	2000	286	170.8	506.3	1	5971
news-n50	0	835	325.9	866.7	1	10726
news-n50	500	668	258.0	583.7	1	7709
news-n50	1200	596	184.9	396.0	1	3557
news-n50	2000	547	145.8	289.2	1	2082
wiki-n200	0	838	291.0	879.1	1	10501
wiki-n200	500	539	189.9	665.7	1	9066
wiki-n200	1200	467	142.3	591.5	1	9530
wiki-n200	2000	433	113.5	479.8	1	7580
wiki-n30	0	14599	68.6	579.7	1	36222
wiki-n30	500	8503	50.1	372.1	1	21786
wiki-n30	1200	6857	40.4	249.9	1	11445
wiki-n30	2000	5914	34.8	187.5	1	7040

**Table 7.1.:** Clustering results for Chinese Whispers applied to the different models.

DDT	freq. threshold	Davies-Bouldin index
ddt-news-n200	0	754.7
ddt-news-n200	500	<b>9.2</b>
ddt-news-n200	1200	<b>4.8</b>
ddt-news-n200	2000	<b>4.6</b>
ddt-news-n50	0	563.2
ddt-news-n50	500	776.9
ddt-news-n50	1200	96.5
ddt-news-n50	2000	<b>15.7</b>
ddt-wiki-n200	0	124.3
ddt-wiki-n200	500	32.2
ddt-wiki-n200	1200	17.0
ddt-wiki-n200	2000	<b>10.3</b>
ddt-wiki-n30	0	81.2
ddt-wiki-n30	500	37.2
ddt-wiki-n30	1200	19.5
ddt-wiki-n30	2000	18.3

**Table 7.2.:** Davies-Bouldin indices for Chinese Whispers applied to the different models, lower values mean better clustering.

DDT	frequency	# clusters	avg. cluster size	size stand. deviation	min. size	max .size
news-n200	0	773	295.0	1099.1	1	15761
news-n200	500	440	254.4	633.9	1	5455
news-n200	1200	375	183.8	456.4	1	5023
news-n200	2000	343	142.8	327.5	1	3158
news-n50	0	929	292.9	786.2	1	10703
news-n50	500	711	242.4	558.6	1	5893
news-n50	1200	619	178.0	375.9	1	3049
news-n50	2000	557	143.2	274.6	1	2003
wiki-n200	0	998	244.3	830.0	1	12516
wiki-n200	500	662	154.6	448.9	1	5888
wiki-n200	1200	559	118.9	346.5	1	3569
wiki-n200	2000	529	92.9	249.3	1	2550
wiki-n30	0	17651	56.7	483.6	1	26854
wiki-n30	500	9665	44.1	331.3	1	17922
wiki-n30	1200	7676	36.1	225.1	1	10078
wiki-n30	2000	6714	30.7	171.5	1	5509

**Table 7.3.:** Clustering results for the Louvain Method applied to the different models.

DDT	freq. threshold	Davies-Bouldin index
ddt-news-n200	0	62.5
ddt-news-n200	500	12.8
ddt-news-n200	1200	<b>3.9</b>
ddt-news-n200	2000	<b>3.9</b>
ddt-news-n50	0	405.1
ddt-news-n50	500	458.1
ddt-news-n50	1200	68.8
ddt-news-n50	2000	<b>12.4</b>
ddt-wiki-n200	0	81.3
ddt-wiki-n200	500	30.9
ddt-wiki-n200	1200	12.5
ddt-wiki-n200	2000	<b>11.3</b>
ddt-wiki-n30	0	63.9
ddt-wiki-n30	500	30.3
ddt-wiki-n30	1200	19.6
ddt-wiki-n30	2000	<b>12.1</b>

**Table 7.4.:** Davies-Bouldin indices for the Louvain Method applied to the different models, lower values mean better clustering.



---



---

frequency	# clusters	avg. size clusters	size stand. deviation	min. size	max .size
2000	1799	27.1	88.1	1	1663

**Table 7.5.:** Clustering results with MCL, the DDT *ddt-news-n200* and 2000 as word frequency threshold.

Compared to the two other methods, Chinese Whispers and the Louvain Method, the Markov Chain Clustering method gives disappointing results. First, it is not as efficient as the other methods. The complexity of MCL is indeed  $O(k * |n^2|)$  with  $n$  the number of nodes and  $k$  the number of iterations. The results can be seen in Table 7.5. With 2000 as the frequency threshold, it took around 100 times longer than the Louvain Method, which was slightly faster than Chinese Whispers. This is a major drawback, since we want our pipeline to complete running in a reasonable amount of time. The Davies-Bouldin index equals to 4.5, which are comparable to those obtained with Chinese Whispers (4.6) and the Louvain Method (3.9). Furthermore, by inspecting the results for the frequency 2000, we noticed multiple uninterpretable topics. In Table 7.5, we can notice that MCL gives more and smaller clusters than CW and LM. This could be corrected by tuning the method parameters, but the clusters quality could only get worse than this, because several clusters would be merged to give less and bigger clusters in the end.

---

### 7.5.1 Conclusion

---

After inspecting results for each method, we decide to keep Chinese Whispers and the Louvain Method and to proceed further with them. On the other hand, we discard Markov Chain Clustering from consideration in our approach, since the two other methods are both better in terms of quality of clustering and faster.

---

## 8 Topic annotation with hypernyms

We want to find a few words that represent a topic well, so that it becomes easier to grasp what this topic is about. To do this, we can use hypernyms. A word A is an hypernym of a word B if we can say that B is a kind of A. For instance, *feline* is the hypernym of *cat* because a cat is a feline. If we manage to find words that are hypernyms of all words in one topic then this hypernym is a relevant word to represent this topic e.g. the word *feline* for a topic which would contain the words *cat*, *lion*, *tiger*, *jaguar*, etc.

In order to find these hypernyms, we use two kinds of lexical resources: a manually built lexical database, WordNet and a IS-A relationships database built with unsupervised methods.

---

### 8.1 Hypernyms from WordNet

---

WordNet [Miller et al., 1990] is a lexical database for the English language developed and maintained at Princeton University. It contains synsets, which are groups of synonyms and relationships between these synsets or the words senses they contain. Among these relationships, the hyponym-hypernym relationship is available for nouns.

WordNet hyponym-hypernym relationships form a hierarchical structure. In Figure 8.1, we can see the hypernym hierarchy of the word *cat* (with the *feline* sense). The word *cat* belongs to the synset (*cat*, *true cat*) which has the synset (*feline*, *felid*) as hypernym, this synset has the synset (*carnivore*) as hypernym and so on until the final synset (*entity*). Finally, the different synsets form a tree structure whose root is the synset (*entity*). Sometimes, a synset has several distinct hypernyms, contrary to the example in Figure 8.1. For instance, the synset (*man*, *adult male*) has two hypernym synsets: (*male*, *male person*) and (*adult*, *grownup*). In this case, the hypernym hierarchy contains several branches which eventually converge to the root synset (*entity*).

depth from the root synset	synset
0	entity
1	physical entity
2	object, physical object
3	whole, unit
4	living thing, animate thing
5	organism, being
6	animal, animate being, beast, brute, creature, fauna
7	chordate
8	vertebrate, craniate
9	mammal, mammalian
10	placental, placental mammal, eutherian, eutherian mammal
11	carnivore
12	feline, felid
13	cat, true cat

**Figure 8.1.:** Hypernym hierarchy of the word *cat* (feline sense).

With the example of the hypernym tree for the word *cat* in Figure 8.1, we can already notice some particularities implied by the hierarchical structure: if we only consider the first hypernym in the tree – the synset (*feline, felid*) in this case – we can miss a lot of relevant information. In the case of *cat*, the only hypernym retrieved would be the synset (*feline, felid*) but we might also want to retrieve the information that a cat is an animal. Binding *cat* and *animal* in a hyponym-hypernym relationship would be possible but it would require to consider not only the first hypernym in the hierarchy but at least the first seven ones. This lets some questions naturally come up: how many hypernyms should we consider in the hierarchy? How should we proceed to figure it out? Is it relevant to fix a depth for the hypernyms search?

depth from the root synset	synset
0	entity
1	physical entity
2	object, physical object
3	whole, unit
4	artifact, artefact
5	instrumentality, instrumentation
6	furnishing
7	furniture, piece of furniture, article of furniture
8	table

**Figure 8.2.:** Hypernym hierarchy of the word *table* (furniture sense).

Let us presume that, based on the previously seen *cat-animal* relationship, we decide to look at the seven first hypernyms in the hierarchy. Figure 8.2 represents the hypernym hierarchy for the word *table* as a furniture. In this case, we see that the most relevant hypernym is likely to be the synset (*furniture, piece of furniture, article of furniture*), the first one in the hierarchy. Moreover,

---

the seventh hypernym in the hierarchy is the synset (*physical entity*). We are touching the problem of our method taking the seven first hypernyms in the WordNet tree: depending on the words for which we look for the hypernyms, the optimal depth to search in the tree varies a lot. We therefore cannot use this naive approach.

In order to fix the problem mentioned above, we explore two different strategies: the first one is to consider only the first hypernym in the hierarchy while in the second method we search hypernyms in the whole hierarchy but assign weights depending on their depth in the tree.

---

### 8.1.1 Searching only the first hypernym in the hierarchy

---

This strategy is quite simple: for each word in the topic, we get its direct hypernyms in the hierarchy and update the counts for these hypernyms. We do not search for hypernyms higher in the hierarchy. This insures us not to end up with hypernyms being too general because they are close to the root synset (*entity*). On the other hand, considering only the first hypernym in the hierarchy might make us miss hypernyms which are a little bit higher in the hypernyms tree.

---

### 8.1.2 Searching all hypernyms in the hierarchy and assigning weights

---

Contrary to the method described above, here we consider the whole hypernym hierarchy. In order not to get always the most general hypernyms as output, we need to penalize the hypernyms which are higher in the hierarchy, we decide to assign a weight  $w$  to each hypernym  $h$  that we encounter in the hierarchy: for a hypernym located  $n$  levels above the base word, we assign the weight  $\frac{1}{n}$ . With this weighting strategy, a hypernym from the second level (*carnivore* in Figure 8.1) needs to come up twice more than a first level (*feline* in Figure 8.1) to get the same score in the end, assuming that hypernyms occur at the same levels of the hierarchy.

---

### 8.1.3 Evaluation of WordNet hypernym annotation methods

---

In order to evaluate two methods introduced above, we first select 100 interpretable topics among our results. These topics were randomly sampled from the set of topics annotated as interpretable in Chapter 10 of this thesis. For each topic, we generate the three most relevant hypernyms according to each method. Then, we manually evaluate the relevance of these hypernyms by tagging them as relevant or not relevant. A relevant hypernym is defined as a hypernym that describes the majority of the words in the topic, without being too general e.g. the hypernym *person* is not relevant for a topic about tennis players. If at least one of the three hypernyms is considered relevant, the set of hypernyms for this topic is tagged as relevant.

With the first method, considering only direct hypernyms, 39 sets of hypernyms were tagged as relevant out of 100. In most of the cases where the hypernyms are not relevant, it has been observed that the topic words were mostly proper nouns, often without an entry in WordNet. With the second strategy, considering the whole hierarchy of hypernyms and assigning a weight penalizing more hypernyms, only 21 sets of hypernyms were judged relevant. Following these results, we decide to annotate our topics with hypernyms found with the first method.

---

## 8.2 Hypernyms from IS-A relation databases

---

As explained briefly in the section about the construction of the DDTs, IS-A relationships are computed during this process and a IS-A relationships database<sup>1</sup> is output by the system. A sample of this database is illustrated in Table 8.1. We can see that each IS-A relationship has a weight. This weight is a occurrence frequency value. The higher this value is, the more relevant the IS-A relationship should be. In this section, we try to use these IS-A relationships instead of the hypernyms coming from WordNet. Using this IS-A relationships database instead of WordNet permits us to be free of any lexical resource: we only use resources that are part of the output of the JoBimText pipeline. On the other hand, since this database is built with unsupervised methods, the relationships that it contains were not verified by humans. Thus, we have to take care of possible errors and adapt our methods to those.

hyponym	hypernym	freq.
ant	habitat	2
ant	health risk	2
ant	here	1
ant	hit	1
ant	home	5
ant	insect	41
ant	insect bite	1
ant	insect pest	1
ant	invertebrate	5
ant	java	1

**Table 8.1.:** Sample of the IS-A relationship database.

While the global idea is the same as with WordNet – for each topic, we look for hypernyms that describe it best – there are some differences. The structure of the database is flat: while WordNet hypernyms form a tree structure (hypernyms have hypernyms which also have hypernyms etc.) in this database all words are on the same level. As a matter of fact, some hypernyms are also present as hyponyms. Therefore, a tree structure is also present but abstracted. It would be possible to extract it such that we deal with the same structure as with WordNet, but this is not explored in this thesis. Instead, we decide to take the database as it is and adapt our methods to the flat structure.

---

### 8.2.1 Counting hypernyms with and without considering weights

---

First, we try to apply the method used previously with WordNet: for each element in a topic, we record its hypernyms and output the most frequent ones. We try two strategies: with the frequency values present in the database and without them. Without the frequency values, the method consists in counting how many times each hypernym appears within the whole topic word

---

<sup>1</sup> [http://panchenko.me/data/joint/taxi/res/en\\_ps59g.csv.gz](http://panchenko.me/data/joint/taxi/res/en_ps59g.csv.gz)

---



---

method used	without weight	with weight
relevant hypernyms (%) without TF-IDF	57	<b>63</b>
relevant hypernyms (%) with TF-IDF	76	<b>79</b>

---

**Table 8.2.:** Accuracy of methods with and without TF-IDF to annotate topics with hypernyms from the IS-A database.

senses IS-A relationships, the output is the hypernyms which appear the most often. When we use the frequency value, we add the value to the hypernym current count (instead of adding 1 in the previous case). For example, applying the first method without the frequency value on the word *ant* of Table 8.1 would make all its hypernyms equal, whereas with the second method, the hypernym *insect* would be considered as more relevant than the others.

---

### 8.2.2 Adapting TF-IDF scheme to discard noisy hypernyms

---

Furthermore, we notice that some hypernyms are very common in the IS-A database. The words *thing* or *product* are two of the most recurrent hypernyms. In the third method, we try to penalize these words. To do so, we apply the popular scheme TF-IDF [Salton and McGill, 1986]. TF-IDF stands for Term Frequency - Inverse Document Frequency. In the context of information retrieval in texts, TF-IDF allows to assess the importance of a term in a document of a corpus, relatively to the whole corpus. The principle is the following: a word which comes up very often in all documents in the corpus is less important than a word which comes up in only a few documents of the corpus. For a given word  $w$  in a document  $d_i$  of a corpus  $D$ , the term frequency is the count of occurrences of  $w$  in  $d_i$  and the inverse document frequency is defined as  $idf = \log \frac{|D|}{\{|d_j:w \in d_j\}}$

We apply the TF-IDF scheme to our case as follows: for each topic, we compute the most relevant hypernyms according to the method without TF-IDF i.e. just by counting the hypernyms, adjusting with their weight or not; we try both methods. For each topic, we keep the 100 hypernyms with the best scores. Each topic is one document, we therefore compute a IDF score for each hypernym. After that, we adapt the score of each hypernym for each topic by multiplying its frequency score by its IDF score. We sort hypernyms for each topic by TF-IDF and keep hypernyms with the three top scores.

---

### 8.2.3 Evaluation of methods to find relevant hypernyms from the IS-A database

---

As an evaluation for the several methods, we apply exactly the same procedure as for the hypernyms from WordNet. We therefore refer the reader to Section 8.1.3. for details.

The results are shown in Table 8.2. As we can see, using weights improves the global relevance of hypernyms, and so does the use of the TF-IDF scheme. As a consequence, we decide to annotate topics using the method with both the weights and the TF-IDF scheme.

---

## **Part IV.**

# **Intrinsic evaluation of the structured topics**

---

---

## 9 Introduction

One of the main challenges when building topics is the evaluation of their quality. Indeed, some topics may contain words which are not highly semantically related. For instance, a topic with words *lion*, *tiger*, *cat*, *jaguar*, *lynx* contains words that are all felines; we can say that the quality of this topic is good. On the contrary, a topic with words *window*, *computer*, *tree*, *Roma*, *Michael* is such that we cannot find coherence in the words it contains. Thus this topic would be a topic of bad quality.

A method to evaluate topics quality is needed for two reasons. First, in this thesis, we tried several models: several sources are used to build the DDT, filtered with four different frequency thresholds and two different methods are used to cluster the graph of word senses. We therefore need to evaluate each of those models. Then, inside each model, the topics quality is heterogeneous: we need to evaluate each topic and discard the bad ones.

Evaluating topics quality is not a simple task. Since the topics are generated by our model, there is no gold standard with which we could compare our results, each set of topics is unique. Several methods have been proposed to evaluate topic models such as in [Wallach et al., 2009]. However, these methods are in practice impossible to apply to our case, because of our specific topics. Indeed, the evaluation methods mostly rely on the fact that topics are defined as distributions over the whole vocabulary, which is not true for our topics.

In this master thesis, we utilize three different methods to assess the quality of our models. First, we manually evaluate the topics to establish and use the results to train a formula aiming at evaluating the topics quality based on the graph metrics. Secondly, we analyze the IS-A relationships for each word in a topic, following the idea that words in a good quality topic should share the same IS-A relationships. Third, we try to map our topics to BabelNet topics and analyze the results. Based on the results of these three evaluation experiments, we select the best model for our topics.



---

# 10 Interpretability of topics

In this first experiment, we manually annotate topics interpretability of a random sample of each topic model. We then compare the obtained results and deduce some insights about topics quality.

---

## 10.1 Experimental settings

---

First, we use manual annotation for evaluating the topics: for each of our models, we manually annotate some of the topics as interpretable or not. Here are the criteria applied to determine whether a topic is interpretable or not: first, if the meaning of a topic can be guessed easily just by looking at its words, because the annotator knows them and detects that they all match one topic then this topic is tagged as interpretable. If this is not the case because the words are unknown to the annotator, then we randomly sample until 10 of the words it contains and search the meaning of each one. After this, if the words match a same topic, it is tagged as interpretable, or tagged as not interpretable otherwise. That way, we get the number of interpretable topics for each model and thus a measure of the average interpretability of topics generated by each model.

Nevertheless, since there are not less than 32 different models to evaluate and each model contains a few hundred topics, annotating all of them would require a tremendous amount of time. To make this experiment feasible, while still relevant, we annotate 50 topics for each model. We sample these topics randomly.

---

## 10.2 Discussion of results

---

We present a sample of interpretable topics in Table 10.1. Three non interpretable topics are presented in Table 10.2. Overall results for this experiment are shown in Table 10.3. The fifth column, "ratio interp. topics" is an estimation of the percentage of interpretable topics for the given model. Since we annotated 50 topics, this value is simply the number of topics tagged as interpretable, multiplied by 2. The sixth column, "# interp. topics" is the estimation of the total number of interpretable topics in this model. It equals to the percentage of interpretable topics multiplied by the number of topics within the model. The number of interpretable topics is the value that we want to maximize eventually, but we would prefer to have a model giving a great proportion of interpretable topics.

In Table 10.3, we notice that there is a tradeoff between the number of topics in a model and the ratio of interpretable topics within this model. Indeed, the models based on the DDT with the most topics, wiki30, are also the ones with the worst ratio of interpretable topics. However, since they contain way more topics, the estimation of the overall number of interpretable topics is the greatest one. On the other hand, models based on the other DDTs contain less topics but the ratio

topic size	20 topic words (randomly sampled)
47	Gretzky, Beckham, Ronaldo, Kewell, free-kick, Carrick, Drogba, Rooney, Capello, Almunia, Bale, Zidane, Bellamy, Robben, Adebayor, Shevchenko, Persie, Anelka, Nistelrooy, Messi ...
250	pistol, razor, armor, bullet, axis, ax, shovel, shotgun, needle, firearm, shield, blade, pike, bent, spike, digging, drill, gunshot, Spear, torch ...
334	chemotherapy, fluke, milder, anesthesia, mole, transplant, catheter, diarrhea, handicap, fatigue, pain, heartbeat, appetite, tuberculosis, Pick, vomiting, cough, insomnia, breathing, headache ...

**Table 10.1.:** Example of three interpretable topics. Extracted from results of DDT news200, 2000 as frequency threshold and clustered with Chinese Whispers.

topic size	topic words
3	Tricia, mother, Mara
10	TEAM, mammal, person, Sport, Species, GROUP, animal, bitch, civilian, livestock, thing
13	Organization, Driver, Agency, Services, group, Reebok, Check, Service, company, member, Mix, Programs, Host

**Table 10.2.:** Example of three non interpretable topics. Extracted from results of DDT news200, 2000 as frequency threshold and clustered with Chinese Whispers.

of interpretable topics is greater – it almost reaches 50% with Chinese Whispers, wiki200 and 2000 as frequency threshold. Since these models contain much less topics than the model wiki30, the estimation for the total number of interpretable topics is smaller.

Based on the analysis above, we could split our set of models into two subsets: the first subset would contain the models based on the DDT wiki30 and the second would contain the models based on the three other DDTs. Models in the first subset contain a lot of topics but a large majority among them are of bad quality, whereas models in the second subset contain fewer topics but quality of these topics is globally better.

freq. threshold	0	500	1200	2000
avg. ratio interp. topics (%)	12.5	15.8	18	<b>25.3</b>
mediane ratio interp. topics (%)	13	13	19	<b>26</b>
avg. # interp. topics	<b>317.4</b>	268.4	144.3	143.1
mediane # interp. topics	<b>155</b>	101.5	100.5	121.5

**Table 10.4.:** Comparison of results of interpretability annotations for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold.

Regarding the frequency threshold used to filter the DDT before the clustering, Table 10.4 shows a comparison of the results. The ratio of interpretable topics is increased when the frequency threshold is increased, the best results are obtained with the threshold 2000. When we look at the

clust. method	DDT	freq. threshold	# topics	ratio interp. topics (%)	# interp. topics
CW	news200	0	595	14	83
CW	news200	500	359	<b>34</b>	88
CW	news200	1200	330	20	66
CW	news200	2000	286	<b>38</b>	109
CW	news50	0	835	18	150
CW	news50	500	668	18	120
CW	news50	1200	596	26	155
CW	news50	2000	547	<b>30</b>	164
CW	wiki200	0	838	22	184
CW	wiki200	500	539	18	97
CW	wiki200	1200	467	24	112
CW	wiki200	2000	433	<b>48</b>	208
CW	wiki30	0	14599	10	<u><b>1460</b></u>
CW	wiki30	500	8503	10	<b>850</b>
CW	wiki30	1200	6857	12	<b>823</b>
CW	wiki30	2000	5914	10	<b>591</b>
LM	news200	0	773	12	93
LM	news200	500	440	16	70
LM	news200	1200	375	24	90
LM	news200	2000	343	<b>28</b>	96
LM	news50	0	929	6	56
LM	news50	500	711	6	43
LM	news50	1200	619	18	111
LM	news50	2000	557	24	134
LM	wiki200	0	998	16	160
LM	wiki200	500	662	16	106
LM	wiki200	1200	559	14	78
LM	wiki200	2000	529	20	106
LM	wiki30	0	17651	2	353
LM	wiki30	500	9665	8	<b>773</b>
LM	wiki30	1200	7676	6	460
LM	wiki30	2000	6714	4	269

**Table 10.3.:** Results of interpretability annotations for the different models. The five best ones are highlighted, the best one is also underlined.

---

---

number of interpretable topics, the analysis is less easy. The average number of interpretable topics decreases when the threshold is increased, while the median values do not show a clear tendency. However, we have to keep in mind the huge number of interpretable topics for the models based on wiki30. These values seem to involve an important bias here.

clust. method	CW	LM
avg. ratio interp. topics (%)	<b>22</b>	13.8
mediane ratio interp. topics (%)	<b>19</b>	15
avg. # interp. topics	<b>328.8</b>	187.4
mediane # interp. topics	<b>152.5</b>	101

**Table 10.5.:** Comparison of results of interpretability annotations for Chinese Whispers and the Louvain Method: for each clustering method, average over all models based on this method.

Regarding the differences between the clustering methods, Chinese Whispers and the Louvain Method, Table 10.5 shows that Chinese Whispers seems to produce more interpretable topics than the Louvain Method, since all the metrics agree on this point.

---

# 11 Hypernym graph analysis

In this second experiment, we analyze topics coherence by building a graph based on hypernym co-occurrences among topic words. We compute the global clustering coefficient for each graph.

---

## 11.1 Experimental settings

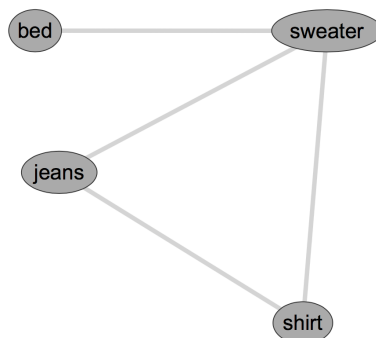
---

In this part, we describe the second method that we use to evaluate the topics models quality. The idea of the method is the following: for each topic, we build a new graph in which the nodes are the word senses of the topic, but the edges are drawn according to node hypernyms. For each node we generate a set of hypernyms of the word associated to this node and we add an edge between two nodes if the two associated words share at least one hypernym. The graph is therefore undirected and the edges are unweighted.

Table 11.1 illustrates how we build the graph of hypernyms based on a simple example. In this example, we consider a small topic made of four words: *bed*, *jeans*, *shirt* and *sweater*. Each of these words has its own list of hypernyms, that are represented in Table 11.1. The words *bed* and *sweater* both share the hypernym *item*, thus an edge is drawn between the two nodes. The words *sweater*, *jeans* and *shirt* all have *clothing* as hypernym, they are therefore connected together.

word	bed	jeans	shirt	sweater
hypernyms	accomodation equipment item	clothes clothing basic	clothes clothing gift	item clothing accessory

**Table 11.1.:** Hypernyms for each word of the demo topic.



**Figure 11.1.:** Hypernyms graph of the demo topic.

topic size	clust. coef. of hypernym graph	20 topic words (randomly sampled)
66	0.98	virtuoso, percussion, harmonica, cellist, fiddle, drumbeat, guitarist, pianist, trumpeter, bass, bassist, saxophonist, typewriter, Baroque, banjo, drummer, drum, whistle, jukebox, bell ...
45	0.86	mousse, biscuit, potato, cheese, pasta, sauce, waffle, sausage, dish, noodle, steak, crust, fruit, fries, toast, burger, Benedict, pudding, cookie, pastry ...
128	0.42	roulette, heroin, passport, firework, needle, booze, adrenaline, inhibitor, Taser, contraceptive, formulation, placebo, capsule, bullion, Pot, medicine, Tobacco, bandage, slave, vaccine ...

**Table 11.2.:** Example of three topics with a high clustering coefficient of their hypernym graph. Extracted from results of DDT news50, 2000 as frequency threshold and clustered with Chinese Whispers.

With this graph we can evaluate the coherence of words inside the topic. The assumption is that a topic is more likely to be of good quality if the words within this topic share many hypernyms with each other. In Figure 11.1, the words *jeans*, *shirt* and *sweater* are connected because they all denote a kind of clothes.

In order to acknowledge how connected the resulting graph is, we compute its global clustering coefficient. The clustering coefficient is a metric based on triplets and triangles of nodes in the graph. A triplet is composed of three connected nodes  $a$ ,  $b$ ,  $c$  such that  $a$  is connected to  $b$  and  $b$  is connected to  $c$ . A triangle is a triplet which is closed, such that  $a$  is connected to  $b$ ,  $b$  is connected to  $c$  and  $c$  is connected to  $a$ . The global clustering coefficient  $C$  is defined as:

$$C = \frac{3 \times \text{number of triangles}}{\text{number of connected triplets}}$$

The more hypernyms the words in the topic share, the more edges (and triangles with them) will be present in the graph and the closer to 1 the clustering coefficient will be. Ideally, if all  $n$  words in the topic share at least one common hypernym, the graph built by our method will be the graph  $K_n$  and the clustering coefficient will equal to 1.

In the previous example Figure 11.1, there is one triangle and five triplets, therefore the clustering coefficient is  $C = \frac{3 \times 1}{5} = 0.6$ .

---

## 11.2 Discussion of results

---

A sample of the results is shown in Table 11.2. Overall results for this experiment are compiled in Table 11.3. Tables 11.4 11.5 and 11.6 show results regarding respectively the clustering method, the DDT and the frequency threshold used. In all figures, the metrics computed are the same:

clustering method	DDT	frequency threshold	# topics	ratio non zero elements (%)	avg. clust. coef. non zero elements	avg. clust. coef. all elements
CW	news200	0	595	25	0.09	0.02
CW	news200	500	359	<u>30</u>	0.21	0.06
CW	news200	1200	330	<u>30</u>	0.52	<b>0.16</b>
CW	news200	2000	286	<u>27</u>	0.61	<b>0.16</b>
CW	news50	0	835	26	0.12	0.03
CW	news50	500	668	26	0.35	0.09
CW	news50	1200	596	<u>27</u>	0.64	<b>0.17</b>
CW	news50	2000	547	<u>28</u>	<b>0.90</b>	<b>0.25</b>
CW	wiki200	0	838	23	0.08	0.02
CW	wiki200	500	539	22	0.31	0.07
CW	wiki200	1200	467	22	0.40	0.09
CW	wiki200	2000	433	20	0.56	0.11
CW	wiki30	0	14599	3	0.23	0.01
CW	wiki30	500	8503	3	0.56	0.02
CW	wiki30	1200	6857	3	<b>0.85</b>	0.03
CW	wiki30	2000	5914	3	<u>0.97</u>	0.03
LM	news200	0	773	18	0.07	0.01
LM	news200	500	440	25	0.39	0.10
LM	news200	1200	375	24	0.60	0.14
LM	news200	2000	343	24	0.72	<b>0.17</b>
LM	news50	0	929	20	0.12	0.02
LM	news50	500	711	18	0.38	0.07
LM	news50	1200	619	18	0.67	0.12
LM	news50	2000	557	20	<b>0.81</b>	<b>0.17</b>
LM	wiki200	0	998	16	0.08	0.01
LM	wiki200	500	662	16	0.37	0.06
LM	wiki200	1200	559	18	0.56	0.10
LM	wiki200	2000	529	18	0.69	0.13
LM	wiki30	0	17651	1	0.08	0.00
LM	wiki30	500	9665	2	0.47	0.01
LM	wiki30	1200	7676	2	0.69	0.01
LM	wiki30	2000	6714	2	<b>0.88</b>	0.02

**Table 11.3.:** Results of hypernym graphs analysis for the different models. The five best ones are highlighted, the best one is also underlined. "ratio non zero el." reflects the proportion of topics for which the clustering coefficient of the hypernym graph is different from zero. The next column "avg. clust. coef. non zero elements" is the average of the clustering coefficients of these topics. The last column "avg. clust. coef. all elements" is the average of the clustering coefficients for all hypernym graphs, including those with a clustering coefficient equal to zero. This the measure that we want to optimize.

metric	CW	LM
avg. ratio non zero el.	<b>0.20</b>	0.15
avg. clust. coef. non zero el.	0.42	<b>0.47</b>
avg. clust. coef. all elements	<b>0.08</b>	0.07

**Table 11.4.:** Comparison of results of hypernyms graph analysis for Chinese Whispers and the Louvain Method: for each clustering method, average over all models based on this method.

metric	news200	news50	wiki200	wiki30
avg. ratio non zero el.	<b>0.25</b>	0.23	0.19	0.02
avg. clust. coef. non zero el.	0.40	0.49	0.38	<b>0.53</b>
avg. clust. coef. all elements	0.10	<b>0.11</b>	0.07	0.01

**Table 11.5.:** Comparison of results of hypernyms graph analysis for the different DDTs: for each DDT, average over all models based on this DDT.

- the ratio of clustering coefficients not equal to zero among all of them (ratio non zero el.)
- the average of the clustering coefficients that are not equal to zero (avg. clust. coef. non zero el.)
- the average over all clustering coefficients including the ones equal to zero (avg. clust. coef. all el.)

We decided to use these three metrics because by looking at the raw results, we noticed that many clustering coefficients equal to zero. This is the case when no triangle is present in the graph, meaning that the topic is probably not coherent. However, values of the non zero clustering coefficients are distributed between 0 and 1, reflecting different levels of coherence from the least (clustering coefficient close to 0) to the most coherent (clustering coefficient close to 1). Therefore, the ratio of non zero clustering coefficients shows the proportion of certainly not interpretable topics, and the average score of the non zero values reflects the average level of interpretability of the rest of the topics. Finally, the average score over all values allows us to get an insight of the global average coherence of the topics.

Table 11.4 compares the results for the two methods: Chinese Whispers and the Louvain Method. There is no clear evidence about which method performs better here: the Louvain Method produces more topics getting a null clustering coefficient, but on the other hand, the topics that get

metric	0	500	1200	2000
avg. ratio non zero el.	0.16	0.17	0.18	<b>0.19</b>
avg. clust. coef. non zero el.	0.10	0.37	0.61	<b>0.74</b>
avg. clust. coef. all elements	0.01	0.05	0.10	<b>0.14</b>

**Table 11.6.:** Comparison of results of hypernyms graph analysis for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold.



---

a non zero clustering coefficient are globally more coherent than the ones produced with Chinese Whispers.

Table 11.5 compares the results with the point of view of the DDTs used. These results match the conclusion of the experiment with manual annotations of the interpretability of the topics: three DDTs (new200, news50 and wiki200) produce globally topics with the same ratio of clustering coefficient equal to zero, around 0.20 but with the last DDT, wiki30, the ratio of non zero clustering coefficients is only equal to 0.02, thus ten times smaller than for the other DDTs. A large proportion of the topics generated with the DDT wiki30 seems not to be coherent. The three other DDTs get globally the same results but we can notice some slight differences: wiki200 gets the worst results in terms of ratio and in terms of average score of non zero clustering coefficients. news200 and news50 get nearly the same average over all clustering coefficients but the two other metrics differ a little: news200 produces slightly more topics with the clustering coefficient equal to zero but, on the other hand, the average score of these topics is smaller than for news50.

Finally, Table 11.6 compares the results according to the frequency threshold used. Here again, the results match the conclusion of the first experiment with manual annotations of interpretability of the topics: the more DDTs word senses were filtered before clustering the graph, the better the quality of the topics is. This remark holds with respect to the three metrics we use in this experiment, increasing the frequency threshold makes the clustering method produce more coherent topics.

---

## 12 Mapping to BabelNet topics

In this the experiment, we compute the cosine similarity between our topics and topics from BabelNet [Navigli and Ponzetto, 2010] and analyze the similarity of the most similar BabelNet topics.

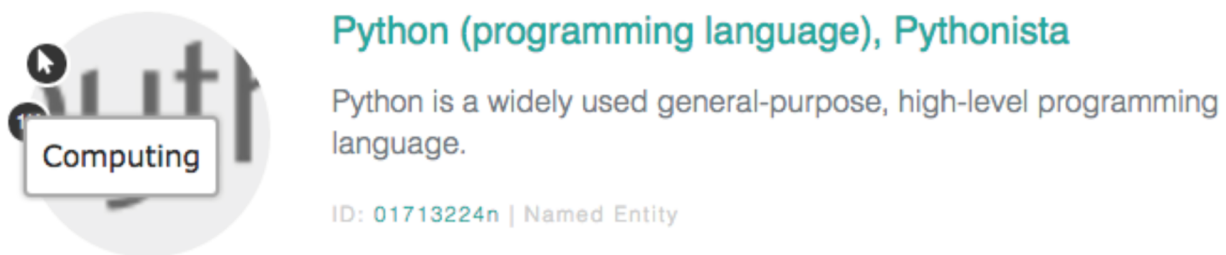
---

### 12.1 Experimental settings

---

In this third experiment, we make use of BabelNet [Navigli and Ponzetto, 2010]. BabelNet is a semantic network and an encyclopedic dictionary. It has been built upon WordNet and Wikipedia and realises a mapping between terms present in Wikipedia and WordNet synsets. BabelNet additionally provides some features such as machine translation. BabelNet has already been used for several applications such as word sense disambiguation [Ponzetto and Navigli, 2010] [Navigli et al., 2013] [Basile et al., 2014] and plagiarism detection [Franco-Salvador et al., 2013].

The key point regarding our experiment is that each synset in BabelNet belongs to a specific topic and therefore, seen from the opposite direction, BabelNet contains topics which are groups of synsets. There are many different topics such as *animals*, *media*, *music*, *technology* and so on. BabelNet topics are presented in Table 12.1. For instance, Figure 12.1 shows the entry for the word *Python* with the sense of the programming language. This entry belongs to the topic *computing*.



**Figure 12.1.:** Example of BabelNet synset for the word Python as a programming language. This synset belongs to the "computing" topic. Source: BabelNet.org.

In this experiment, we try to establish a mapping between our topics and BabelNet topics: for each of our topics, we find the BabelNet topic which is most similar to this topic. The similarity measure is computed using cosine similarity between the two topics.

Cosine similarity is a measure of similarity between two vectors in a given space. In the context of text documents, each term is a dimension of the space. Vector's coordinates are defined according to the occurrences of each term in the document. The angle between the vectors representing two documents is comprised between 0 and 90 degrees (vectors coordinates are non negative, since a

---

topic name	# synsets
Engineering and technology	6041
Language and Linguistics	12708
Warfare and Defense	50169
Food and Drink	10722
Physics and Astronomy	18875
Farming	6965
Education	33698
Politics and Government	30777
Textile and Clothing	5330
Mathematics	17172
Heraldry Honors and Vexillology	3761
Health and Medicine	22843
Art Architecture and Archaeology	25973
Geology and Geophysics	5977
Media	77826
Business Economics and Finance	8532
Law and Crime	16034
Animals	27953
Music	106610
Culture and Society	990
Numismatics and Currencies	2499
Meteorology	4451
Computing	20481
Religion Myticism and Mythology	25967
Philosophy and Psychology	10134
Geography and Places	54789
Royalty and Nobility	23477
Chemistry and Mineralogy	17052
Literature and Theater	27823
Biology	31229
Games and Video Games	16972
Transport and Travel	40121
Sport and Recreation	96288
History	8429

**Table 12.1.:** BabelNet topics names and number of synsets per topic.

topic size	most similar BabelNet topic	cosine similarity	20 topic words (randomly sampled)
867	food/drink	1.89E-4	mousse, biscuit, potato, cheese, pasta, sauce, waffle, sausage, dish, noodle, steak, crust, fruit, fries, toast, Benedict, burger, pudding, cookie, pastry, bread, cream, salad ...
104	politics/government	1.51E-4	voting, Votes, ballot, bloc, counting, front-runner, Santorum, Electoral, sweepstakes, hustings, proceeding, Elections, Vote, senate, balloting, candidature, judging, Pazz, Polling, Votes, Vote, e-mail ...
391	animals	0.86E-4	poodle, terrier, retriever, labrador, shepherd, aquarium, bacterium, fungus, moth, flea, cockroach, iceberg, spider, deer, organism, cricket, moose, raccoon, worm, elk, rabbit, coyote, parasite ...

**Table 12.2.:** Example of BabelNet mapping results for three topics. Extracted from results of DDT news50, 2000 as frequency threshold and clustered with Chinese Whispers.

word cannot occur a negative number of times). The cosine of this angle is the cosine similarity between the two documents: the closer to 0 the angle is, the closer to 1 the cosine is and the most similar are the two documents since they share most of their terms.

## 12.2 Discussion of results

After computing the cosine similarity between each pair of topics (our topic and all the BabelNet topics), we record how similar to our topic the most similar BabelNet topic is. A sample of the results for three topics is shown in Table 12.2. The overall results for this experiment are compiled in Table 12.3. The fourth column contains the average cluster size for each model, as a recall of Tables 7.1 and 7.3. Indeed, we believe that the cluster size plays a role in the results of this experiment.

Table 12.4 compares results obtained with Chinese Whispers and the Louvain Method. The two clustering methods provide comparable results, with a slight advantage for the Louvain Method.

method	CW	LM
avg. max cosine similarity	0.01226	<b>0.01363</b>

**Table 12.4.:** Comparison of results of experiment with BabelNet topics for the different clustering methods: for each DDT, average over all models based on this DDT.

Table 12.5 compares results obtained with the four DDTs. In this experiment, models built upon the DDT wiki30 get better results than the three others. Among them, models built with news200 obtain the worst results, news50 and wiki200 are get nearly the same results. If we refer to the cluster average size in each model, shown in Table 12.3, we notice that the score for this experiment is inversely proportional to the average cluster size: fine-grained models get better

clust. method	DDT	freq. thres.	avg. cluster size	avg. max cosine similarity
CW	news200	0	382.7	0.00074
CW	news200	500	311.4	0.00079
CW	news200	1200	208.5	0.00083
CW	news200	2000	170.8	0.00077
CW	news50	0	325.9	0.00145
CW	news50	500	258.0	0.00168
CW	news50	1200	184.9	0.00170
CW	news50	2000	145.8	0.00194
CW	wiki200	0	291.0	0.00169
CW	wiki200	500	189.9	0.00170
CW	wiki200	1200	142.3	0.00176
CW	wiki200	2000	113.5	0.00170
CW	wiki30	0	68.6	<b>0.04592</b>
CW	wiki30	500	50.1	<b>0.04406</b>
CW	wiki30	1200	40.4	0.04120
CW	wiki30	2000	34.8	0.03735
LM	news200	0	295.0	0.00109
LM	news200	500	254.4	0.00097
LM	news200	1200	183.8	0.00103
LM	news200	2000	142.8	0.00091
LM	news50	0	292.9	0.00164
LM	news50	500	242.4	0.00184
LM	news50	1200	178.0	0.00181
LM	news50	2000	143.2	0.00194
LM	wiki200	0	244.3	0.00204
LM	wiki200	500	154.6	0.00214
LM	wiki200	1200	118.9	0.00218
LM	wiki200	2000	92.9	0.00219
LM	wiki30	0	56.7	<b>0.05670</b>
LM	wiki30	500	44.1	<b>0.05096</b>
LM	wiki30	1200	36.1	<b>0.04747</b>
LM	wiki30	2000	30.7	0.04310

**Table 12.3.:** Overall results of experiment with BabelNet topics. The five best ones are highlighted, the best one is also underlined.

scores. This might be due to the fact that vectors built from bigger topics tend to have a greater angle to BabelNet topics, because they simply contain more terms and thus the number of their terms not included in BabelNet topics tends to increase, increasing the angle as a consequence.

DDT	news200	news50	wiki200	wiki30
avg. max cosine similarity	0.00089	0.00179	0.00193	<b>0.04585</b>

**Table 12.5.:** Comparison of results of experiment with BabelNet topics for the different DDTs: for each clustering method, average over all models based on this method.

Regarding the frequency threshold used to filter the DDTs before clustering, results are shown in Table 12.6. They show that in average, decreasing this threshold gives better results. However, looking back at Table 12.3, we notice that this is true only for the models built upon the DDT wiki30. For the other models, the tendency is the opposite.

frequency threshold	0	500	1200	2000
avg. max cosine similarity	<b>0.01569</b>	0.01302	0.01225	0.01124

**Table 12.6.:** Comparison of results of experiment with BabelNet topics for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold.

According to this experiment, the best topic models are the models based on the DDT wiki30, 0 as frequency threshold. Chinese Whispers and the Louvain Method provide comparable results.

---

## 13 Conclusion

We have performed three different experiments to compare the quality of our topic models. In the first experiment, we inspected a random sample of topics from each model and manually annotated these topics as interpretable or not. In the second experiment, we used a hypernyms database to build a graph for each topic in which an edge is drawn between two topics terms if they share a common hypernym. We then analyzed the structure of the produced graph by computing its global clustering coefficient. In the third experiment, we used topics of BabelNet synsets and tried to map each of our topics to the most similar BabelNet topic. We then compared the cosine similarities between our topics and the most similar BabelNet topic.

Overall results of the three experiments are compiled in Table 13.1. We selected the most representative measure for each experiment:

- the ratio of interpretable topics according to the first experiment
- the average clustering coefficients of all topics per model, according to the second experiment
- the average of the maximal cosine similarities to BabelNet topics, according to the third experiment

Regarding the method used to cluster the graph, the three experiments do not fully agree on which model gives the best results: according to the first experiment, it is Chinese Whispers, in the second and the third experiments they are considered equal. Nevertheless, taking into account the three experiments, we can conclude that Chinese Whispers has a slight advantage over the Louvain Method in our use case.

Regarding the frequency threshold used to filter the graph before clustering, the two first experiments agree on the point that the best models are produced with the greatest threshold. In the last experiment, it is not the case, especially for the models built using the DDT wiki30, for which the result is the opposite. For the other models, there is no clear tendency. As a consequence, according to the first and second experiments, we conclude that greater frequency thresholds produce better quality topics.

Regarding the DDT used as input of our system, the first and second experiments agree with each other but the third experiment gives other conclusions. According to the first two experiments, models based on the DDTs news50, news200, wiki200 give better quality topics than the ones based on wiki30. However, the third experiment gives opposite conclusions. Among the three DDTs news50, news200, wiki200, they obtain similar results in the first and second experiments. wiki200 gives slightly better results in the first experiment while news50 and news200 models seem a bit better in the second experiment. Considering these results, it is difficult to acknowledge which model gives the best results.

Finally, we have noticed that coarsed-grained models perform better according to the two first experiments, but fine-grained topics (models based on DDT wiki30) perform better regarding the experiment with BabelNet topics.

clustering method	DDT	frequency threshold	# of topics	ratio interp. topics (%)	avg. clust. coef. all elements	avg. max cosine similarity
CW	news200	0	595	14	0.02	0.00074
CW	news200	500	359	<b>34</b>	0.06	0.00079
CW	news200	1200	330	20	<b>0.16</b>	0.00083
CW	news200	2000	286	<b>38</b>	<b>0.16</b>	0.00077
CW	news50	0	835	18	0.03	0.00145
CW	news50	500	668	18	0.09	0.00168
CW	news50	1200	596	26	<b>0.17</b>	0.00170
CW	news50	2000	547	<b>30</b>	<u><b>0.25</b></u>	0.00194
CW	wiki200	0	838	22	0.02	0.00169
CW	wiki200	500	539	18	0.07	0.00170
CW	wiki200	1200	467	24	0.09	0.00176
CW	wiki200	2000	433	<b>48</b>	0.11	0.00170
CW	wiki30	0	14599	10	0.01	<b>0.04592</b>
CW	wiki30	500	8503	10	0.02	<b>0.04406</b>
CW	wiki30	1200	6857	12	0.03	0.04120
CW	wiki30	2000	5914	10	0.03	0.03735
LM	news200	0	773	12	0.01	0.00109
LM	news200	500	440	16	0.10	0.00097
LM	news200	1200	375	24	0.14	0.00103
LM	news200	2000	343	<b>28</b>	<b>0.17</b>	0.00091
LM	news50	0	929	6	0.02	0.00164
LM	news50	500	711	6	0.07	0.00184
LM	news50	1200	619	18	0.12	0.00181
LM	news50	2000	557	24	<b>0.17</b>	0.00194
LM	wiki200	0	998	16	0.01	0.00204
LM	wiki200	500	662	16	0.06	0.00214
LM	wiki200	1200	559	14	0.10	0.00218
LM	wiki200	2000	529	20	0.13	0.00219
LM	wiki30	0	17651	2	0.01	<u><b>0.05670</b></u>
LM	wiki30	500	9665	8	0.01	<b>0.05096</b>
LM	wiki30	1200	7676	6	0.01	<b>0.04747</b>
LM	wiki30	2000	6714	4	0.02	0.04310

**Table 13.1.:** Comparison results of the three experiments we conducted. The five best ones are highlighted, the best one is also underlined.



---

## **Part V.**

# **Application of the structured topics to text categorization**

---

---

# 14 Visualization of topics

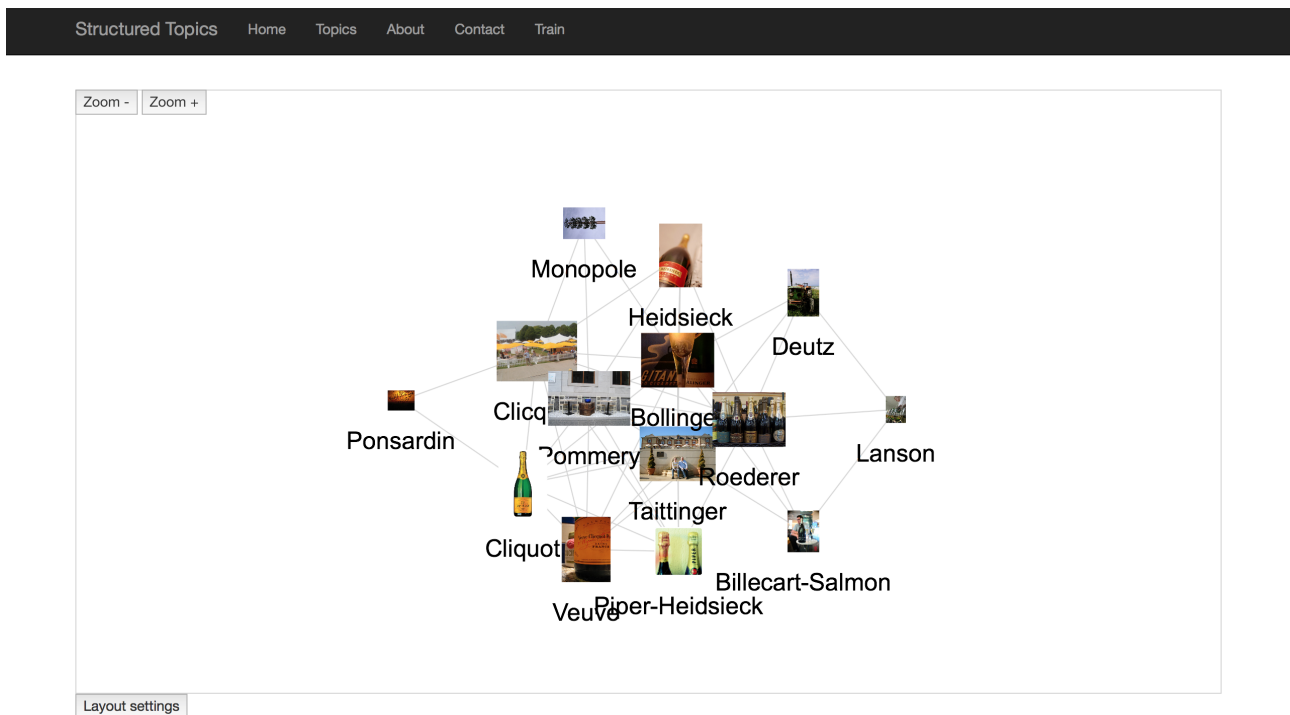
---

## 14.1 Introduction

---

Finally, we introduce a web application which allows the user to interact with our topic models. In particular, possibility is offered to visualize topics as bags of words or as graphs and to search for a text best matching topics. This application is composed of several entities that we will explain in the following. It is implemented using Java EE, the Bootstrap<sup>1</sup> framework and the vis.js library<sup>2</sup>.

As part of this web application, there is the possibility to visualize each topic's graph. More precisely, the subgraph induced by the topic words from the original DDT graph is displayed. This visualization is implemented with a vis.js network. An example can be seen in Figure 14.1.



**Figure 14.1.:** Graph visualisation for a topic about wine/champagne.

---

## 14.2 Nodes positions

---

<sup>1</sup> <http://getbootstrap.com>

<sup>2</sup> <http://visjs.org>

---

---

### 14.2.1 Positioning algorithm

---

The graph informations are obtained from the original graph and the nodes are located such that highly connected groups of nodes will be close of each other. Consequently, nodes with a great centrality value will be located in the center of this subgraph.

---

### 14.2.2 Storing positions

---

The positions need to be computed from scratch by the vis.js algorithm<sup>3</sup>: it can take up to several minutes for big topics, resulting in a bad usability for users. To avoid such an issue, nodes positions can be stored once the positioning algorithm has come to an end or the positions sufficiently reflect the network structure. This action is done using the button *Store Positions* just over the vis.js network container.

Once the positions for a given topic have been stored, this topic network will be directly loaded with those positions, saving the positions calculation time. A second button *Reset Positions* is also present to allow a user to remove former stored positions for the topic. In this case, the algorithm is run again to calculate new nodes positions.

A script was written to automate the process of loading a topic graph, waiting for its stabilization and then storing the positions. It uses PhantomJS<sup>4</sup> to perform all these actions without the need of a browser.

---

## 14.3 Images

---

Images are assigned for each node in the topic and pictured along the words in the graph visualization. Images allow the user to understand what the topic is about faster and easier than by reading topic words.

---

### 14.3.1 Images sources

---

The images come from three different sources. The first source database is the Serelex database [Panchenko et al., 2013]. Serelex<sup>5</sup> is a visualisation tool for semantically related terms. Actually, it is more or less a thesaurus visualisation tool: we can search for one word and get a network visualisation of this term and its most similar words, with all the connections between them. Each word is represented by an image. For each word in our topics, we look in the Serelex images database and get the image for this word. However, some words do not have an image in this database. In this case, DBpedia [Auer et al., 2007] is queried. DBpedia is a database containing structured information from Wikipedia. Among the information extracted from each article is the main image of this article, that is displayed at the top right of the article. If a Wikipedia article


---

<sup>3</sup> <http://visjs.org/docs/network/physics.html>

<sup>4</sup> <http://phantomjs.org>

<sup>5</sup> <http://serelex.cental.be>

---



---

title matches the word, its image is retrieved. Nevertheless, it can happen that neither Serelex nor DBpedia provide an image for a word. As a last resort, an image is searched in Flickr<sup>6</sup>, a pictures hosting website.

---

### 14.3.2 Images size

---

The images size varies with the node centrality in the topic. The greater the centrality value, the bigger the image. This way, words which are more connected, and thus more important in the network, are brought forward. The centrality involved here is the same as the centrality used in Section 7.5 to compute the centroid of each cluster. We therefore refer the reader to this part of the thesis for its definition.

---

<sup>6</sup> <http://flickr.com>

---

# 15 Topics exploration

---

## 15.1 General presentation

---

In this master thesis we tried several models to generate the topics. For each model, the user can visualize the list of all topics generated. Figure 15.1 shows an example. For each topic, the 50 most significant words are displayed as well as the topic id, the total number of elements within this topic, hypernyms and IS-A relationships to help grasp what the topic is about. Each entry in the list also contains two links: one to the graph visualization of this topic and another one to a page in which all terms in the topic are displayed, this page can be seen in Figure 15.2. The topics can be sorted by topic size, by topic score or by topic id, randomly assigned during the topic computation pipeline. For each criteria, the topics can be sorted from the largest to the smallest or the other way around.

---

## 15.2 Sorting topics by quality

---

Among the criteria available to sort topics in the list of all topics described above, there is the possibility to sort them by score. This score is the result of a measure that we developed in order to globally sort the topics by quality. It uses the topic size and the clustering coefficient of the topic subgraph. Its goal is to easily retrieve topics which neither too small nor too big and that are of good quality. The clustering coefficient of the topic subgraph gives information on the latter part. The score  $S$  is computed as follows:

$$S = C \times \exp^{-(n-N)^2}$$

where  $C$  is the clustering coefficient of the topic subgraph,  $N$  the average topic size of the model and  $n$  is the topic size

We did not formally prove that a high score  $S$  means that the quality of the topic is good, however, we noticed that this tends to be the case.

Browse all topics

Select a model: Single word Expressions Topics (385?)  
 Sorted by: Score + to - OK

**Topic id:** 153 [Full topic](#) [Graphical view](#)

**Topic size:** 9179

**50 topic words by frequency:** Smith, Moore, Johnson, Miller, Davis, Jones, Williams, Walker, Brown, Thompson, Anderson, Harris, Wilson, Robinson, Taylor, Wright, Evans, Jackson, Griffin, Murphy, Baker, Bailey, Bennett, Coleman, Sanders, Hayes, Henderson, Lewis, Watson, Peterson, Jenkins, Turner, Clark, Carter, Phillips, Parker, Richardson, Robertson, Dixon, Tucker, Patterson, Sullivan, Reynolds, Allen, Adams, Reed, Edwards, Roberson, Mitchell, Barnes

**Hypernyms:** 2, C, MD

**IsA relationships:** music(91), said(62), band(153), style(77), subject(106), answer(79), value(85), artist(135), act(68), entertainment (54)

**Topic id:** 213 [Full topic](#) [Graphical view](#)

**Topic size:** 5989

**50 topic words by frequency:** cheese, bean, potato, vegetable, tomato, rice, bread, bacon, onion, sausage, pea, sauce, mushroom, pepper, spinach, shrimp, salad, ham, chicken, butter, fruit, egg, carrot, apple, broth, cream, meat, juice, chili, noodle, soup, pork, cabbage, pasta, beef, asparagus, lettuce, couscous, cauliflower, nut, cake, eggplant, strawberry, banana, polenta, vinegar, cucumber, milk, ginger, pesto

**Hypernyms:** cast, cell, good

**IsA relationships:** scully(5), sport(42), way(5), game(15), sockettimeoutexception(7), event(25), item(4), weather(8), activity(23), option(6)

Figure 15.1.: Topic list.

Marcus is cooking chicken with rice

Topic: [Score = 0.08433662] [Graphical view](#)

**Cluster words by frequency:** cheese, bean, potato, vegetable, tomato, rice, bread, bacon, onion, sausage, pea, sauce, mushroom, pepper, spinach, shrimp, salad, ham, chicken, butter, fruit, egg, carrot, apple, broth, cream, meat, juice, chili, noodle, soup, pork, cabbage, pasta, beef, asparagus, lettuce, couscous, cauliflower, nut, cake, eggplant, strawberry, banana, polenta, vinegar, cucumber, milk, ginger, pesto, yogurt, chutney, confit, seafood, steak, corn, salmon, lobster, slaw, celery, berry, slice, prosciutto, tuna, honey, herb, chocolate, chive, oyster, artichoke, sandwich, scallop, peanut, saut, pie, spice, scallion, mustard, almond, pear, tofu, fries, zucchini, pizza, pancake, Parmesan, fennel, shallot, curry, squash, ricotta, salt, clam, chickpea, lentil, chorizo, stew, grits, pineapple, pudding, lamb, macaroni, leek, salsa, chily, sugar, syrup, turnip, broccoli, mayonnaise, coriander, yam, plantain, lasagna, burger, mint, avocado, arugulum, cumin, pistachio, walnut, coffee, vinaigrette, beet, pate, basil, puree, saffron, radish, mozzarella, turkey, raisin, dessert, peach, veggy, dish, meatloaf, gnocchus, okra, tea, hazelnut, paprika, pastry, cilantro, calamarius, prawn, flour, anchovy, meatball, tartare, cereal, oatmeal, pickle, mussel, peppercorn, parsley, sardine, parmesan, hummus, toast, pancetta, thyme, crouton, oregano, dumpling, roast, carpaccio, apricot, gratin, appetizer, chowder, pumpkin, truffle, kebab, coconut, cracker, mignon, terrine, risotto, venison, omelet, cookie, grapefruit, custard, dill, rosemary, brisket, mango, biscuit, ravioli, bisque, salami, gravy, fish, soda, rind, flan, rib, watercress, gras, lime, hamburger, cranberry, currant, papaya, fig, spaghetti, coleslaw, leaf, marmalade, cheddar, ragout, ketchup, sauerkraut, horseradish, margarine, chile, tarragon, buttermilk, crumb, linguine, jus, swordfish, veal, endive, raspberry, quiche, cashew, oats, soy, compote, powder, molasses, tenderloin, clove, sweetbread, feta, wine, waffle, marshmallow, rhubarb, zest, souffle, pur, caviar, sirloin, gumbo, tortilla, watermelon, kale, aiolus, flake, melon, pecan, fillet, anise, casserole, roe, curd, sherry, loin, seaweed, nutmeg, guacamole, caper, tamale, cinnamon, scone, yolk, stir-fry, seasoning, applesauce, barley, filet, patty, chard, chestnut, mousse, radicchio, lemonade, nacho, cutlet, enchilada, cider, tempura, baguette, shellfish, blueberry, squid, pomegranate, taco, cobbler, cod, sage, chanterelle, brie, cheeseburger, pretzel, mayo, vanilla, crust, shank, filling, empanada, jelly, cocoa, strudel, choy, bun, loaf, lemongrass, burrito, sushi, grape, pilaf, cornmeal, candy, sashimus, citrus, stuffing, croquette, fraiche, halibut, dough, cheesecake, oil, pepperoni, crabmeat, quesadilla, gelatin, morel, tablespoon, cardamom, masalum, cornstarch, shiitake, cob, quesoadilla, Gorgonzola, granolum, chillus, porcinius, parsnip, cornbread, marinade, brulee, miso, muffin, fettuccine, starch, crepe, wasabus, plum, bagel, vert, cayenne, popcorn, flavoring, espresso, antipasto, kimchus, grouper, wheat, brioche, entree, portobello, bratwurst, porridge, escarole, teaspoon, tartar, meringue, mascarpone, pita, escargot, teriyakus, cube, cheese, paellum, fajita, lard, ham, couli, ceviche, vermicelli, quinoa, cantaloupe, yoghurt, romaine, savory, jicama, verde, jam, brownie, bruschetta, penne, quail, tomatillo, mutton, edamame, rye, cappuccino, tripe, rabe, collard, monkfish, hominy, dressing, breast, orzo, mesclun, smoothie, barbecue, daikon, crackling, crostinus, bulgur, gazpacho, provolone, tapenade, prune, octopus, olive, matzo, breadcrumb, beer, dal, pimento, tortellinus, scampi, marinara, oxtail, lima, dressing, mein, pastrami, sorbet, gorgonzolum, tempeh, shaving, tapioca, homemade, malt, sauce, focaccium, beetroot, mahi-mahus, bologna, seed, jalapeno, fried, nectarine, poblano, tomatoes, hotdog, kasha, pecorino, charcuterie, marsalum, chevre, fondue, spicy, frite, tabbouleh, mash, paneer, fontina, veg, sorrel, sprout, chicory, pimiento, sprig, doughnut, wonton, blini, jalapeno, kraut, rutabaga, bouillon, Spam, adobo, garlic, giblets, roquefort, farro, kielbasa, Brie, jambalaya, flatbread, buttery, tequila, hock, cooked, chilli, bran, lingonberry, masa, biscotto, hash, baked, floret, yuca, napoleon, fritter, goulash, codfish, broccolinus, fava, capona, che, garbanzo, Marsala, sundae, cola, cornflake, spaetzle

Figure 15.2.: View of a topic with all words.

---

# 16 Text categorization

---

## 16.1 Front-end

---

Another entity present in the web application lets the user input a text and find the most similar topics to this text. The user can select the model that he wants to use. Figure 16.1 shows the search page, in which the user entered a text about Python, the programming language.

After selecting a model and submitting a piece of text, a results page is presented to the user, it contains the top 3 topics in the specified model given the text. For each of them, topic words, topic hypernyms from WordNet and topic IS-A relationships are displayed, as well as a list of matches with the submitted text words and a link to the topic visualization. An example is shown in Figure 16.2.

---

## 16.2 Back-end: a search engine

---

---

### 16.2.1 Introduction

---

We want to use our models to tag any text with relevant topics. To perform the tagging process, we create an index with all the topics and perform searches on it. For each query, the text is analyzed and a matching process between the analyzed text and the topics is performed. The topics receive a score between 0 and 1. The higher the score is, the more relevant the topic is. For the implementation, we use an ElasticSearch<sup>1</sup> index.

---

<sup>1</sup> <http://elastic.co>

The screenshot shows a web interface for searching topics. At the top is a dark navigation bar with links: 'Structured Topics', 'Home', 'Topics', 'About', 'Contact', and 'Train'. Below the navigation bar is a white box with the text 'Enter a text and look for its topics!'. Underneath is a large text area containing three paragraphs of text about Python. Below the text area is a dropdown menu showing 'Single word Expressions Topics (385?)' and a 'Submit' button.

Figure 16.1.: View of the search home page.

Marcus is cooking chicken with rice

Topic: [Score = 0.08433662] [Graphical view](#)  
 Cluster words by frequency: cheese, bean, potato, vegetable, tomato, rice, bread, bacon, onion, sausage, pea, sauce, mushroom, pepper, spinach, shrimp, salad, ham, chicken, butter, fruit, egg, carrot, apple, broth, cream, meat, juice, chili, noodle, soup, pork, cabbage, pasta, beef, asparagus, lettuce, couscous, cauliflower, nut, cake, eggplant, strawberry, banana, polenta, vinegar, cucumber, milk, ginger, pesto, yogurt, c hutney, confit, seafood, steak, corn, salmon, lobster, slaw, celery, berry, slice, prosciutto, tuna, honey, herb, chocolate, chive, oyster, ar tichoke, sandwich, scallop, peanut, saut, pie, spice, scallion, mustard, almond, pear, tofu, fries, zucchini, pizza, pancake, Parmesan, fennel , shallot, curry, squash, ricotta, salt, clam, chickpea, lentil, chorizo, stew, grits, pineapple, pudding, lamb, macaroni, leek, salsa, chily, sugar, syrup, turnip, broccoli, mayonnaise, coriander, yam, plantain, lasagna, burger, mint, avocado, arugulum, cumin, pistachio, walnut, cof fee, vinaigrette, beet, pate, basil, puree, saffron, radish, mozzarella, turkey, raisin, dessert, peach, veggy, dish, meatloaf, gnocchus, okr a, tea, hazelnut, paprika, pastry, cilantro, calamarius, prawn, flour, anchovy, meatball, tartare, cereal, oatmeal, pickle, mussel, peppercorn, parsley, sardine, parmesan, hummus, toast, pancetta, thyme, crouton, oregano, dumpling, roast, carpaccio, apricot, gratin, appetizer, chowder , pumpkin, truffle, kebab, coconut, cracker, mignon, terrine, risotto, venison, omelet, cookie, grapefruit, custard, dill, rosemary, brisket, mango, biscuit, ravioli, bisque, salami, gravy, fish, soda, rind, flan, rib, watercress, gras, lime, hamburger, cranberry, currant, papaya, fi g, spaghetti, coleslaw, leaf, marmalade, cheddar, ragout, ketchup, sauerkraut, horseradish, margarine, chile, tarragon, buttermilk, crumb, lin guine, jus, swordfish, veal, endive, raspberry, quiche, cashew, oats, soy, compote, powder, molasses, tenderloin, clove, sweetbread, feta, win e, waffle, marshmallow, rhubarb, zest, souffle, pur, caviar, sirloin, gumbo, tortilla, watermelon, kale, aiolus, flake, melon, pecan, fillet, anise, casserole, roe, curd, sherry, loin, seaweed, nutmeg, guacamole, caper, tamale, cinnamon, scone, yolk, stir-fry, seasoning, applesauce, barley, filet, patty, chard, chestnut, mousse, radicchio, lemonade, nacho, cutlet, enchilada, cider, tempura, baguette, shellfish, blueberry, squid, pomegranate, taco, cobbler, cod, sage, chanterelle, brie, cheeseburger, pretzel, mayo, vanilla, crust, shank, filling, empanada, jelly, cocoa, strudel, choy, bun, loaf, lemongrass, burrito, sushi, grape, pilaf, cornmeal, candy, sashimus, citrus, stuffing, croquette, fraiche, h alibut, dough, cheesecake, oil, pepperoni, crabmeat, quesadillum, gelatin, morel, tablespoon, cardamom, masalum, cornstarch, shiitake, cob, qu esadilla, Gorgonzola, granolum, chillus, porcinus, parsnip, cornbread, marinade, brulee, miso, muffin, fettuccine, starch, crepe, wasabus, plu m, bagel, vert, cayenne, popcorn, flavoring, espresso, antipasto, kimchus, grouper, wheat, brioche, entree, portobello, bratwurst, porridge, e scarole, teaspoon, tartar, meringue, mascarpone, pita, escargot, teriyakus, cube, cheese, paellum, fajita, lard, ham, couli, ceviche, vermicel li, quinoa, cantaloupe, yoghurt, romaine, savory, jicama, verde, jam, brownie, bruschetta, penne, quail, tomatillo, mutton, edamame, rye, capp uccino, tripe, rabe, collard, monkfish, hominy, dressing, breast, orzo, mesclun, smoothie, barbecue, daikon, crackling, crostinus, bulgur, gaz pacho, provolone, tapenade, prune, octopus, olive, matzo, breadcrumb, beer, dal, pimento, tortellinus, scampi, marinara, oxtail, lima, dressin g, mein, pastrami, sorbet, gorgonzolum, tempeh, shaving, tapioca, homemade, malt, sauce, focaccium, beetroot, mahi-mahus, bologna, seed, jalap e, fried, nectarine, poblano, tomatoes, hotdog, kasha, pecorino, charcuterie, marsalum, chevre, fondue, spicy, frite, tabbouleh, mash, paneer, fontina, veg, sorrel, sprout, chicory, pimienta, sprig, doughnut, wonton, blini, jalapeno, kraut, rutabaga, bouillon, Spam, adobo, garlic, gi blets, roquefort, farro, kielbasa, Brie, jambalaya, flatbread, buttery, tequila, hock, cooked, chilli, bran, lingonberry, masa, biscotto, hash , baked, floret, yuca, napoleon, fritter, goulash, codfish, broccolinus, fava, capona, che, garbanzo, Marsala, sundae, cola, cornflake, spaetz

Figure 16.2.: View of the search results page for the text "Marcus is cooking chicken with rice."

In order to test the search method, we run a first experiment using only 7 topics in the index. These topics are about fishes, medical drugs, professional cyclists, dogs, theater characters, famous navigators and wine, they were selected manually. The goal of this experiment is to make sure that the matching process is correctly performed and gives relevant results. We decided to use few topics to keep a reasonable size for the process.

The principle of this experiment is the following: we want to query this 7-topics-index with texts about each of these topics. The texts we submit are extracted from Wikipedia: we select articles from lists of articles included in wikipedia e.g. *List of common fish names* for the topic about fishes. We use the dump *enwiki-20160113-pages-articles-multistream.xml.bz2* to get those articles. We submit only the abstract of each article, so that all texts have a comparable size. Metrics for this set of text are outlined in Table 16.1.

topic	# texts	avg. texts length
drugs	390	931
fishes	1079	751
theater	217	978
explorers	458	869
cyclists	208	791
wine	576	637
dog breeds	502	570

Table 16.1.: Metrics of test texts for search engine.



---

---

## 16.2.2 nDCG

---

To get a better insight of the results quality, we use the normalized discounted cumulative gain measure (nDCG) with the logarithm base 2.

For a topic  $j$ ,  $nDCG_j = \frac{DCG_j}{IdealDCG_j}$ :

The  $nDCG_j$  value is between 0 and 1,  $nDCG_j = 0$  means that no article about the topic  $j$  has the topic  $j$  in its results (for the articles about fishes, none of them has the topic fish as a result),  $nDCG_j = 1$  means that all the articles about the topic  $j$  have the right topic  $j$  as first or second result (for the articles about fishes, all of them have the topic fish as first or second result).

---

## 16.2.3 Baseline results with naive random ranking

---

In order to have a baseline, we consider the case where the search engine returns the topics ranked in a random order. Statistically, we should get a uniform distribution over the 7 topics. The result does not depend on the topic or the number of articles but let us consider the topic 3 about theater as an example. For this topic, we have 217 texts. Since each text will get the 7 topics in a random order, each topic should get around  $217/7=31$  topics for each ranking position. This is illustrated in Table 16.2.

topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	31	31	31	31	31	31	31
2nd result	31	31	31	31	31	31	31
3rd result	31	31	31	31	31	31	31
4th result	31	31	31	31	31	31	31
5th result	31	31	31	31	31	31	31
6th result	31	31	31	31	31	31	31
7th result	31	31	31	31	31	31	31

**Table 16.2.:** Results for 7 topics randomly distributed for each text.

The nDCG measure is computed as follows:

$$DCG_1 = 31 + \frac{31}{\log_2 2} + \frac{31}{\log_2 3} + \frac{31}{\log_2 4} + \frac{31}{\log_2 5} + \frac{31}{\log_2 6} + \frac{31}{\log_2 7}$$

$$DCG_1 = 31 \times \left(1 + \frac{1}{\log_2 2} + \frac{1}{\log_2 3} + \frac{1}{\log_2 4} + \frac{1}{\log_2 5} + \frac{1}{\log_2 6} + \frac{1}{\log_2 7}\right)$$

$$DCG_1 = 31 \times 4.30$$

$$IDCG_1 = 217 = 31 \times 7$$

$$nDGC_1 = \frac{DCG_1}{IDCG_1} = \frac{4.30}{7} = 0.61$$

For 7 articles, the baseline nDCG score is 0,61.

---

#### 16.2.4 Simple TF-IDF query

---

First, we try a simple matching query: each article text is tokenized using an English tokenizer and English stopwords are filtered out. Each resulting token is lowercased and then compared to the words in each topic. A score between 0 and 1 is returned for each topic using TF-IDF [Salton and McGill, 1986] and the topics are ranked according to this score. In the following, we refer to this method as the *Method 1*.

The results for each list of articles can be seen in Table 16.3, Table 16.4, Table 16.5, Table 16.6, Table 16.7, Table 16.8, Table 16.9. We count the number of results for each topic and each ranking position, so that we can then compute the nDCG score described before.

Example: List of articles about drugs (386 articles): (82 in the first column and the first row means that for 82 out of 386 articles, the topic about fishes was the first result) Note: Since for some articles, there is no topic found at all, the number of articles does not necessarily equal to the sum of values in the first row (or in any other row).

topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	202	8	10	11	43	43	17
2nd result	32	19	33	16	50	32	19
3rd result	9	12	15	29	15	13	20
4th result	7	7	3	11	3	10	8
5th result	0	2	2	7	3	0	2
6th result	0	0	0	2	0	0	2
7th result	0	0	0	0	0	0	0

**Table 16.3.:** Search results for texts about drugs with the *Method 1*.

topic	drugs	<b>fishes</b>	theater	explorers	cyclists	wine	dog breeds
1st result	0	823	7	31	48	67	6
2nd result	2	90	20	105	138	165	48
3rd result	5	14	16	96	65	62	24
4th result	3	2	14	48	13	18	18
5th result	3	0	2	3	3	3	14
6th result	0	0	0	0	0	0	2
7th result	0	0	0	0	0	0	0

**Table 16.4.:** Search results for texts about fishes with the *Method 1*.

topic	drugs	fishes	<b>theater</b>	explorers	cyclists	wine	dog breeds
1st result	1	1	39	13	22	69	23
2nd result	8	3	23	11	16	20	24
3rd result	2	5	11	6	11	10	9
4th result	1	2	2	9	3	3	4
5th result	0	1	0	1	0	0	5
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.5.:** Search results for texts about theater with the *Method 1*.

topic	drugs	fishes	theater	<b>explorers</b>	cyclists	wine	dog breeds
1st result	2	26	52	73	54	97	20
2nd result	4	16	39	37	28	28	22
3rd result	2	5	15	28	16	6	17
4th result	2	4	4	7	3	7	6
5th result	1	2	1	3	0	1	4
6th result	0	1	1	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.6.:** Search results for texts about explorers with the *Method 1*.

topic	drugs	fishes	theater	explorers	<b>cyclists</b>	wine	dog breeds
1st result	1	4	5	15	152	16	0
2nd result	4	14	16	39	21	26	5
3rd result	6	6	16	13	0	11	8
4th result	2	2	7	5	0	3	3
5th result	0	0	1	3	0	3	2
6th result	0	0	0	0	0	0	1
7th result	0	0	0	0	0	0	0

**Table 16.7.:** Search results for texts about cyclists with the *Method 1*.

topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	0	0	0	0	0	575	0
2nd result	2	17	36	25	30	0	26
3rd result	1	5	3	7	5	0	8
4th result	0	0	1	4	0	0	1
5th result	0	0	0	0	0	0	0
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.8.:** Search results for texts about wine with the *Method 1*.

topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	0	18	18	10	6	35	406
2nd result	0	34	56	18	31	58	67
3rd result	0	18	22	22	15	22	10
4th result	1	4	10	10	15	5	1
5th result	0	2	3	12	3	3	0
6th result	0	0	0	2	1	0	0
7th result	0	0	0	0	0	0	0

**Table 16.9.:** Search results for texts about dog breeds with the *Method 1*.

topic	nDCG
drugs	0.62
fishes	0.86
theater	0.32
explorers	0.29
cyclists	0.84
wine	1.00
dog breeds	0.95
avg.	$0.70 \pm 0.27$

**Table 16.10.:** Overall search results with the *Method 1*.

The overall results with the *Method 1* are shown in Table 16.10. They are better than the baseline, the average nDCG is 0.70 against 0.61 with a random ranking. However, this search method has a drawback: it can match parts of words, for example if the input text contains the word cat and a topic contains the word catholicism, there will be a match because the word cat is contained in the word catholicism. This can cause relevance problems in the search results.

---



---

## 16.2.5 Exact term matching

---

In order to address the issue previously mentioned, we try a second method in which the matching process is performed with full words exclusively. Furthermore, since our topics are only constituted of nouns, we apply a part of speech tagger to the text and keep only the nouns and proper nouns. We refer to this method as *Method 2*.

topic	<b>drugs</b>	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	313	23	3	1	10	4	4
2nd result	12	48	14	4	28	12	20
3rd result	0	10	7	11	17	13	9
4th result	0	4	0	17	2	7	1
5th result	0	0	0	4	1	2	1
6th result	0	0	0	1	0	3	0
7th result	0	0	0	0	0	0	0

**Table 16.11.:** Search results for texts about drugs with the *Method 2*.

topic	drugs	<b>fishes</b>	theater	explorers	cyclists	wine	dog breeds
1st result	8	947	0	0	20	0	8
2nd result	22	18	11	14	92	29	53
3rd result	2	3	7	46	10	13	12
4th result	0	0	0	11	1	4	3
5th result	0	0	0	1	0	0	0
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.12.:** Search results for texts about fishes with the *Method 2*.

topic	drugs	fishes	<b>theater</b>	explorers	cyclists	wine	dog breeds
1st result	4	3	155	7	10	0	7
2nd result	7	10	6	26	11	1	24
3rd result	3	0	4	6	3	3	8
4th result	1	1	0	2	0	2	1
5th result	1	0	0	0	0	2	0
6th result	0	0	0	0	0	0	1
7th result	0	0	0	0	0	0	0

**Table 16.13.:** Search results for texts about theater with the *Method 2*.

topic	drugs	fishes	theater	<b>explorers</b>	cyclists	wine	dog breeds
1st result	9	45	59	75	19	6	27
2nd result	4	8	25	24	9	10	10
3rd result	1	2	0	15	2	6	9
4th result	0	1	1	2	1	0	2
5th result	0	0	0	2	0	0	0
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.14.:** Search results for texts about explorers with the *Method 2*.

topic	drugs	fishes	theater	explorers	<b>cyclists</b>	wine	dog breeds
1st result	6	4	5	3	103	1	5
2nd result	3	5	10	9	2	1	8
3rd result	1	5	0	3	1	3	1
4th result	0	0	0	2	0	3	2
5th result	0	0	0	1	0	0	0
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.15.:** Search results for texts about cyclists with the *Method 2*.

topic	drugs	fishes	theater	explorers	cyclists	<b>wine</b>	dog breeds
1st result	2	1	0	1	0	569	0
2nd result	27	332	84	28	15	3	14
3rd result	3	112	31	10	4	1	7
4th result	0	32	2	1	0	0	1
5th result	0	2	0	3	0	0	0
6th result	0	0	0	0	0	0	1
7th result	0	0	0	0	0	0	0

**Table 16.16.:** Search results for texts about wine with the *Method 2*.

topic	drugs	fishes	theater	explorers	cyclists	wine	<b>dog breeds</b>
1st result	17	55	22	18	17	3	338
2nd result	3	33	11	5	11	3	105
3rd result	1	11	2	9	13	27	18
4th result	0	2	0	13	1	11	0
5th result	0	0	0	0	0	3	0
6th result	0	0	0	0	0	0	0
7th result	0	0	0	0	0	0	0

**Table 16.17.:** Search results for texts about dog breeds with the *Method 2*.

---



---

topic	nDCG
drugs	0.83
fishes	0.90
theater	0.75
explorers	0.24
cyclists	0.51
wine	1.00
dog breeds	0.91
avg.	$0.73 \pm 0.25$

**Table 16.18.:** Overall search results with the *Method 2*.

The *Method 2* gives slightly better results than the first method (0.73 against 0.70). Nevertheless, there tend to be less topics in the results because of less matches.

---

### 16.2.6 Combining both methods

---

Finally, we try to combine both queries to keep the advantages of each one: the first method gives many results but some of them might not be relevant whereas the second one gives more relevant results but sometimes there are not a lot of them. We therefore apply both queries. We boost the results given by the second method, by multiplying their score by 10, such that these results are also the highest ranked in the global results and we add the results from the first method so that we have more results.

topic	<b>drugs</b>	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	304	23	2	2	20	13	8
2nd result	16	47	33	20	62	39	30
3rd result	8	17	26	31	20	29	18
4th result	5	14	10	17	9	17	9
5th result	0	3	7	6	3	12	2
6th result	0	2	1	1	1	1	2
7th result	0	0	0	1	0	1	1

**Table 16.19.:** Search results for texts about drugs with the *Method 3*.

topic	drugs	<b>fishes</b>	theater	explorers	cyclists	wine	dog breeds
1st result	1	986	2	3	18	12	13
2nd result	22	19	25	115	170	182	75
3rd result	6	7	19	111	68	92	22
4th result	6	5	14	51	16	41	13
5th result	4	0	7	9	6	7	3
6th result	0	0	0	0	0	3	5
7th result	0	0	0	0	0	0	0

**Table 16.20.:** Search results for texts about fishes with the *Method 3*.

topic	drugs	fishes	<b>theater</b>	explorers	cyclists	wine	dog breeds
1st result	3	4	148	17	15	3	13
2nd result	8	10	19	22	15	44	34
3rd result	8	1	6	22	7	33	11
4th result	0	5	1	7	10	18	1
5th result	2	0	1	1	4	5	2
6th result	0	0	0	0	1	0	4
7th result	0	0	0	0	0	0	0

**Table 16.21.:** Search results for texts about theater with the *Method 3*.

topic	drugs	fishes	theater	<b>explorers</b>	cyclists	wine	dog breeds
1st result	5	42	70	105	43	54	34
2nd result	8	14	30	64	29	45	17
3rd result	3	10	8	40	21	24	12
4th result	0	3	8	9	8	12	6
5th result	1	1	4	5	2	4	0
6th result	0	1	0	0	0	1	0
7th result	0	0	0	0	0	0	0

**Table 16.22.:** Search results for texts about explorers with the *Method 3*.

topic	drugs	fishes	theater	explorers	<b>cyclists</b>	wine	dog breeds
1st result	4	6	7	14	149	10	5
2nd result	8	13	21	42	20	18	7
3rd result	0	8	11	20	3	24	3
4th result	3	2	7	3	2	3	6
5th result	0	1	3	4	1	4	2
6th result	1	0	0	0	0	1	0
7th result	1	0	0	0	0	0	0

**Table 16.23.:** Search results for texts about cyclists with the *Method 3*.



topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	0	0	0	0	0	574	0
2nd result	28	333	84	29	15	0	17
3rd result	4	112	55	29	14	0	10
4th result	1	31	4	12	5	0	4
5th result	0	4	0	4	1	0	4
6th result	0	0	0	0	0	0	1
7th result	0	1	0	0	0	0	0

**Table 16.24.:** Search results for texts about wine with the *Method 3*.

topic	drugs	fishes	theater	explorers	cyclists	wine	dog breeds
1st result	1	36	10	4	9	4	429
2nd result	12	46	56	28	27	72	52
3rd result	5	18	27	29	22	51	2
4th result	4	4	17	20	12	13	0
5th result	0	7	8	7	3	10	0
6th result	0	0	2	2	0	0	1
7th result	0	0	0	0	0	0	0

**Table 16.25.:** Search results for texts about dog breeds with the *Method 3*.

topic	nDCG
drugs	0.84
fishes	0.94
theater	0.79
explorers	0.44
cyclists	0.83
wine	1.00
dog breeds	0.96
avg.	$0.83 \pm 0.17$

**Table 16.26.:** Overall search results with the *Method 3*.

The third method, combining both the first and the second method and boosting results from the second one, gives the best results. The average nDCG equals 0.83 against 0.70 and 0.74 with the first and second method. The standard deviation is also lower than before (0.17 against 0.27 and 0.25) reflecting the fact that the results are more homogeneous. As a consequence, the third method is the one applied in the web application.

---

---

## 16.3 Conclusion

---

In this part, we have presented a web application which allows to interact with topics. Several features have been introduced: the user can explore a topic model and each topic that it contains, the topic can also be visualized graphically. Finally, the user can input some text and find the topics best matching this text.

---

## **Part VI.**

# **Conclusion and future work**

---

---

## 17 Conclusion

In this thesis, we introduced a novel method and a system to generate topics using a Distributional Disambiguated Thesaurus. Our method can be decomposed into three subsystems. First, the DDT is preprocessed, discarding low frequency terms. Second, a clustering method isolates clusters of word senses within the DDT, these clusters are our topics. Finally, topics are annotated by hypernyms to represent the gist of the topic. We explored two hypernyms sources: a man-made lexical resource, WordNet and an automatically built IS-A relationships database, extracted using lexical-syntactic patterns.

Different values were tried out for the several parameters in our method: as input we had four different DDTs built using two different corpora, several thresholds for the term frequency based filtering during the preprocessing step and three unsupervised clustering methods. Variations in these parameters values resulted in 32 different topical models, that we compared through three independent experiments. In the first experiment, we manually inspected topics interpretability, in the second one we analyzed topics coherence using a hypernyms database and the last experiment was focused on mapping our topics to topics available in BabelNet. We found out that coarsed-grained models give globally better quality topics according to the two first experiments. Additionally, increasing the frequency threshold for the word senses in the DDT also improves the quality of our topics. According to the experiment with BabelNet topics, fine-grained topics were shown to get better results. In this experiment, models with lower frequency thresholds were also better.

Regarding topic annotations with hypernyms, we explored several strategies to make these hypernyms more relevant to the topic itself, with and without considering weights, more or less deep in the Wordnet hypernyms hierarchy and applying or not a TF-IDF scheme.

Finally, we introduced a web application which allows the user to interact with our topic models. In particular, possibility is offered to visualize topics as bags of words or as graphs and to search for a text best matching topics.

The implementation of the system for building topics, as well as the user interface is available on GitHub<sup>1</sup>.

---

<sup>1</sup> <http://github.com/smndf/Statistical-Models-of-Semantics-using-Structured-Topics>

---

## 18 Future work

Building on the work presented in this thesis, future work could include new experiments to evaluate topics quality, since the three experiments presented in this thesis do not give the same results. New experiments could definitely tilt the balance in favour of one topic model.

In the scope of this thesis, the DDTs were preprocessed in two different ways: keeping only the word senses tagged as nouns and proper nouns, and filtering word senses based on a frequency threshold. Future work could focus on elaborating more intelligent filtering methods, in order to improve the eventual topic model. Regarding the annotation of topics of hypernyms, other strategies, of weighting for example, could be explored.

Finally, a more general perspective is to bridge the gap between topic models generated by our system and existing common methods used to find topics, such as the Gibbs sampling method. Currently, this gap mainly results from the intrinsic characteristics of our topic models, such as the topics being a split of the whole vocabulary instead of a distribution over this vocabulary in the case of Gibbs sampling. This can be done using the graph structure of the DDT: for each of our topics, we could first compute a central node  $c$  for this topic e.g. by taking the node with the max centrality. Then, we could run a shortest path algorithm over the whole graph from the previously found central node  $c$ . Since edge weights in the graph denote semantic similarity between nodes, the distance between a given node and the central node  $c$  would be inversely proportional to the probability of occurrence of this given node in the topic of the central node  $c$ . This way, we could obtain distributions of words over the whole vocabulary for each topic, assuming the graph to be connected.

---

---

## List of Figures

4.1. Results page for the word "car" in thesaurus.com. . . . .	14
4.2. Pipeline for the computation of semantic similarity values. Source: [Biemann and Riedl, 2013]	15
4.3. Dependency parsing output for the sentence <i>I suffered from a cold and took aspirin.</i> . . . .	15
4.4. Neighborhood graph of the word tablet. Source: [Simon, 2015] . . . . .	16
5.1. Dataflow diagram of our system for building topics. . . . .	20
7.1. Example of the two steps of the Louvain Method. . . . .	26
7.2. Example of the steps during the Chinese Whispers clustering method. . . . .	27
8.1. Hypernym hierarchy of the word <i>cat</i> (feline sense). . . . .	33
8.2. Hypernym hierarchy of the word <i>table</i> (furniture sense). . . . .	33
11.1. Hypernyms graph of the demo topic. . . . .	43
12.1. Example of BabelNet synset for the word Python as a programming language. This synset belongs to the "computing" topic. Source: BabelNet.org. . . . .	48
14.1. Graph visualisation for a topic about wine/champagne. . . . .	56
15.1. Topic list. . . . .	60
15.2. View of a topic with all words. . . . .	60
16.1. View of the search home page. . . . .	61
16.2. View of the search results page for the text "Marcus is cooking chicken with rice." . . . .	62

---

---

## List of Tables

4.1. Sample from a thesaurus in which none of the words are disambiguated. . . . .	16
4.2. Sample from a thesaurus in which only the entry word is disambiguated. . . . .	17
4.3. Sample from a fully disambiguated thesaurus. . . . .	17
4.4. Statistics of the datasets used in our experiments. . . . .	18
6.1. Datasets metrics after removing all word senses with non noun tags. . . . .	21
6.2. <i>ddt-news-n200</i> filtered with different frequency thresholds. . . . .	22
6.3. <i>ddt-news-n50</i> filtered with different frequency thresholds. . . . .	22
6.4. <i>ddt-wiki-n200</i> filtered with different frequency thresholds. . . . .	22
6.5. <i>ddt-wiki-n30</i> filtered with different frequency thresholds. . . . .	22
7.1. Clustering results for Chinese Whispers applied to the different models. . . . .	29
7.2. Davies-Bouldin indices for Chinese Whispers applied to the different models, lower values mean better clustering. . . . .	29
7.3. Clustering results for the Louvain Method applied to the different models. . . . .	30
7.4. Davies-Bouldin indices for the Louvain Method applied to the different models, lower values mean better clustering. . . . .	30
7.5. Clustering results with MCL, the DDT <i>ddt-news-n200</i> and 2000 as word frequency threshold. . . . .	31
8.1. Sample of the IS-A relationship database. . . . .	35
8.2. Accuracy of methods with and without TF-IDF to annotate topics with hypernyms from the IS-A database. . . . .	36
10.1. Example of three interpretable topics. Extracted from results of DDT news200, 2000 as frequency threshold and clustered with Chinese Whispers. . . . .	40
10.2. Example of three non interpretable topics. Extracted from results of DDT news200, 2000 as frequency threshold and clustered with Chinese Whispers. . . . .	40
10.4. Comparison of results of interpretability annotations for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold. . . . .	40
10.3. Results of interpretability annotations for the different models. The five best ones are highlighted, the best one is also underlined. . . . .	41
10.5. Comparison of results of interpretability annotations for Chinese Whispers and the Louvain Method: for each clustering method, average over all models based on this method. . . . .	42
11.1. Hypernyms for each word of the demo topic. . . . .	43
11.2. Example of three topics with a high clustering coefficient of their hypernym graph. Extracted from results of DDT news50, 2000 as frequency threshold and clustered with Chinese Whispers. . . . .	44

---

11.3. Results of hypernym graphs analysis for the different models. The five best ones are highlighted, the best one is also underlined. "ratio non zero el." reflects the proportion of topics for which the clustering coefficient of the hypernym graph is different from zero. The next column "avg. clust. coef. non zero elements" is the average of the clustering coefficients of these topics. The last column "avg. clust. coef. all elements" is the average of the clustering coefficients for all hypernym graphs, including those with a clustering coefficient equal to zero. This the measure that we want to optimize. . . . .	45
11.4. Comparison of results of hypernoms graph analysis for Chinese Whispers and the Louvain Method: for each clustering method, average over all models based on this method. . . . .	46
11.5. Comparison of results of hypernoms graph analysis for the different DDTs: for each DDT, average over all models based on this DDT. . . . .	46
11.6. Comparison of results of hypernoms graph analysis for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold. . . . .	46
12.1. BabelNet topics names and number of synsets per topic. . . . .	49
12.2. Example of BabelNet mapping results for three topics. Extracted from results of DDT news50, 2000 as frequency threshold and clustered with Chinese Whispers. . . . .	50
12.4. Comparison of results of experiment with BabelNet topics for the different clustering methods: for each DDT, average over all models based on this DDT. . . . .	50
12.3. Overall results of experiment with BabelNet topics. The five best ones are highlighted, the best one is also underlined. . . . .	51
12.5. Comparison of results of experiment with BabelNet topics for the different DDTs: for each clustering method, average over all models based on this method. . . . .	52
12.6. Comparison of results of experiment with BabelNet topics for the different frequency thresholds: for each frequency threshold, average over all models based on this threshold. . . . .	52
13.1. Comparison results of the three experiments we conducted. The five best ones are highlighted, the best one is also underlined. . . . .	54
16.1. Metrics of test texts for search engine. . . . .	62
16.2. Results for 7 topics randomly distributed for each text. . . . .	63
16.3. Search results for texts about drugs with the <i>Method 1</i> . . . . .	64
16.4. Search results for texts about fishes with the <i>Method 1</i> . . . . .	65
16.5. Search results for texts about theater with the <i>Method 1</i> . . . . .	65
16.6. Search results for texts about explorers with the <i>Method 1</i> . . . . .	65
16.7. Search results for texts about cyclists with the <i>Method 1</i> . . . . .	65
16.8. Search results for texts about wine with the <i>Method 1</i> . . . . .	66
16.9. Search results for texts about dog breeds with the <i>Method 1</i> . . . . .	66
16.10 Overall search results with the <i>Method 1</i> . . . . .	66
16.11 Search results for texts about drugs with the <i>Method 2</i> . . . . .	67
16.12 Search results for texts about fishes with the <i>Method 2</i> . . . . .	67
16.13 Search results for texts about theater with the <i>Method 2</i> . . . . .	67
16.14 Search results for texts about explorers with the <i>Method 2</i> . . . . .	68



---

---

16.15	Search results for texts about cyclists with the <i>Method 2</i> . . . . .	68
16.16	Search results for texts about wine with the <i>Method 2</i> . . . . .	68
16.17	Search results for texts about dog breeds with the <i>Method 2</i> . . . . .	68
16.18	Overall search results with the <i>Method 2</i> . . . . .	69
16.19	Search results for texts about drugs with the <i>Method 3</i> . . . . .	69
16.20	Search results for texts about fishes with the <i>Method 3</i> . . . . .	70
16.21	Search results for texts about theater with the <i>Method 3</i> . . . . .	70
16.22	Search results for texts about explorers with the <i>Method 3</i> . . . . .	70
16.23	Search results for texts about cyclists with the <i>Method 3</i> . . . . .	70
16.24	Search results for texts about wine with the <i>Method 3</i> . . . . .	71
16.25	Search results for texts about dog breeds with the <i>Method 3</i> . . . . .	71
16.26	Overall search results with the <i>Method 3</i> . . . . .	71

---

## Bibliography

- [Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735.
- [Basile et al., 2014] Basile, P., Caputo, A., and Semeraro, G. (2014). An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING2014, the 25th International Conference on Computational Linguistics*, pages 1591–1600.
- [Biemann, 2006] Biemann, C. (2006). Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 73–80.
- [Biemann et al., 2007] Biemann, C., Heyer, G., Quasthoff, U., and Richter, M. (2007). The leipzig corpora collection-monolingual corpora of standard size. *Proceedings of Corpus Linguistic*.
- [Biemann and Riedl, 2013] Biemann, C. and Riedl, M. (2013). Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, pages 55–95.
- [Blei and Lafferty, 2006] Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.
- [Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, pages 993–1022.
- [Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, page P10008.
- [Davies and Bouldin, 1979] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, pages 224–227.
- [Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, pages 391–407.
- [Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, pages 269–271.
- [Donoho and Stodden, 2004] Donoho, D. and Stodden, V. (2004). When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16*, pages 1141–1148.
- [Faralli et al., 2016] Faralli, S., Panchenko, A., Biemann, C., and Ponzetto, S. P. (2016). Linking lexical resources to disambiguated distributional semantic networks. In *Proceedings of the 15th International Semantic Web Conference*.

- 
- [Ferraresi et al., 2008] Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.
- [Franco-Salvador et al., 2013] Franco-Salvador, M., Gupta, P., and Rosso, P. (2013). Cross-language plagiarism detection using a multilingual semantic network. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, pages 710–713.
- [Gabrilovich and Markovitch, 2007] Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *International Joint Conferences on Artificial Intelligence*, pages 1606–1611.
- [Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, pages 721–741.
- [Gliozzo et al., 2013] Gliozzo, A., Biemann, C., Riedl, M., Coppola, B., Glass, M. R., and Hatem, M. (2013). Jobimtext visualizer: a graph-based approach to contextualizing distributional similarity. *Graph-Based Methods for Natural Language Processing*, pages 6–10.
- [Graff and Cieri, 2003] Graff, D. and Cieri, C. (2003). English gigaword corpus. *Linguistic Data Consortium*.
- [Griffiths, 2002] Griffiths, T. (2002). Gibbs sampling in the generative model of latent dirichlet allocation. *Tech. rep., Stanford University*.
- [Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics. Nantes, France*, pages 539–545.
- [Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the 15th conference on Uncertainty in artificial intelligence*, pages 289–296.
- [Klema and Laub, 1980] Klema, V. and Laub, A. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, pages 164–176.
- [Likas et al., 2003] Likas, A., Vlassis, N., and Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition*, pages 451–461.
- [Miller et al., 1990] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, pages 235–244.
- [Miller and Charles, 1991] Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, pages 1–28.
- [Navigli et al., 2013] Navigli, R., Jurgens, D., and Vannella, D. (2013). Semeval-2013 task 12: Multilingual word sense disambiguation. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 222–231.
- [Navigli and Ponzetto, 2010] Navigli, R. and Ponzetto, S. P. (2010). Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225.

- 
- [Panchenko et al., 2013] Panchenko, A., Romanov, P., Morozova, O., Naets, H., Philippovich, A., Romanov, A., and Fairon, C. (2013). Serelex: Search and visualization of semantically related words. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, pages 837–840.
- [Ponzetto and Navigli, 2010] Ponzetto, S. P and Navigli, R. (2010). Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1522–1531.
- [Rosen-Zvi et al., 2004] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494.
- [Salton and McGill, 1986] Salton, G. and McGill, M. J. (1986). *Introduction to modern information retrieval*.
- [Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, pages 613–620.
- [Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, pages 1–47.
- [Simon, 2015] Simon, J. (2015). Word sense induction using distributional semantics. Master’s thesis, Technische Universität Darmstadt, Darmstadt, Germany.
- [Van Dongen, 2001] Van Dongen, S. M. (2001). *Graph clustering by flow simulation*. PhD thesis, University of Utrecht.
- [Wallach, 2006] Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984.
- [Wallach et al., 2009] Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. (2009). Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112.