UNIVERSITÉ DE LYON - LUMIÈRE LYON 2

ECOLE DOCTORALE INFOMATHS

P H D T H E S I S

to obtain the title of

PhD of Computer Science

of the University of Lyon - Lumière 2

Defended by

Edmundo-Pavel Soriano-Morales

Hypergraphs and information fusion for term representation enrichment. Applications to word sense disambiguation and named entity recognition

Thesis Advisor: Sabine Loudcher and Julien Ah-Pine

prepared at Université de Lyon 2, ERIC Laboratory defended on

Jury :

Abstract

Making sense of textual data is an essential requirement in order to make computers understand our language. To extract actionable information from text, we need to represent it by means of descriptors before using knowledge discovery techniques. The goal of this thesis is to shed light into heterogeneous representations of words and how to leverage them while addressing their implicit sparse nature.

First, we propose a hypergraph network model that holds heterogeneous linguistic data in a single unified model. In other words, we introduce a model that represents words by means of different linguistic properties and links them together according to said properties. Our proposition differs to other types of linguistic networks in that we aim to provide a general structure that can hold several types of descriptive text features, instead of a single one as in most representations. This representation may be used to analyze the inherent properties of language from different points of view, or to be the departing point of an applied NLP task pipeline. Secondly, we employ feature fusion techniques to provide a final single enriched representation that exploits the heterogeneous nature of the model and alleviates the sparseness of each representation. These type of techniques are regularly used exclusively to combine multi-media data. In our approach, we consider different text representations as distinct sources of information which can be enriched by themselves. This approach has not been explored before, to the best of our knowledge. Thirdly, we propose an algorithm that exploits the characteristics of the network to identify and group semantically related words by exploiting the real-world properties of the networks. In contrast with similar methods that are also based on the structure of the network, our algorithm reduces the number of required parameters and more importantly, allows for the use of either lexical or syntactic networks to discover said groups of words, instead of the single type of features usually employed.

We perform our experiments and developments using open-access corpora. We focus on two different natural language processing tasks: Word Sense Induction and Disambiguation (WSI/WSD), and Named Entity Recognition (NER). In total, we test our propositions on three different datasets. The results obtained allow us to show the pertinence of our contributions and also give us some insights into the properties of heterogeneous features and their combinations with fusion methods. Specifically, our experiments are twofold: (1) we show that using fusion-enriched heterogeneous fea-

tures, coming from our proposed linguistic network, we outperform the performance of single features' systems and other basic baselines. We analyze the fusion operators used to get to these improvements. In general, using single fusion operators is not as efficient as using a combination of them to arrive to a final space representation. We test on both WSI/WSD and NER tasks mentioned above. And (2), we address the WSI/WSD task with our network-based proposed method in order to demonstrate its relevancy to the task. We compare the performance of the algorithm we propose to determine a set of senses and then assign them against those related network-based methods. We show the different results obtained with either lexical or syntactic features, and discuss their characteristics on this task. Finally, we parse a corpus based on the English Wikipedia and then store it following the proposed network model. The parsed Wikipedia version serves as a linguistic resource to be used by other researchers. Contrary to other similar resources, instead of just storing its part of speech tag and its dependency relations, we also take into account the constituency-tree information of each word analyzed. The hope is for this resource to be used on future developments without the need to compile such resource from zero.

Keywords. Natural Language Processing, Linguistic Network, Word Representation, Fusion Techniques, Word Sense Induction and Disambiguation, Named Entity Recognition

Contents

List of Tables

List of Figures

Chapter 1

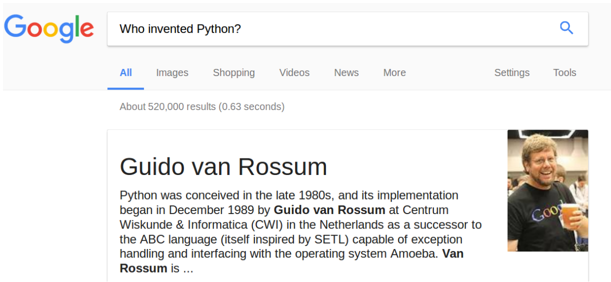
Introduction

Contents

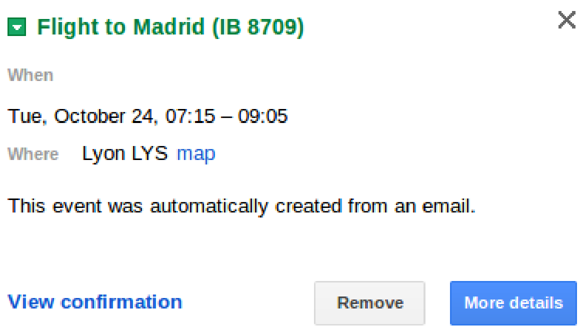
1.1 Context

Making sense of texts plays a vital role on the evolution of general artificial intelligence. Given the constantly-growing generation of textual data, there is the need of computational systems able to extract useful information from large quantities of textual collections, mainly to facilitate our day-to-day activities and, not less important, to find useful latent information hidden behind these large quantities of data. For example (see Figure 1.1), Google, the search engine giant, is now able to conveniently answer short questions by analyzing textual knowledge bases, such as the English Wikipedia, in order to find the answer. Furthermore, Gmail, Google's electronic mail client, now automatically identifies events, and sometimes their location and participants, from our personal emails and then adds them to our online agendas. On the other hand, finding relations among concepts within a set of documents can be a rich source of knowledge. An example: using text mining techniques, in the biomedical domain, facts can be linked across publications generating new hypotheses directly from the literature [Garten 2010].

Indeed, making computers learn, via theories, algorithms and applications, is the general objective of artificial intelligence research [Sugiyama 2016]. Coming from this



(a)



(b)

Figure 1.1: (a) While searching *Who invented Python?*, Google recognizes the simple question an directly gives us the answer from Wikipedia. (b) Gmail detects we received an email from an airline and parses it, finds the date, and automatically creates the corresponding event in our calendar.

multi-disciplinary area, Natural Language Processing (NLP) is the domain that aims to make machines understand our language [Jurafsky 2009] and thus making it possible to communicate with them in our own language. Specifically, speech and text, the latter being the focus of this work.

Although a challenging task, primarily given the ambiguity and dynamics of human language, NLP has developed rapidly [Clark 2010] during the last two decades mainly due to the combination of three factors:

- The availability of **large quantities of freely-accessible textual data**: primarily enabled by the current Web technologies, we are today able to download with a single click the entire content of the English (or other languages) Wikipedia. In the same sense, we can also download thousands of gigabytes of Web crawled data. This information is used to derive knowledge about the text itself, as we will see in the rest of this dissertation.

- The **computational power** at our disposition: from consumer-based computers able to perform parallel computations with considerably large datasets; to on-demand distributed cloud platforms with high performance computing nodes. The latter may be from private providers, e.g., AWS Cloud Service[1], Microsoft Azure[2], etc; or furnished by public organizations, such as France's Lyon 1 University[3] or the National Institute of Nuclear Physics computing centers[4].

- The **large quantity of open-source text mining and data science analysis tools**. Luckily, it is becoming more common for NLP laboratories around the world to make their developments available to the general public, e.g, Stanford University CoreNLP[5], Antwerp's University CLiPS Pattern[6]. Additionally, large Web companies, such as Facebook[7] and Google[8], frequently publish their research code and utilities. Lastly, communities of individuals develop libraries that grow to become essential building blocks of several applications and research in the domain. Notably, `scikit-learn`[9], a popular data science library implementing several well-known machine learning algorithms. Regarding NLP specifically, two up-to-date libraries stand out: `gensim`[10] and `spaCy`[11]. These are, for the most part, cross-platform, high performance, optimized, well maintained, documented, and easily installable libraries.

Solutions to NLP tasks generally follow three steps to achieve their respective goals [Aggarwal 2012, Jurafsky 2009]. We can see in Figure 1.2 the typical steps of a NLP system. First, in **Preprocessing**, an input corpus is "normalized" so that it will be easier to treat it in the following steps. Secondly, in **Feature Representation**, numerous features are extracted from the preprocessed text. Thirdly, in **Knowledge Discovery**, a machine learning or rule-based (less common nowadays) technique is used to learn a model able to provide an interesting insight within the existing data as well as on new future instances. The output of said system is usually the model or the language

[1] https://aws.amazon.com/
[2] https://azure.microsoft.com/en-us/
[3] https://p2chpd.univ-lyon1.fr/
[4] https://cc.in2p3.fr/
[5] https://stanfordnlp.github.io/CoreNLP/
[6] http://www.clips.ua.ac.be/pattern
[7] https://github.com/facebookresearch
[8] https://github.com/google
[9] http://scikit-learn.org/
[10] https://radimrehurek.com/gensim/
[11] https://spacy.io/

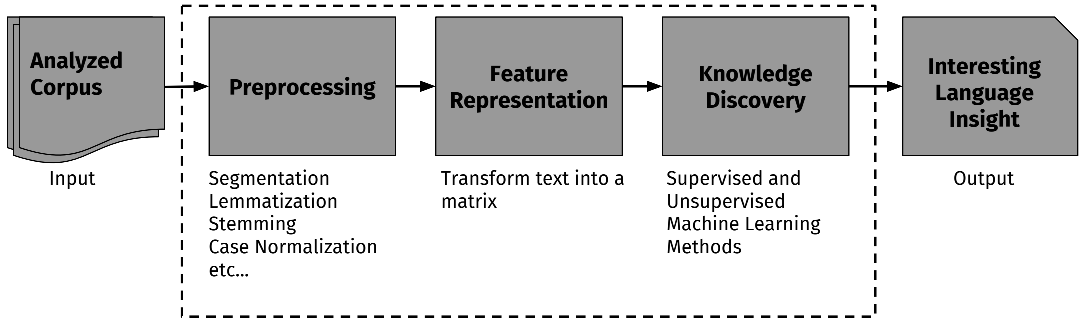


Figure 1.2: Typical steps of Natural Language Processing applications.

knowledge that reveals an interesting piece of information contained in the input corpus.

Natural Language Processing is used today for several practical applications. From the elementary tasks that aim to extract linguistic features directly from the text, to more applied systems that employ said features to solve challenging problems. For example, as elementary tasks there is Part-of-Speech (PoS) tagging and Named Entity Recognition (NER). The former, PoS tagging, aims to determine a syntactic class (part of speech) for each word in a corpus [Jurafsky 2009]. The latter, NER, determines if a proper noun is a place, a person, an organization or any other type of entity required by the final domain and application [Nadeau 2007] . An intermediate task, Word Sense Induction and Disambiguation (WSI/WSD) determines and assigns the semantic meaning of a given word according to its context [Clark 2010].

More complex tasks that generally employ one or more of the aforementioned techniques in order to get more descriptive features from the text and thus get us closer to understand what is being discussed. As an example, sentiment analysis which ultimate goal of this task is to determine the positiveness or negativeness (or neutrality) of an opinion expressed in a text [Liu 2012]. In this case, it would be useful to know what words express a “sentiment”: usually adjectives, categorized via PoS tagging; it would be informative to know what (or about who) we are talking about, with NER tagging; as well as the specific context of the words that are being used in the opinion, via WSI/WSD.

1.2 Challenges and Contributions

There are several research challenges that arise from the choices taken in each one of the steps comprising the NLP system's flow (Figure 1.2). In this thesis, we particularly focus on three challenges arising in both the Feature Representation and Knowledge Discovery phases. These challenges are: (1) modeling, extracting, and storing different types of linguistic features from raw text, (2) dealing with the sparsity inherent to text data features and also successfully combining them to get better representations, and (3) finding relations between words and then leveraging them in order to discover their latent relatedness and be able to solve NLP tasks.

We propose three contributions, one in terms of theoretical modelization and two in terms of NLP applications. Specifically, the contributions that we propose in this work are the following:

- a hypergraph network-based model to hold and combine heterogeneous linguistic data
- a network-based algorithm to discover semantic relatedness between linked words
- a method to combine heterogeneous representations while at the same time alleviating the sparsity problem common while dealing with text features.

These contributions are tested and evaluated using two different NLP semantic tasks: Word Sense Induction and Disambiguation, and Named Entity Recognition. We chose these two tasks as they are semantic problems directly benefited by those methods that are able to determine the relatedness among words, which is the case of the techniques we propose. Not less important, we attack these tasks as they are central building blocks of more intricate text analysis systems. Our propositions are built using open source tools and trained/tested using freely accessible corpora. We aim to make our software implementations are to multi-threaded CPU computers when applicable.

1.2.1 Modeling linguistic features

Challenge Representing unstructured text within a model that describes textual units and their corresponding features is a critical step within a NLP process. Textual units – either words, sentences, paragraphs, documents, etc – need to be represented by some kind of model that will allow for numerical analyses to be applied. Usually, textual units are represented in a vectorial space, where each dimension represents a feature; or in a graph-like structure, where features link units together. Concerning the features themselves, their selection is often an empirical process determined by the final goal of the NLP process at hand. Nonetheless, we have access to several types of linguistic features, each one representing the text from different points of view. Furthermore, texts usually containing large vocabularies involves the need of an efficient way of storing a corpus and its features. These possibilities entail the following research questions: **what type of model can we employ to represent a corpus through a set of heterogeneous features, extracted from itself, while keeping record of the relationships between textual units? How can we organize and store this model as simply and efficiently possible?** Answering these questions would allow us to properly design and build a linguistic resource containing heterogeneous descriptions of the textual units[12] adapted to solve NLP tasks.

Contribution During the last decade, graphs have been used to model textual data given its ability to naturally represent the dynamics and structured of text. We propose a linguistic resource in the form of an heterogeneous language network to be used as a first essential data model to solve Natural Language Processing tasks.

The originality of our work consists in taking into account different types of features, e.g., lexical, syntactical, and orthographic information; and unifying them under a single hypergraph structure. An hypergraph differs from a graph in that its edges may link several nodes together at the same time. This flexibility allows for simple and efficient access to the stored elements, either specific types of words or specific features. (revoir: advantages of hyperedges] We use the proposed model as the starting point of our other two contributions: solving Word Sense Disambiguation and Induction and Named Entity Recognition.

Lastly, as a proof of concept and in order to test the implementation practicality of our model, we process the English Wikipedia corpus and store its heterogeneous features under the form of the proposed model. We particularly focus onto the lexical and syntactical characteristics of words.

[12]In this work we focus on words. As such, the rest of this dissertation deals with the representation of words.

1.2.2 Leveraging the network to find semantic relatedness

Challenge Leveraging the structure within the proposed linguistic network is one of our main reasons to build such a graph-based language resource. This structure, namely the features linking words together, originate groups or communities of related words within the network. In that sense, leveraging these latent communities is still today an open question in the domain of graph-based NLP. Particularly in the context of semantic NLP tasks, where determining the relation among words is of utmost importance, we rise the following questions: **what kind of communities exist within language networks? How can we find and employ them to solve NLP tasks?** Furthermore, assuming an heterogeneous network like the one we propose, **what are the quantitative and qualitative differences, both in terms of performance and results, between the different representations existing within the network?** Determining the structure inside a language network, as well as devising an algorithm to exploit it would allow us to better understand the role of communities in graph-based approaches for NLP. Finally, getting a glimpse of the differences between each heterogeneous feature can help us to decide which is the most appropriate according to a NLP system objective.

Contribution Linguistic networks are complex structures that may hold heterogeneous entities and links together. Properly leveraging these structures has been indeed a popular area of research in the NLP literature.

We propose a variant to a literature algorithm that solves word sens induction and disambiguation mainly by leveraging the structure of a language network in. The assumption of the algorithm is that of the network having "real-world" characteristics, broadly, this means that there are several tight-knit groups of words within the structure. Nonetheless, contrary to the existent model, our proposition differs regarding the considerably lower number of parameters by adjusting them automatically according to the statistics of the concerned network. We also allow for more flexibility of the studied contexts of each word. Furthermore, we leverage the structure of our proposed linguistic model and go beyond the classic homogeneous cooccurrences by studying the effect of heterogeneous features on the quality of the senses induced by the system. Our experiments show the interest of our method by improving on the performance of similar methods and by being on the same ballpark of state-of-the-art methods. We also thoroughly analyze the characteristics of the results –the word senses– according to the different types of obtained by our system.

We improve the overall performance compared to other similar graph-based techniques.

1.2.3 Combining features and dealing with sparsity

Challenge While the proposed linguistic network contain heterogeneous features, in our previous propositions we have exclusively employed them separately. Nonetheless, employing these different attributes on a single textual representation is equally useful in terms of solving NLP applications. A certain type of feature may indicate relations that are completely unknown in another representation space. Thus a certain type of features can complement another to improve the overall description of words.

Another challenge that arises when building large cooccurrence networks, such as ours, is data sparsity. Indeed, sparsity is one of the main characteristics of textual data. Natural language processing systems rely on accurate information being found within a corpus. However, it is hard to see all the possible word cooccurrences in an input corpus and thus a system trained from it is not able to apply the acquired knowledge when it encounters unseen words and their cooccurrences.

Towards addressing both challenges previously described we pose the following questions: **how to alleviate data sparsity on textual data?** Concerning combining linguistic features, h**ow can we produce a single textual representation that is able to leverage the complementarity among features?** Lastly, **what is the behavior of combining features against using them independently?** The answer to these questions may shed light into more robust NLP systems, able to cope with sparsity while leveraging at the same time useful information coming from different types features.

Contribution Addressing the sparsity of textual data is not an easy task and often involves complex procedures and loss of information. To alleviate this issue, we propose the application of multimedia analysis fusion techniques to solve NLP semantic tasks. The fusion methods we employ comprise a set of methods to combine (or fuse) different types of features into a single unique representation. While combining attributes we also enrich them by leveraging the complementary information they carry individually. Furthermore, we address the challenge of data sparsity by transferring unseen relations from one feature space to another, that is, we obtain a denser similarity space by joining together both feature spaces. The experiments we carry out,

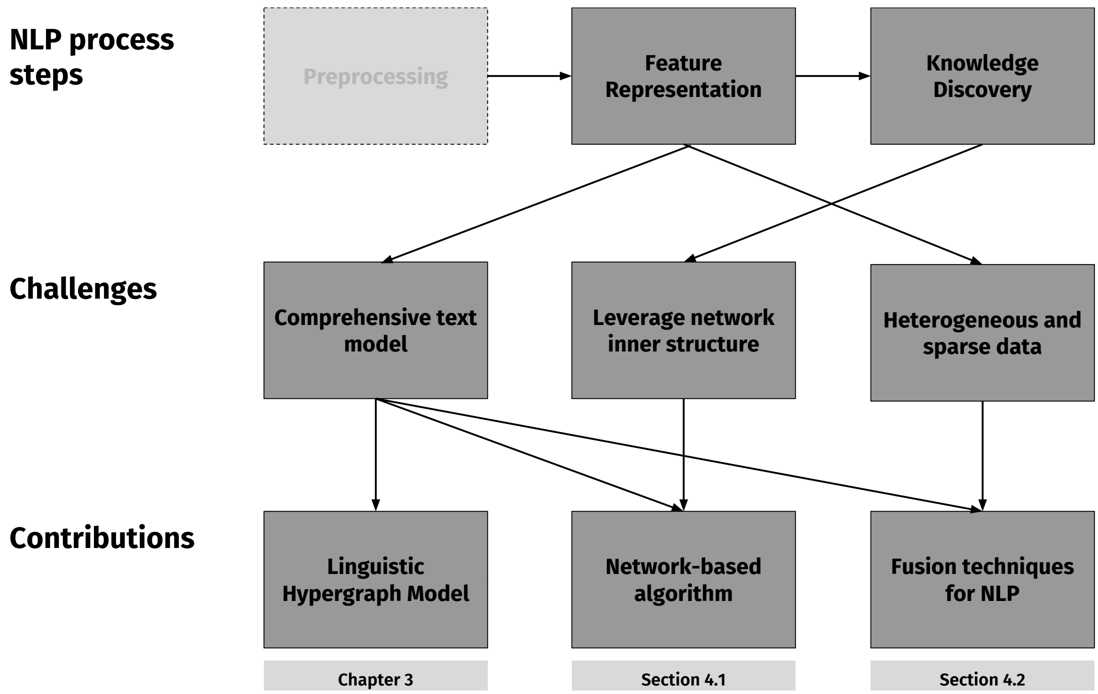


Figure 1.3: Block diagram of the NLP steps of interest, the challenges we address, and the contributions we propose.

in word sense induction and disambiguation and named entity recognition, show the pertinence of our approach. Specifically, we try different fusion techniques as well as several fusion configurations to improve the tasks' performance compared with using representations independently. Additionally, we study to what extent each type of fusion employed affects the performance of the tasks we evaluate.

1.3 Structure of the Dissertation

Figure 1.3 synthesizes the concerned NLP process-flow stages of this work (first line), the challenges that we aim to alleviate (second line), and their effect on the contributions we propose (third line). The remainder of this thesis is structured as follows.

Chapter 2 This chapter contains the theoretical background on the concepts discussed in this thesis. At the same time, we present the state of the art on the techniques that are relevant to our work. Specifically, we discuss the basics on text representation and how they are all related together by the distributional hypothesis. We then introduce the two main types of mathematical entities to manipulate text in a computer: vector-space models and graph-based models. Given our choice to work with graphs, we continue this path and introduce the types of textual graphs that concern us. Finally, we describe an inherent problem to text data: sparsity.

Chapter 3 We begin by giving a review on how linguistic networks are used in the literature. This contextualizes our first contribution. We present and define a theoretical contribution: a novel structure to hold language information based on a hypergraph linguistic model. We discuss its characteristics and the intuitions behind its conception: the choice of the structure, the role of nodes and heterogeneous edges, the type of features stored, the advantages it represents in terms of accessing and manipulating the data. Then, we introduce a set of techniques to make use of heterogeneous relations while dealing with sparsity, the feature fusion techniques. These methods are integral part of our contributions. Finally, we present a concrete application of our hypergraph model using the English Wikipedia. We describe its properties and motivations. Contents of this chapter are published in [Loudcher 2015, Soriano-Morales 2016a, Guille 2016].

Chapter 4 In this chapter, we present our two applied contributions. First, an algorithm that exploits the structure of the network, i.e., the connections between nodes to solve word sense induction and disambiguation. We test the linguistic and lexical features and discuss about its qualities. Our results improve on the performance of similar propositions from the literature. Secondly, we explore the application of multimedia fusion techniques using linguistic features to solve NLP tasks. We experiment with these methods on three datasets for named entity recognition and one dataset for word sense induction and disambiguation. Indeed, we show that using certain configurations of fusion techniques can lead to improvements over single-feature and trivial-concatenation representation matrices. Furthermore, we explore the contribution from each feature space for each sense and class in each task respectively. This work has been published or accepted for publication in [Soriano-Morales 2016b, Soriano-Morales 2017].

Chapter 5 We conclude this dissertation and present possible avenues for future work.

Chapter 2

Background

Abstract. *This chapter goes into detail about the notions of the theoretical our work is based on. First, we introduce the distributional hypothesis and the parameters involved in the generation of descriptive contexts from a corpus. Secondly, present how can we describe the distributional contexts within a model, either directly through a vectorial representation or by means of a graph-based representation. Thirdly, we discuss one of the main challenges of dealing with textual data: data sparsity. We cover what is it, its consequences, as well as existent solutions to it. Finally, we summarize the concepts introduced and contextualize our propositions.*

Contents

2.1 Distributional Hypothesis

The work we present in this thesis is prominently based on the distributional hypothesis (DH). This is also the case for the large majority of semantic approaches in NLP today. This context-analysis insight is usually credited primarily to [Harris 1954]. The

1. A bottle of **tesgüino** is on the table.
2. Everybody likes **tesgüino**.
3. **Tesgüino** makes you drunk.
4. We make **tesgüino** out of corn.

Figure 2.1: Even though *tesgüino* is a relatively obscure word, from its context we can understand its an alcoholic drink made from corn.

hypothesis is simple yet powerful: it can be formulated as "You shall know a word by the company it keeps." [Firth 1957]. More formally, it states that the similarity of meaning correlates with the similarity of the word's context distribution. It follows that the meaning of a word can be determined by the set of contexts in which it occurs in a corpus. Consequently, for two target words, the larger the number of shared contexts, the semantically closer these words are.

Taking the classic example by [Nida 1979], shown in the four lines in Figure 2.1, even if we do not know that the word tesgüino (or tejuino) refers to a ceremonial corn beer consumed by the native people of the north of Mexico, we can infer, by looking at its context, that we are indeed referring to some sort of corn alcoholic beverage. We can then easily infer that tesgüino is similar to other drinks such as tequila or mezcal.

Although we usually find in the literature that the work of Harris is the most important concerning the DH, we should note that the hypothesis rests on two theories [Sahlgren 2008, Turney 2010]: the statistical semantics hypothesis [Booth 1955], and the structuralism theory, as described by [De Saussure 1916]. The former is important as it place the DH in a within a larger context. Broadly, it affirms that statistical patterns of human word usage can be employed to understand what people mean. The latter, gives the DH a more robust approach towards the definition of what kind of distributional context should we use to determine similarities, as well as what does it the meaning of the contextual relations between words. In plain words Saussure's version of the structuralist theory indicates that the differences in the contexts of a word determine its role within a language system.

To this end, Saussure proposed two kinds of context relations: syntagmatic and paradigmatic. Syntagmatic relations concern the context defined by words that co-occur in the text, such as collocations (multi-word expressions that occur frequently in a corpus) and colligations (relations between a lexical item and a grammatical category) [Lehecka 2015]. On the other hand, paradigmatic relations associate words

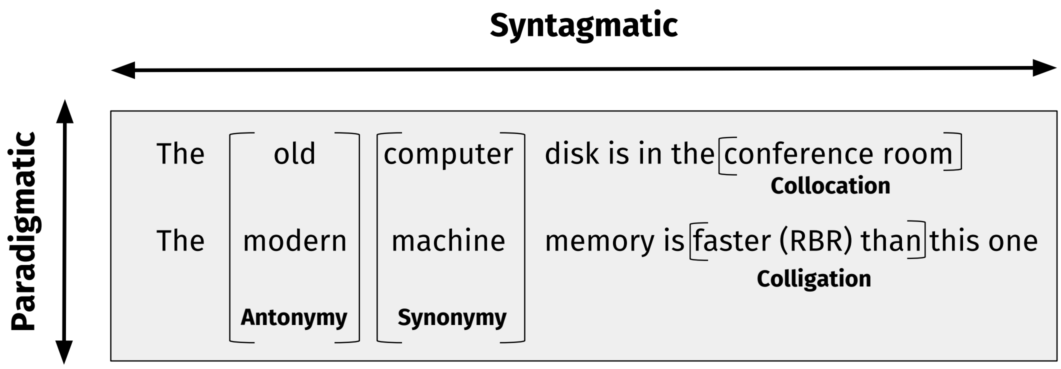


Figure 2.2: Examples of syntagmatic and paradigmatic contexts and their corresponding semantic relationships. Based on the examples by [Sahlgren 2008, Molino 2017].

according to whether they appear or not in the same context. These contexts define classic semantic relations such as synonymy-antonymy or hypernymy-hiponymy. Coupling both the DH with structuralist theory gives birth to the more refined definition of the distributional hypothesis, introduced by [Sahlgren 2008]: the refined distributional hypothesis. We note that this distinction, syntagmatic versus paradigmatic relations, is also more recently defined in [Schütze 1993]. In their work, syntagmatic relations are called first-order co-occurrences, while paradigmatic relations are referred to as second-order co-occurrences.

An example of syntagmatic and paradigmatic relationships can be seen in Figure 2.2. Vertically looking at the words, *old* and *modern*, in the first and second phrase respectively, share a paradigmatic context which leads to an antonym semantic relation. Something similar happens with words *computer* and *machine*, this time sharing a synonym relation. With respect to the syntagmatic relationships, horizontally looking at the words, we find the collocation *conference room* in the first phrase, as well as the colligation in the second expression between the word *than* usually preceded by a comparative adverb (RBR), *faster* in this case.

In spite of Sahlgren's distributional hypothesis definition, determining the types and meaning of semantic relations, obtained with distributional methods, is still an open challenge on distributional models and methods. Determining the specific type of semantic relation (e.g., synonymy, hyponymy, meronymy) is still an open issue in the community [Turney 2010, Fabre 2015, Périnet 2015a]. While distributional models can give us fast access to semantic relations between words within a corpus, they are

The code in the chip works

An interaction between a chip and a coil

The coil creates a magnetic field

Figure 2.3: Example text. Functional words are grayed-out.

most of the times ambiguous relations. It is still our task, as users, to determine the type of semantic relations found, in the case these distinctions are needed by the NLP system at hand.

Distributional methods, based on the DH, have been used for a long time now [Jurafsky 2009], although computationally automatized since the 1990s [Périnet 2015a]. Being a mature research field, systems based on these distributional models are varied and cover a large range of NLP tasks being obviously most popular on semantic tasks [Bruni 2014]. We do note that nowadays, they have somewhat resurfaced (although they really never went away) thanks to the recent re-introduction of word embeddings, or simply word distributional representations. In short, a word embedding, in the context of newer developments, is a vectorial representation that "embeds" words into a low-dimensional space, usually generated either by means of some sort of matrix reduction [Lebret 2013, Levy 2014b] or by using neural networks [Collobert 2011, Mikolov 2013]. These representations are usually obtained from very large bodies of text and they have shown to be quite effective for solving NLP tasks.

The actual implementation of a distributional model consists in three steps: (1) determine what type of context is going to be used, (2) chose a computable context representation, and (3) determine a weighting scheme and a relatedness measure.

We move now onto the description of what are the types of contexts commonly used while implementing a distributional model to represent words. We cover two types: lexical co-occurrence and syntactic co-occurrence. In this work we will exclusively focus on those two contexts. The first one describes a word's context based on its nearby words. The second defines a word's context according to the syntactic relations between the word and its neighbors. We will use the example phrases in Figure 2.3 to illustrate the kind of contexts we will describe below.

Table 2.1: Lexical contexts of the words *code*, *chip*, and *coil* appearing in each one of the phrases on Figure 2.3. The context is paradigmatic, the window being the word and 2 words to the left and right.

Words	Lexical Context
code	code;**w+1**:chip; **w+2**:works
chip	**w-2**:interaction; chip; **w+1**:coil
coil	coil;**w+1**:creates; **w+2**:magnetic

2.1.1 Lexical Contexts

Also called linear contexts, lexical contexts consist on those words that co-occur with a given word in a predetermined neighborhood: either in a sentence, a paragraph or larger units of text such as full documents [Levy 2014a, Sahlgren 2008]. Nowadays, the lexical context is usually determined by a window of n tokens to each side of the target word. As an example is shown in Table 2.1, where the context -2+2 of the words *code*, *chip*, and *coil* is shown. The size of the context depends of course of the application of the system. Indeed, determining the size is actually a quite empirical decision. Nonetheless, it seems that today the literature [Daume 2006, Mikolov 2013, Levy 2014a, Levy 2014b] has settled for a two-words to the left and two-words to the right context window (plus the target word), otherwise represented as -2+2. [Sahlgren 2006] discusses the motivations of selecting a window of a particular size. He notes that given the literature evidence, a shorter context window (specifically a -2+2 window) is preferable for acquiring semantic information. In this sense, [Jurafsky 2009] determines that generally the context size used lies between 1 and 8 words on each side, or 3 and 17 in total. In practical terms, the choice of the size affects the scope of the semantic relatedness: the shorter the context, the more specific is the information about a target word, approaching syntactic relations. Furthermore, these relations are "stronger" in the sense of being semantically similar, we could in theory substitute one word for another as the shared relation is antonymy. On the contrary, the larger the window, the broader the information conferred by the context words.

Lexical co-occurrences are the most popular way to represent distributional contexts. They are easy to obtain as there is no need for external information except the input corpus itself.

2.1.2 Syntactic Contexts

The second type of context depends on a more profound analysis of text. As its name implies, syntactic contexts are based on the analysis (or parsing) of text in order to obtain sense from them. Lexical contexts are able to somehow take into account the order of appearance of words in a phrase. Still, words in a sentence are not related among them like a list: semantic information is indeed extracted from words themselves, however syntax highly affects the way information is combined into semantic structures. Words tend to form groups between themselves, called constituents or chunks, which relate to other constituents to form a single phrase unit [Bender 2013].

Constituent Tree Indeed, constituents are represented with tree structures aptly named constituents parse tree, or simply parse tree (see Figure 2.4) [Jurafsky 2009]. These trees actually represent the context-free grammars models that we use to describe the chunk structure. As such, the parse tree differentiates between terminal, pre-terminals and non-terminal nodes. Non-terminal nodes refer to chunk labels (e.g., noun phrases[1]: *NP*, verbal phrases: *VP*, prepositional phrases: *PRP*), pre-terminal nodes pertain to Part of Speech (PoS) categories (e.g., determinants: *DT*; adjectives: *JJ*; nouns: *NN*). Finally, terminal nodes indicate the word itself.

A constituents tree is illustrated in Figure 2.4. The image corresponds to the parse tree of the first phrase of the example in 2.3: *The code in the chip works*. From the bottom-up, looking at the node labeled *chip*, we see it is a token of type noun (pre-terminal labeled *NN*) and it belongs to a noun phrase (non-terminal *NP*) which in turn belongs to a prepositional phrase (*PP*) which finally is part of the main noun phrase of the sentence *S*. Constituents usually include a word with a prominent role: the **head** of the constituent. In practical terms, the head (or governor) is the most important word in the chunk because it determines what kind of words (either a verb, an adjective, a noun, etc.) will be joining it within the constituent.

The context that can be extracted from a constituency parse is similar to that of the lexical contexts, in that words themselves are part of the co-occurrent neighborhood. Yet, with the information from the parse tree, we can restrict the window to a chunk and heal consider only certain structural units. The context of *code* and *chip* of the parse in Figure 2.4 are shown in Table 2.2. They consist simply in th non-functional words that co-occur with each word in a given constituent, in this case a noun phrase

[1] This nomenclature is the most prevalent: the Penn Tree Bank annotation.

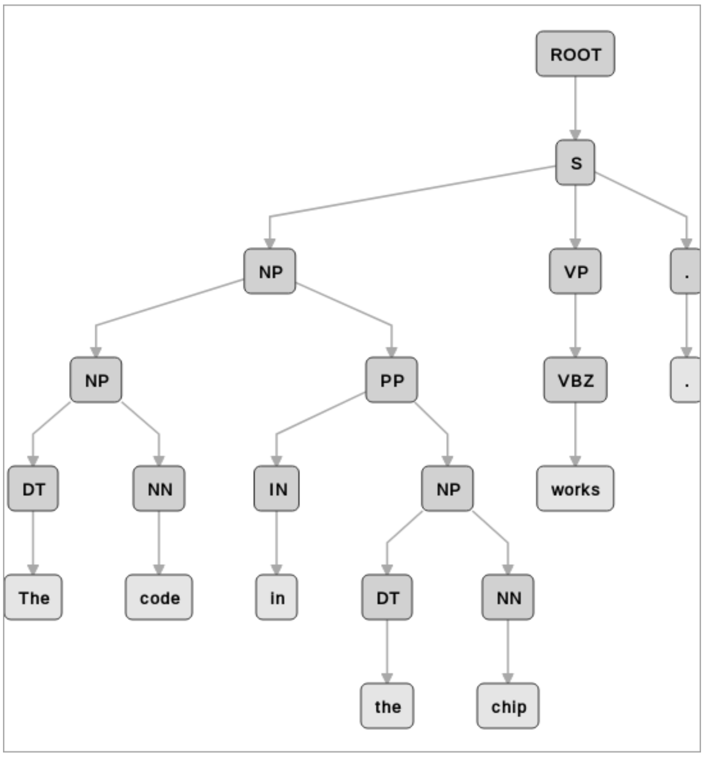


Figure 2.4: Constituents tree parse of the phrase *The code in the chip works.*

Words	Syntactic Context
code	**NP**:chip
chip	**NP**:code

Table 2.2: Syntactic contexts, based on the constituents tree in Figure 2.4, corresponding to the words *code* and *chip*, from the first phrase on Figure 2.3.

(NP).

Dependency Tree While a parse tree represent the units existing within a sentence. As a complementary approach we can also formalize syntactic information with dependency trees. This time, the syntactic structure of a sentence is described in terms of words and asymmetric binary grammatical functions between these words [Clark 2010]. The trees are directed, all nodes are terminal and they represent words and they are linked following a direction from the head to its **modifier** (or dependent). An edge thus represent one of these dependency functions which are labeled with tags that, just as PoS tags and chunk tags, describe what kind of relation exists between two words [Bird 2006]. For example, the Universal Dependencies[2] tagset [Nivre 2016, Schuster 2016], which we use in this work, includes tags such as *det*: determiner, the relation between a noun head (governor) and its determiner, *nmod*: nominal modifier, the same but with a modifier, or *conj*: conjunction, two elements connected by a conjunction.

To illustrate dependency trees, we can observe in Figure 2.5 the dependency parse of the second phrase shown in 2.3. In this particular case, the relation tags used are the "enhanced" universal dependencies by [Schuster 2016]. The difference is that relations are made more explicit by collapsing them (reducing two relation edges into a single one) and including the modifier (or adjunct) directly into the label. Consequently, they can be more useful to determine the relatedness between words.

The context that can be extracted from dependency relations varies. Still, the usual consensus is to treat the relation as the triple it is: $(\mathtt{head}, \mathtt{relation}, \mathtt{dependent})$ and based on it extract a certain type of context. In the example of Figure 2.5, a context of the word *chip*, according to [Lin 1997] would be: $(\mathtt{conj:and}, \mathtt{coil}, \mathtt{head})$. This indicates that *chip* is connected to *coil* by the conjunction *and*. More recent context definitions, such as those of [Baroni 2010, Levy 2014a, Panchenko 2017] also include the inverse relation a word participates in, i.e., if the target word is a dependent, its dependency relation is also included but indicated as "inverse". Again, using the previous example with the word *chip*, the contexts now would then be: interaction/nmod:between^{-1}; coil/conj:and . These contexts and other example can be seen in Table 2.3.

Syntactic contexts are less used than their lexical counterpart in large part due to the process of obtaining the trees discussed before. While nowadays there are several software solutions able to extract this kind of information, the process is decidedly more complex than counting words in a lexical context setting. Furthermore, the information is not 100 % accurate, as the systems are trained using human annotated banks of trees (called treebanks) which is in itself an ambiguous and hard process [Jurafsky 2009, Périnet 2015a]. This difficulty stems yet another problem with syntactic information: syntactic parsers are not easily available for languages other than English and the rest of those in the indo-european family. As a last downside, these contexts are known to generate sparse computational representations as they are more specific than simpler lexical contexts [Sahlgren 2006]. Still, syntactic con-

[2]This set of tags share a large quantity of labels with the more classic Stanford Dependencies [De Marneffe 2006, De Marneffe 2008] tagset. Briefly, universal dependencies aim to develop cross-linguistically and cross-language consistent annotations [Nivre 2016].

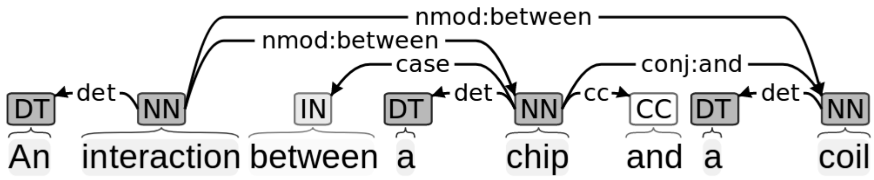


Figure 2.5: Dependency tree parse of the phrase *An interaction between a chip and a coil.*

Words	Syntactic Context
interaction	coil/nmod:between; chip/nmod:between
chip	interaction/nmod:between^{-1}; coil/conj:and
coil	interaction/nmod:between^{-1}; chip/conj:and^{-1}

Table 2.3: Syntactic contexts, based on the dependencies tree in Figure 2.5, corresponding to the words *interaction*, *code*, and *chip*, from the second phrase on Figure 2.3

texts are shown to be able to contribute more information about word's relations than simple lexical contexts [Lin 1997, Padó 2003, Turney 2010, Baroni 2010, Levy 2014a, Panchenko 2017]. In other instances, lexical contexts are shown to perform better in NLP tasks except when the syntactic contexts are extracted from very large corpora (such as Google Syntactic N-gram corpus [Goldberg 2013] containing 10^{11} tokens) [Kiela 2014].

We have presented the two main types of co-occurrence contexts that can represent a word in a corpus. In the next section we present the structures that allow mathematical and computational manipulation of the information provided by these contexts.

2.2 Vector Space Models

The whole point of determining contexts, either lexical or syntactic, for a set of words in a corpus is to be able to assess how similar their meaning is among them. This assessment of relatedness thus need to be measured by a metric in order to determine its level. The way we measure the relatedness between words relies on well-known algebraic operations, such as the dot product. In order to calculate a dot product we need vectors. It follows that to calculate relatedness among words we need to represent words by means of vectors, where each vector describe a word and each dimension a context of it.

The Vector Space Model (VSM) consists in representing textual units in a multi-dimensional space. The textual units represented are not constrained to words themselves. We may describe co-occurrent features for documents, phrases, paragraphs, or other types [Manning 1999]. A matrix is used as the structure that holds each object and its context features. Indeed, in practical terms, a VSM is then an array of real-number vectors, where each one represent a text unit and the columns describes the co-occurrent contexts the word participates in. To illustrate this, in Table 2.4 we represent words of the previous examples in a *word space*. Each entry of this matrix (called a co-occurrence matrix) represent a weight that infers the importance of the row word (or target word) with respect to the column (context) co-occurrence in a given context, within an input corpus [Jurafsky 2009]. That is, the word *code*, co-occurs once with the context indicated by the second and third columns, which in turn correspond to the words *chip* and *works*.

In the example, the weights consist merely on the frequency of co-occurrence of each word with each context. Indeed, there are still other two related parameters that affect the meaning extracted from a distributional model: the weight each cell in the matrix has, or how do each co-occurrence affect each word; and the similarity measure between vectors we will use to determine the semantic relatedness among words. For a complete analysis on a wide range of parameters affecting vector space models, see [Baroni 2010, Kiela 2014, Levy 2015].

2.2.1 Matrix Weights

The weight is an important parameter in the creation of a VSM for a NLP application. Weights can be binary, simply indicating presence or absence. They can count the number of co-occurrences of a word and the context, their absolute frequency. Weights may also be a type of discriminative measure that usually tries to give more importance to those contexts that co-occur more frequently with the target word while being less frequent with the rest of the words in the text [Jurafsky 2009, Clark 2010].

Point-wise Mutual Information (PMI) [Church 1990] and Positive Point-wise Mutual Information (PPMI) [Niwa 1994] are two popular choices to weight terms in a co-occurrence matrix [Turney 2010, Jurafsky 2017]. We describe both of them below.

Table 2.4: Matrix representation of the lexical contexts of the words appearing in the phrases of Figure 2.3. The window is the complete phrase where the word occurs.

Words	Contexts							
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
code$_{w_1}$	0	1	1	0	0	0	0	0
chip$_{w_2}$	1	0	1	1	1	0	0	0
works$_{w_3}$	1	1	0	0	0	0	0	0
interaction$_{w_4}$	0	1	0	0	1	0	0	0
coil$_{w_5}$	0	1	0	1	0	1	1	1
creates$_{w_6}$	0	0	0	0	1	1	1	1
magnetic$_{w_7}$	0	0	0	0	1	1	0	1
field$_{w_8}$	0	0	0	0	1	1	1	0

Given a co-occurrence matrix M, containing W words (rows) and C contexts (columns), where $f_{ij} \in \mathbb{R}^{W \times C}$ denotes the frequency of target word w_i frequency in the context c_j , i.e., how many times they both co-occur. $N = \sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}$ represents the sum of all the matrix cells. PMI is defined as:

$$\text{PMI}(w_i, c_j) = \log \frac{P(t_{ij}|c_j)}{P(t_{ij})P(c_j)} \tag{2.1}$$

where $P(t_{ij}|c_j) = \frac{f_{ij}}{N}$ tells us how many times the word and the context appeared together, normalized by the total context frequency. $P(t_{ij}) = \frac{f_{ij}}{N}$, and $P(c_j) = \frac{f_j}{N}$. The ratio gives us an estimate of how much more the target and context co-occur than we expect by chance.

While PMI is often used as a weighting choice, it has three main downsides [Jurafsky 2017, Levy 2015]: (1) PMI is biased towards co-occurrences of rare events, that is, a low-frequency context c co-occurring with any word w will yield a large PMI. Also (2), PMI may yield negative values, which would indicate a certain level of semantic "unrelatedness", which is not a very intuitive concept. And (3), if a context and a target word are not observed together (something that is very possible to happen because the co-occurrence matrix is sparse, we will look into that in the following paragraphs), the denominator of 2.1 is zero and thus PMI_{ij} becomes undefined.

To solve the first issue, [Levy 2015] proposes a smoothed version of PMI, defined as:

$$\mathrm{PMI}_\alpha(w_i, c_j) = \log \frac{P(t_{ij}|c_j)}{P(t_{ij})P_\alpha(c_j)} \tag{2.2}$$

with $P_\alpha(c_j) = \frac{f_j^\alpha}{N^\alpha}$, where α is a smoothing parameter that affects the contexts counts in order to alleviate the bias of PMI towards rare contexts co-occurrences : the probability of a low-frequency context c_j will be larger thanks to α, which makes the denominator of 2.2 larger, which in turns make PMI_α smaller. Thus, addressing the bias for all words when co-occurring with a low-frequency context.

The second and third inconvenient are resolved by using Positive Point-wise Mutual Information (PPMI). PPMI simply replaces all values lower than zero (including $-\infty$) by a zero:

$$\mathrm{PMI}(w_i, c_j) = \max(\mathrm{PMI}(w_i, c_j), 0) \tag{2.3}$$

2.2.2 Defining Vector Similarity

The second parameter to consider after weighting the co-occurrence matrix is how to actually determine the similarity between two word vectors.

As with weighting schemes, there are multiple metrics (defined and compared to greater detail in [Clark 2010, Ferret 2010, Kiela 2014, Clark 2015]) used in the literature to determine the similarity between two vectors. We will focus on two that are of interest to this thesis: cosine and Jaccard similarity. More types of metrics and their comparison can be found in the previously cited literature. While there does not seem to be a single best measure of similarity, we usually use the cosine similarity, as it naturally can deal with real-valued vectors. On the other hand, when dealing with binary presence-absence vectors, it is more common to use Jaccard similarity.

Cosine Similarity The cosine similarity determines the angle between two multidimensional vectors. It is simply defined as the dot product between two vectors, normalized by the multiplication of their Euclidean length [Manning 2008]. The cosine similarity is bounded between $[0, 1]$, yet we usually interpret the result in the positive space, where 0 means there is an angle of $90°$ between the two word vectors, thus no similarity at all; and 1 means there is no angle between them, so they are completely similar. Furthermore, if the weights of the matrix are non-negative values, the cosine similarity is bounded to the range $[0, 1]$. The cosine similarity is defined as:

$$\mathrm{sim}_{\mathrm{cosine}}(\overrightarrow{w_1}, \overrightarrow{w_2}) = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||\overrightarrow{w_1}|| \, ||\overrightarrow{w_2}||} = \frac{\sum_{i=1}^{C} w_{1_i} \times w_{2_i}}{\sqrt{\sum_{i=1}^{C} w_{1_i}^2} \sqrt{\sum_{i=1}^{C} w_{2_i}^2}} \tag{2.4}$$

Jaccard Index Also known as the Tanimoto index, the Jaccard index [Jaccard 1908] determines the similarity between binary vectors, it is defined, in terms of dot products:

$$\mathrm{sim}_{\mathrm{Jaccard}}(\overrightarrow{w_1}, \overrightarrow{w_2}) = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||\overrightarrow{w_1}||^2 + ||\overrightarrow{w_2}||^2 - \overrightarrow{w_1} \cdot \overrightarrow{w_2}} \tag{2.5}$$

In terms of two sets, A and B, the Jaccard index calculates the ratio between the cardinality of the intersection of two word vectors divided by the cardinality of their union: $\mathrm{sim}_{\mathrm{Jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. We prefer the definition in terms of dot products because in that way it is more straight-forward to implement it computationally.

We have been discussing vectorial space representations and their parameters (matrix weighting, similarity measure). While VSM models are the most popular to describe the semantic similarity between words, there are other structures that make it easier to model the interactions that take place among lexical units within a corpus. In that sense, in the next section we introduce the fundamentals of graph-based representations for NLP, which are part of the contributions of this thesis.

2.3 Network Models

Network[3] based models have been studied deeply during the last years in the NLP field [Mihalcea 2011]. While we can represent a graph as a matrix, and thus as a vector space model, graphs are useful representation formalism that can be applied to a large set of linguistic characteristics, from the relation between words in a text or between the features that describe them. Indeed, language being a dynamic complex system, networks provide an adequate model to represent and study the structure and evolution of linguistic systems [Choudhury 2009a].

Furthermore, based on graph theory, we can conceive efficient and sophisticated solutions to NLP tasks, such as PoS tagging, role-labeling, word sense induction

[3] We will use the notion of *network* and *graph* interchangeably during the rest of this dissertation, unless stated otherwise.

and disambiguation, chunk parsing [Mihalcea 2011]. Notably, NLP graph-based approaches are largely employed to solve unsupervised tasks, where we can expect to get insights from the data by looking at the links existing between entities; and semi-supervised problems, again by leveraging relations to propagate across the network small quantities of hand-crafted tags [Nastase 2015]. An additional non-negligible advantage of graph models are that they allow human interpretation and analysis through their visualization (nonetheless with relatively small samples of text).

Based on the advantages just mentioned, in this thesis we base our linguistic[4] model proposition on a graph-based structure. In the following paragraphs we discuss the types of networks used to represent textual data, which closely relates to the co-occurrence representations that we covered in the vector space model. Indeed, graph-based methods follow the same distributional principles as VSM. Thus, as we will see, the relationships among nodes on these networks are very similar to the types of contexts described in Section **??**.

2.3.1 Linguistic Networks

A graph is a data structure consisting of a set of vertices connected by a set of edges that model relationships between objects. Formally it is defined as a set $G = (V, E)$, where V is a collection of vertices $V = \{V_i, i = 1, n\}$ and E is a collection of edges over V, $E_{ij} = \{(V_i, V_j), V_i \in V, V_j \in V\}$.

When referring to language networks, nodes represent lexical units (most of the time words) and the edges represent the relationships between words. We present below the types of linguistic networks.

2.3.2 Types of Linguistic Networks

According to their objectives, we can consider two types of contributions in the linguistic-network literature: on the one hand, there are those approaches that investigate the nature of language via its graph representation, and on the other hand, we find those that propose a practical solution to a given NLP problem [Choudhury 2009a]. In particular, we pay attention to two aspects of a given network-based technique: (1) the characteristics of the linguistic data within the network, and (2), the algorithms used to extract knowledge from it.

[4]We will refer to a linguistic representation as an structure that holds textual units linked by their linguistic features, in this case, distributional co-occurrences.

In the following paragraphs we introduce the general categories of linguistic networks according to their type of content and relations. We will introduce these categories as well as the approaches that make use of them.

[Mihalcea 2011] defines four types of Language Networks (LN): co-occurrence network, syntactic-dependency network, semantic network and similarity network. Meanwhile, from a deeper linguistic point of view, [Choudhury 2009a] introduces broader network's categories, each having several sub-types. The main difference (in our context) between both definitions lies in the separation of categories. In [Choudhury 2009a], they conflate syntactic-dependency and co-occurrence networks into the same category: word co-occurrence networks. Similarly, they join semantic and similarity networks together and place them inside a broader category of lexical networks. The third family defined concerns phonological networks which is out of the scope of this work. In this work we will explore four categories of linguistic networks: lexical co-occurrence, syntactic co-occurrence, semantic and heterogeneous networks. Based on the previously cited works, the following paragraphs will elucidate what does each kind of network represent.

Lexical Co-occurrence Networks (LCN) In these structures, nodes represent words and edges indicate co-occurrence between them, i.e., two words appear together in the same context. The context is also defined by a window of terms. It may vary from a couple of words to a full document, although it is usually defined at sentence level. The edges' weight represent the strength of a link and is generally a frequency based metric that takes into account the number of apparitions of each word independently and together. Thus, usually the same type of weights as described are used to represent the *strength* of a relation. An example of such network is shown in Figure 2.6. The words such as *control, systems, power* co-occur in the same window of terms to the word *project*.

Syntactic Co-occurrence Networks (SCN) A Syntactic Co-occurrence Network (SCN) is very similar to a LCN in the sense that both exploit the distributional hypothesis. Nonetheless, SCNs go further by leveraging syntactic information extracted from the text. Both of the two types of syntactic parses are used: constituency-based and dependency-based parse trees. SCNs employ, most of the time, dependency trees to create a graph that relates words according to their syntactic relations. In Figure 2.7, a small syntactic network is shown. In this case, the head word *color* is related to

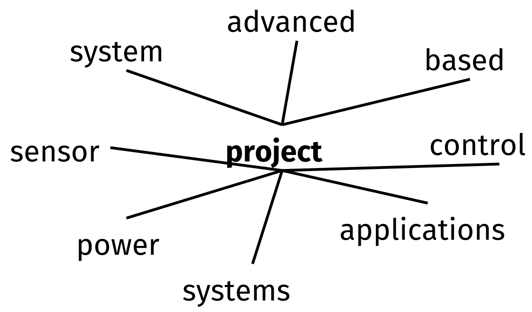


Figure 2.6: Lexical network for the word *project*.

the words *sky* and *weight* by means of a noun-modifier dependency relation (*nmod*). Other words are linked similarly according to other dependency relations.

These networks are usually used to perform WSI employing a very similar approach to those systems using LCNs. As before, the main difference being the semantic relatedness found with one or another type of network.

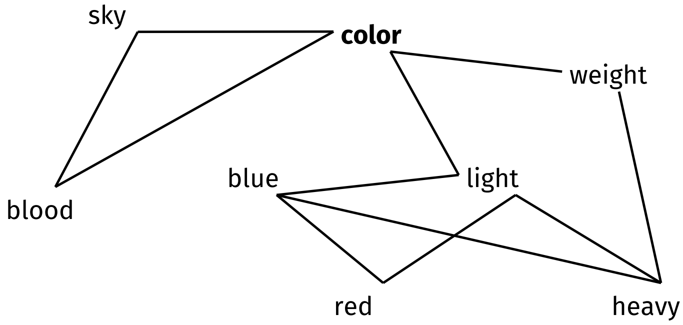


Figure 2.7: Syntactic Network of the word *color*.

Semantic Networks A Semantic Network (SN) relates words, or concepts, according to well-defined semantic relations. The classical example of a SN is the renowned knowledge base Wordnet. This network, which serves also as an ontology, contains sets of synonyms (called *synsets*) as vertices and semantic relations as their edges.

Typical semantic relationships include synonymy-antonimy, hypernymy-hyponymy, holonymy-meronymy. However, other semantic similarities can be defined. The edges are usually not weighted, although in some cases certain graph similarity measures may be used.

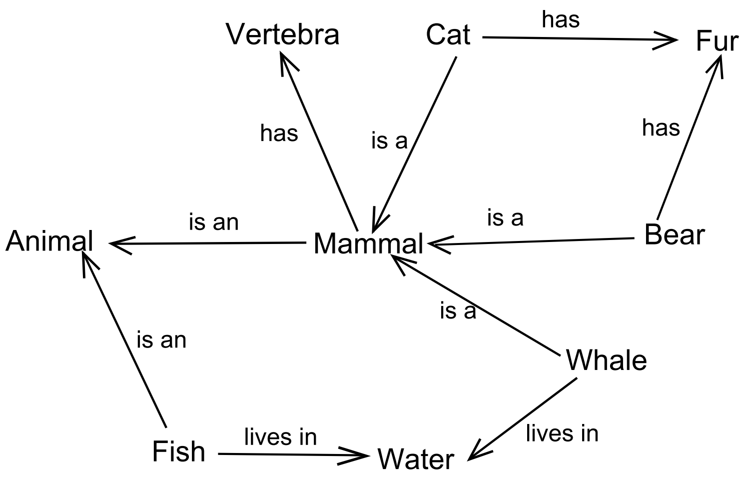


Figure 2.8: Semantic Network of the word *mammal*.

Heterogeneous Networks Until now, we have described different types of networks with single types of nodes and relations. Lately, heterogeneous networks have been defined in order to model multi-typed information in a single structure [Han 2009]. In reality, we could argue that syntactic-based and semantic networks are already heterogeneous on their own right, as both of them contain edges that represent different types of relations.

Without regards to their type, network-based structures are ultimately transformed into matrices before being treated computationally. Therefore, given that we are still modeling language (words), graphs suffer from sparsity just as vector space models. Indeed, data sparsity is an issue that affects the performance of knowledge discovery approaches [Aggarwal 2012, Périnet 2015b] applied to textual data.

2.4 Data Sparsity

Representing word's contexts as multidimensional vectors, either directly or through a graph-based structure, is indeed a straight-forward, simple and yet powerful method to transform textual data into actionable structures. The model links textual information, in te form of words and contexts, with the methods used in machine learning.

Nonetheless, there is an important issue that needs to be considered when dealing with vector space models: data sparsity. A sparse data matrix has most of its entries equal to zero. Thus, the majority of the words (rows) in the corpus are described by very few contexts (columns). This is a significant problem as on the Knowledge Discovery phase of any NLP system we aim to train a learning model that will eventually predict, classify, group our words in one way ot another. If the words are represented by a limited number of context, the learning algorithms will not be able to generalize properly. Furthermore, when testing the systems, the system will not be able to handle unseen word-context co-occurrences. This will lead to reduced performance [Phan 2008].

This phenomenon is not the consequence of using vector space models per se, as the vectors are merely a representation of word's distribution within a text. Indeed, words tend to be distributed in a text in a very predictable fashion. In any natural language corpus, most of the words occur very few times. On the other hand, very few words occur multiple times. The consequence is that most of the entries in a co-occurrence are zero because we observe very few unique word-context co-occurrences. Put differently, words co-occur most of the times with the same words and very few times with other words [Sahlgren 2006]. Given that any corpus is limited, acceptable English co-occurrences will be missing from it and their weight will be zero while they actually happen in other corpora [Jurafsky 2017]. To illustrate sparsity, Table 2.4 co-occurrence matrix contains eight words and eight contexts (each of the words), for a total of 64 entries. Among these entries, only 20 values are non-zero, and more importantly each word is only represented by 2.5 contexts in average. While it is a toy example, and 20 non-zero entries from 64 in a matrix would hardly be considered sparse, it reflects what actually happens with larger corpora, as this problem is corpus-size independent (and even more important with smaller corpora [Périnet 2015a]).

In order to deal with the distributional representation sparsity, we discuss below multiple techniques used in the literature [Sahlgren 2006, Ratinov 2009, Molino 2017] that aim to alleviate matrix sparsity. In the following paragraphs we discuss explicit and implicit representations. We note that by explicit representation we refer to the classic weighted co-occurrence matrix, introduced before, where cell's values represent target word in a specific context. It is explicit in the sense that each column

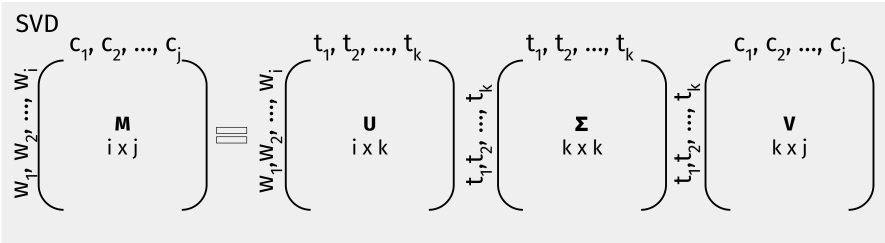


Figure 2.9: Factorization of the matrix M using SVD. Three matrices are generated: U, Σ, and V. U is the one that really interest us, it contains $|W|$ words and $k, k < C$ columns.

directly represent a context seen in the input corpus.

Implicit Representations These methods aim to reduce the sparsity as well as the dimension of the co-occurrence matrix while keeping latent (or implicit) features that best represent the original spaces. These techniques reduce the feature space to k dimensions (usually k is much smaller than the original number of columns) and thus the dimensions are no longer directly interpretable.

Classic examples of implicit representations include Latent Semantic Analysis and Linear Discriminant Analysis, both methods generate a word-document (terms in rows, documents in columns) co-occurrence matrix, weighted by tf-idf. Then, the matrix is reduced into a smaller dimension by means of a Singular Value Decomposition (SVD). SVD keeps the top k singular values that maximize the variance among the features, k being smaller than the original number of dimensions.

Recalling the definition of a co-occurrence matrix M (where $f_{ij} \in \mathbb{R}^{W \times C}$), SVD factorizes M into three matrices: two orthogonal matrices U and V; and one diagonal, containing ordered eigenvalues Σ, and V, such that $M_{i \times j} = U_{i \times k} \Sigma_{k \times k} V_{k \times j}$, as shown in Figure 2.9. We are interested in matrix $W = U_k \Sigma_k$, as it contains the words represented by k singular values, and we thus can substitute M with it. In the same fashion, we can obtain the same reduced representation for the contexts by using directly V_k. [Levy 2015] found that, empirically, we can even dismiss the eigenvalues matrix V_k in W and obtain better general performance.

A similar approach, that of the Non-negative Matrix Factorization [Lee 2001] (NMF) is also used to find latent dimensions in the co-occurrence matrix. As shown in Figure 2.10, NMF factorizes M into only two matrices: $M_{i \times j} \approx W_{i \times k} H_{k \times j}$. The first matrix W contains the words represented by a lower dimension k. On the other

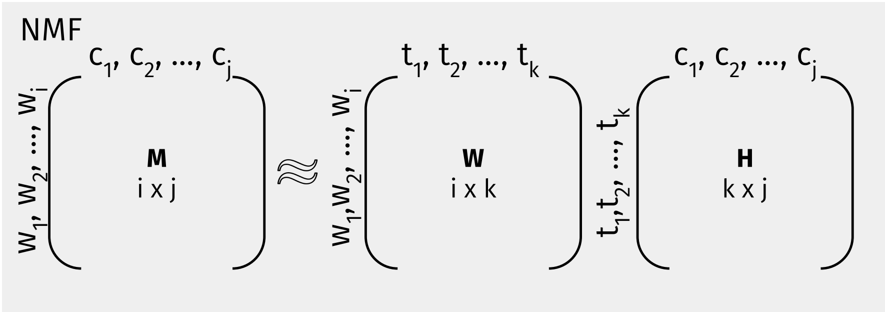


Figure 2.10: Factorization of the matrix M using NMF. Two matrices are generated: W and H. W contains the words in a k reduced space. H contains the contexts in the reduced space.

hand, H represents the relation between the contexts and the newly induced dimensions. This procedure solves certain issues with SVD, such as containing only positive values in the factorized matrices, which is more intuitive when dealing with text, i.e., the original matrix is the result of the addition of the factorized values. Also, depending on the implementation, NMF is able to solve the Kullback-Leibler divergence (instead of the Euclidean distance, as SVD) as objective function. This measure is better suited for textual data, for example in the factorization of both lexical and syntactic co-occurrence matrices [Van de Cruys 2011].

Lastly, the most popular implicit approach nowadays is the use of distributed representations, also known more popularly as word embeddings. The goal is to represent words with a rich dense matrix and relatively few dimensions. These representations are generated from very large bodies of text, creating a regular co-occurrence matrix and calculating, via either a neural network approach [Bengio 2003, Collobert 2011, Mikolov 2013] or a matrix factorization [Pennington 2014, Levy 2014b], a lower dimensional and dense matrix representation. This area is fast developing, with new methods and propositions every few months. Furthermore, the community is not really certain still on which approach is better, either depending exclusively on co-occurrence frequencies and matrix factorization methods, or on sophisticated prediction approaches coupled with neural networks techniques [Baroni 2014, Levy 2015].

Enriched Explicit Representations While implicit representations modify the interpretation of the columns of the co-occurrence matrix so that they are no longer

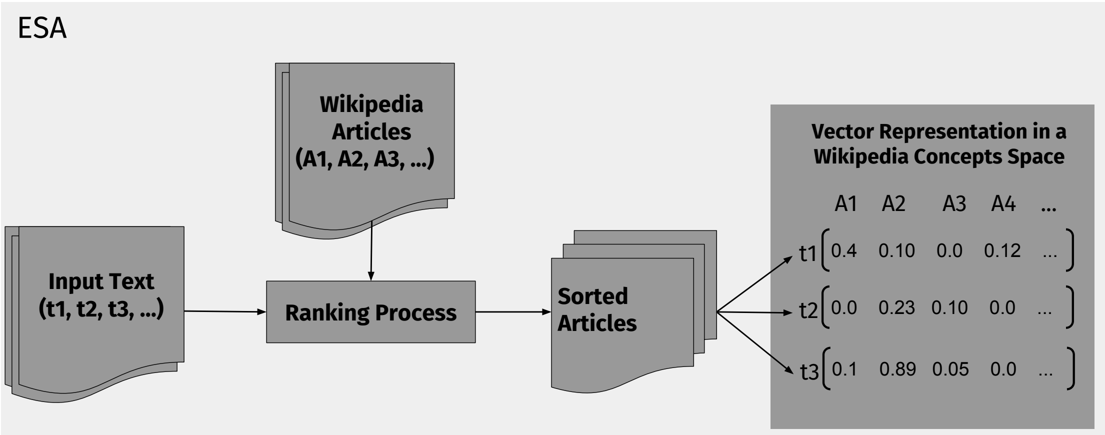


Figure 2.11: Block description of the ESA process by [Gabrilovich 2007]. The method ranks Wikipedia articles according to an input document. The document is then described by a weighted vector, where each dimension is a Wikipedia article, or concept.

directly interpretable[5], enriched explicit representations modify the meaning of the contexts with the addition of replacement of information to them in order to better cover a larger set of target words and thus reduce the sparsity of the matrix. More importantly, the new contexts are still *natural* concepts in the sense that are still easily interpretable by us humans.

For example, leveraging the large size of the English Wikipedia corpus, the Explicit Semantic Analysis [Gabrilovich 2007] aims to augment text presentation by describing words in terms of Wikipedia concepts. The method yields weighted vectors of Wikipedia concepts that best represent, according to a document ranking, an input text (see Figure 2.11). The advantage of this representation is that indeed the concepts are human interpretable, not like implicit models, and it covers a very large body of information, the Wikipedia corpus. As a downside, vectors using this representation tend to be very large representing a computational challenge. What is more, the representations are dependent on the size of the Wikipedia corpus used. Vectors based on non-English versions may suffer from the reduction of textual data, as English Wikipedia is considerably larger than most languages.

Fusion techniques are another option to enrich and eventually densify feature spaces. While not largely applied in the textual analysis domain as in the multi-media information retrieval and phonetics fields [Ozkan 2010, Ah-Pine 2015], they represent a set of simple yet powerful methods to merge information and create more power-

[5] Still, the meaning can be inferred to a certain level with the help of the reduced matrices.

ful representations. Indeed, these operations are based on the combination of feature spaces to obtain representations that leverage the complementarity of the original spaces. [Bruni 2014] has employed fusion methods to generate enriched multi-media semantic spaces while blending images and text to define the similarity among words. While feature spaces are combined, these techniques do not modify the meaning of the contexts and they remain interpretable. In this thesis, as we described in the introduction of this work, and as we will see later on, we also use these techniques to fight data sparsity by combining linguistic spaces, without resorting to other types of data. Namely, by leveraging the properties of two distributional representations, both lexical and syntactic co-occurrences, we can get more dense and stronger word representations.

2.5 Positioning

We have introduced four axis that define the work that we carry out in this thesis. Our propositionas are based on the distributional hypothesis: we assume that words that share a common context are related. The relation is determined by the type of context: whether lexical or syntactic properties. If we choose a lexical context, the size of the window (how many words to the right, to the left) should be determined according to the ultimate goal of the NLP task at hand. This window has an effect on the semantic properties of the relatedness among words: the shorter the window, the closer we get to a synonymy similarity, i.e., we may be able to intercheange words one for another and keep a coherent phrase. The larger the window, the more topical the relatedness is, i.e., words are related in a broader sense. On the other hand, when using shorter windows we indeed approach to the relatedness provided by syntactic relations [Sahlgren 2006], which relate words that participate in the same syntactic dependency functions, also known as functional similarity [Levy 2014a]. Again, this implies that words could be substituted without losing sense within a phrase.

These contexts need to be represented computationally in order to perform some sort of analysis over them. Vector space models allow for this representation. By mean of co-occurrence matrices, we keep track of what words are seen with what contexts within a corpus. These counts may then be affected by some weight that determines the relevance of said co-occurrences in terms of uniqueness in terms of the whole set of co-occurrences found. Once these word's vectors are created, we can thus finally calculate a degree of relatedness between them by employing vector similarity metrics, notably the cosine similarity (for real-valued vectors) and Jaccard (binary valued vectors) metrics.

While matrices are the fundamental structure used in computational operations, we can model the links among words intuitively with graph-based structures. Indeed, by modeling text as graphs we gain access to established graph-theory techniques which helps us elucidate the inner structure of textual data.

Whether its vector based or graph based, a textual, explicit, and distributional representation will be sparse. There are too many words in a text and its assured that, while they could occur in other texts, they will not occur in a single text. This becomes an important problem with NLP systems: words are described by only a small set of features.

Chapter 3

Fusion-Enriched Hypergraph Linguistic Model

Abstract. *In the previous chapter we presented the theoretical notions used to represent text with a distributional approach, that is, the parameters, the models to implement them and the problems that naturally arise from these kinds of representations. In this chapter we introduce and define the first set of our contributions. Briefly, we present a linguistic framework to represent textual data. Feature fusion techniques are then applied over this network in order to better leverage the data contained in it while addressing the sparsity issue.*

We organize this chapter in four parts: we present a brief state of the art on how the information contained in linguistic graphs is used for WSD/WSI and NER. Secondly, we introduce our model. Thirdly, we present the method used to combine the data held in our mode, specifically using feature fusion techniques. Finally, we materialize the proposed model by transforming an English Wikipedia based corpus into our proposed framework.

Contents

3.1 Introduction

In the previous chapter we covered the details concerning the parameters regarding the construction of a distributional representation model, as well as its challenges. The challenges that we will address in this chapter are two: (1) how to organize heterogeneous textual information within a single linguistic resource, and (2), how to leverage said information to obtain complementary representation spaces, while taking into account the feature sparsity issue that is characteristic of textual data.

A fusion-enriched hypergraph linguistic model is the first contribution of this thesis. The model consists on two componentsw which address two research questions each: the issue of making sparsity less severe and leveraging different types of features by using a single feature representation space. We will describe our motivations and its characteristics in the following paragraphs. We can appreciate the block diagram of the ensemble of our propositions on Figure 3.1. On the top, our enriched linguistic model proposition, which is the focus of this chapter. On the bottom and grayed out, the application of said model in NLP tasks, described thoroughly in Chapter 4.

The model we present here entails three important characteristics: firstly, the possibility to leverage different[1] types of information. Secondly, as the words will be linked together, there is an inner structure that will emerge from the model and which we exploit in our experiments. Thirdly, given that we treat unstructured text data, the relations (or features) between words are sparse, this is alleviated by combining features via fusion techniques.

Our network is based on the distributional hypothesis, as described in the previous chapter. As co-occurrence features, we select both lexical and syntactic contexts, indeed creating a linguistic resource that hold both types of information in order to get a complementary insight of words' relations. Our network sets a lexical window, a co-occurrence weighting, and the definition of similarity between vectors according to two semantic NLP tasks we treat in the following: word sense induction and disambiguation and named entity recognition. Nonetheless, the parameters chosen can be

[1]We use three in our model definition: syntactic, lexical and what we will call standard features (explained later on).

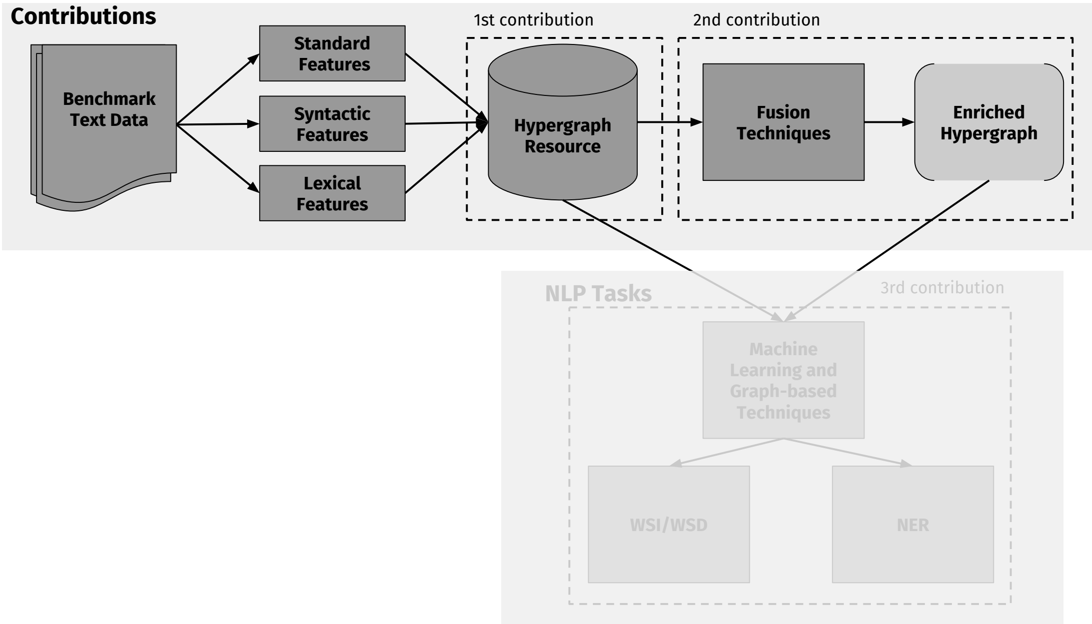


Figure 3.1: Modular description of the contributions of this thesis. Notably the fusion enriched hypergraph model we propose and in the bottom (grayed out) the NLP tasks applications based on said model.

easily changed. Regarding representation, we base our model on a graph-based structure which holds the words as nodes and the linguistic relations as edges. Being able to use the structure of the graph is useful to solve NLP semantic tasks [Nastase 2015]. The following paragraphs will describe this structure in detail.

In order to contextualize our proposition, we begin with the state of the art on how linguistic networks are employed in the literature. We are interested on those networks previously covered: lexical, syntactical and semantic co-occurrence networks. We give emphasis to two aspects: how the inner structure is used to solve the tasks at hand, and what type of graph-based algorithm is used. The literature on graph-based approaches for NLP is vast, we thus focus specifically on semantic tasks, notably Word Sense Induction and Disambiguation, and Named Entity Recognition. We chose these two tasks as we focus on both of them for the rest of our contributions. Afterwards, we introduce our model, how to build it and its properties. We focus on the description of hypergraphs and how they allow us to join together different linguistic networks. Next, we introduce the concept of fusion techniques used to combine the linguistic features and obtain an enriched representation.

In that sense, this chapter presents a word representation model based on a generalization of graphs (we employ hypergraphs) that contains different types of linguistic data to characterize the terms (or words) contained within. These representations are then combined with fusion techniques in order to ideally get an enriched and complementary feature space for each word. Finally, as a proof-of-concept, we describe the characteristics of the transformation of an English Wikipedia-based corpus into the framework we propose, a hypergraph model and its single enriched representation produced via fusion techniques. We show an example of the results produced by these fusion methods.

.

3.2 Linguistic Networks in Semantic NLP Tasks

We present here an overview of linguistic network's related work. We discuss what and how different methods are used with language networks to extract knowledge from their structural properties. Finally, we discuss the limitations of the current propositions and the advantages of the model we propose.

Using Lexical Co-occurrence Networks Lexical Co-occurrence Networks (LCN) are popular since they do not require any special treatment to obtain them, just the input corpus. It is then natural that truly-unsupervised[2] word sense induction approaches leverage these type of networks, and in return, the distributional hypothesis, to automatically discover senses for a given target word. That is why several WSI methods [Véronis 2004, Klapaftis 2007, Navigli 2010, Klapaftis 2008, Di Marco 2011, Jurgens 2011] are tightly related to LCNs. The cited works use a LCN as described before while other works such as [Navigli 2007, Qian 4 08] represent, as we do,u the co-occurrence by means of a hypergraph scheme. In short, a hypergraph structure is a graph generalization where an edge (called hyperedge) can link more than two vertices per edge and thus it is able to provide a more complete description of the interaction between several nodes [Estrada 2005].

In their paper, given an input document with several contexts for each target word, they first group together the contexts via a topic-modeling technique. Thus, each context is assigned a particular group (in this case, a topic). Secondly, a hypergraph is built where the vertices represent contexts and the hyperedges link two nodes together

[2]Without the need of human-crafted semantic networks.

if they share the same topic. Thirdly, the hypergraph is clustered and the words of each context (of each node) are used to build vectorial representations

WSI systems generally perform four steps. Given an input text with a set of target words and their contexts (target words must have several instances throughout the document to cluster them), the steps are the following:

1. Build a LCN, assigning tokens as nodes and establishing edges between them if they co-occur in a given context (usually if they both appear in the same sentence),

2. Determine the weights for each edge according to a frequency metric,

3. Apply a graph clustering algorithm. Each cluster found will represent a sense of the polysemous word, and

4. Match target word instances with the clusters found by leveraging each target word context. Specifically, assign a cluster (a sense) to each instance by looking at the tokens in the context.

As with semantic networks, not only WSD or WSI can be solved with LCNs. Finding semantic associated terms in a corpus is a critical step in several NLP systems. This task is solved in the system proposed by [Liu 2011]. They also use a LCN although instead of a co-occurrence graph, they also employ a co-occurrence hypergraph, where nodes represent words and edges describe co-occurrence at the paragraph level. In this work, they use such structure to find related terms in a given corpus. In order to do it, they mine the hypergraph as in a frequent itemsets problem, where the items are the words from a text. The method consists in first finding similar itemsets by means of measuring similarity between nodes. Once the 2-itemsets are found, they induce a graph from the original hypergraph by drawing edges between nodes that have a similarity superior to an arbitrary threshold. Lastly, in order to find k-itemsets ($k > 2$), the find either complete or connected components in the induced graph.

As with WSD, while the LCNs used are mostly the same among approaches, there are certain moving parts that make up the difference between WSI approaches. The most common differences that can arise are:

- The clustering algorithm to find senses in the LCN graph.
- The technique used to match context words to clusters.
- The weight used in the graph edges.

Using Syntactic Co-occurrence Networks A network representation that is on the border line between being a LCN and a SCN is that of [Bronselaer 2013]. They propose a graph document modelization. In their network, nodes represent words and edges their co-occurrence, as any LCN. Still, their graph resembles a SCN because the edges may represent one of three types of words: either prepositions, conjunctions or verbs. As a result, they need to first extract syntactic information from a document, namely the part-of-speech tags of each word. They find the most relevant words of a given text by ranking the nodes of the graph. The words that best represent a document can be used to summarize it, as they show in their work.

Approaches based on SCN are rarely used in WSD or WSI systems, and therefore they are an interesting research avenue to explore.

Using Semantic Networks Word sense induction is indeed a task usually solved using semantic networks, specially Wordnet (and to a lesser extent, BabelNet) [Mihalcea 2004, Sinha 2007, Tsatsaronis 7 01, Navigli 2007, Agirre 2008, Klapaftis 2008, Agirre 2009, Klapaftis 2010, Silberer 2010, Moro 2014]. Given an input text with a set of ambiguous target words to process, these approaches follow a two-step algorithm:

1. Link target words (usually nouns, skipping stop-words and functional words) with their corresponding sense (or synset in the case of WordNet-like dictionaries) and extract their vertices and edges into a new, smaller, SN.

2. Apply a node ranking technique, usually a random-walk-based method, and select, for each ambiguous word in the input text, its top ranking synset node as the correct sense.

The amount of edges a SN has grows depending of the size of the version of Wordnet used or the level of polysemy of a given word. In order to avoid redundancy or contradiction between linking nodes, [Mihalcea 2004, Navigli 2007] applied pruning techniques to avoid *contamination* while calculating ranking metrics in order to define a pertinent sense. Regarding edge similarity measures, in [Sinha 2007, Tsatsaronis 7 01] they test some metrics individually and also combined. They found that the best results are indeed obtained when several metrics are used at the same time.

Concerning the measure of semantic affinity between two terms, in [Yeh 2009] they quantify word similarity by means of projecting them into a Wikipedia space. First, they represent each word by a vector representing its most pertinent pages, and then they calculate a vectorial similarity measure between them.

Finally, extracting entities from text can also benefit from the use of SNs. The work proposed by [Kivimäki 2013] aims to extract technical skills from a document. Again, using Wikipedia as SN, they first represent each article and the input document in a token vector space model. Next, they find the document top 200 similar pages by calculating the cosine similarity between the document and each page. This serves to extract a Wikipedia subgraph which is used to calculate the most relevant pages for the entry document. Finally, the top pages are filtered by means of selecting those articles that actually represent a skill using a fixed list of skill-related tokens. Once again, the nodes represent Wikipedia articles and the edges the hyperlinks that join them.

The cited methods vary in how they make use of their SN, not so much in the network per se. These differences boil down to three aspects:

- Type of relationship implied by the edges linking the nodes of the network,
- The algorithm used to rank the nodes after the semantic network is built, and
- The weight assigned to each edge.

Using Heterogeneous Networks Even though this kind of structure seems to open new avenues of research in the semantic analysis domain, only few explicitly take advantage of them, as is the case of [Saluja 2013]. In their approach, they build a graph that links together features with words, and discover similarity measures that leverage the multi-typed nature of their network.

3.2.1 Algorithms used in Linguistic Networks

We have discussed until now the different types of networks from a content point of view. In this subsection, we address the details of the graph-based algorithms used to solve semantic NLP tasks. In this section we cover the details of four different types of graph algorithms.

Edge Weights We begin by describing the metrics used to determine similarity between nodes, usually stored as edge weights. As stated in the previous sections, most of the metrics are frequency based, specially when dealing with LCNs. The main idea of these measures is to assign a weight that decreases as the association frequency of the words increases. Among these measures, the most popular are the Dice coefficient [Navigli 2010, Di Marco 2011, Di Marco 2013], normalized pointwise mutual information [Hope 3 06], and a graph-adapted tf-idf variant [Tsatsaronis 7 01] which aims to give importance to frequent edges while also favoring those that are rare.

Edge weights can also be calculated when the vertices of a network do not represent words. Such is the case of [Klapaftis 2010], where nodes represents a target word context (set of tokens around an ambiguous word). This time the Jaccard index is used to quantify similarity between them while considering how many words are shared between a pair of context nodes.

When the nodes represent synsets (or concepts), certain approaches leverage only the intrinsic nature of the network connections, leveraged by random walk algorithms, without explicitly using weighted edges [Mihalcea 2004]. On the other hand, there are techniques that assign a frequency-based weight to represent the importance of a semantic relation, particularly those found in the reviews by [Sinha 2007, Navigli 2007], where several weights are tested.

A more sophisticated approach to edge weighting is proposed in [Saluja 2013] where they employ custom-defined functions in order to learn the most appropriate edges' weights for a given set of seed vertices inside a network. The main idea is to enforce *smoothness* (keeping two nodes close if they have related edges) across the network.

As a way to rank edges according to their importance, the ratio of triangles (cycles of length 3), squares (cycles of length 4), and diamonds (graphs with 4 vertices and 5 edges, forming a square with a diagonal) in which an edge participates are calculated [Navigli 2010, Di Marco 2013]. Once the top edges are found, they create a subgraph containing only these edges (and its corresponding vertices).

Finally, instead of applying weights to edges, a case where nodes are weighted can be found in [Kivimäki 2013]. They measure and remove popular nodes in order to avoid their bias during the application of their random walk approach.

Graph Search Usually, in a WSD approach, the first step to follow is to build a graph from a LKB. The goal is to explore the semantic network and find all the senses linked to those found in the context of an ambiguous word. Aside from custom search heuristics applied by certain works [Agirre 2006, Sinha 2007, Agirre 2009], researchers also use well-known graph techniques such as depth-first search [Navigli 2007], breadthfirst search [Agirre 2008] and even the Dijsktra algorithm to find the group of closest senses in the network [Matuschek 3 05].

Node Connectivity Measures A Connectivity Measure (CM) determines the importance of a node in a graph according to its association with other nodes. In most cases its value ranges from zero to one, where the 0 indicates that the node is of minor importance while 1 suggests a relatively high significance. Nowadays, the most widely used measures are those based on random walks.

A Random Walk (RW) can be simply defined as the traversal of a graph beginning from a given node and randomly jumping to another in the next time step.

PageRank [Brin 8 04], the popular random walk based algorithm is used commonly in WSD. The recursive intuition of PageRank is to give importance to a node according to the PageRank value of the nodes that are connected to it. Nonetheless, as a regular random-walk algorithm, in PageRank the probability distribution to change from a node to another is uniform. In such case, the jumps a random walker performs depend solely on the nature of the graph studied. Among the approaches surveyed, those that use the most PageRank are those that solve word sense disambiguation [Mihalcea 2004, Agirre 2006, Navigli 2007, Silberer 2010]. These approaches make a conventional use of PageRank: they apply it and rank nodes to select the most appropriate senses for ambiguous words. Still, there are some improvements over the classical use of PageRank in WSD. Some techniques employ a different version of PageRank called Personalized PageRank (or PageRank with restart [Murphy 2012] or PPR) were a random walker may return to a specific starting node with certain probability rather than jumping to a random node. This formulation allows researchers to assign more weight to certain nodes. For example, in [Agirre 2009] they are able to use the complete Wordnet graph as their SN. They do this by directly adding context words of a polysemous token into Wordnet and then giving a uniform initial distribution to only these nodes. In this way, they force PageRank to give more importance to the context words without the need of extracting a subgraph from the SN. In [Moro 2014] they apply the same technique to obtain a *semantic signature* of a given sense vertex. After applying PPR, they obtain a frequency distribution over all the nodes in the graph. The so-called semantic signature consists in those nodes that were visited more than an arbitrary threshold and that best represent an input sense node.

There are other methods which share the properties of random walk approaches. In [Tsatsaronis 7 01, Kivimäki 2013] they apply a method known as spreading activation. This algorithm aims to iteratively diffuse the initial weights of a set of seed nodes across the network. Specifically, once a weighted semantic network is built, they *activate* the nodes representing the context senses, assigning a value of 1, while *deactivating* the rest by setting them to 0. They determine the most pertinent senses to the input nodes by storing, for each of them, the last active sense node with the highest activation value.

Beyond WSD and into the task of determining word similarities, we found the work of [Yeh 2009], where they calculate a semantic similarity between a pair of words while leveraging a Wikipedia SN. For each word, they apply PPR to find the articles that best represent a word. In [Saluja 2013], they also employ PPR to find synonym words given a word-similarity matrix and a new unknown word (also known as out-of-vocabulary word). They link the new word to its corresponding feature nodes and they normalize the similarity matrix to use the weights as probabilities and thus bias the random walk. In [Kivimäki 2013] they use centrality measures to determine the most relevant nodes in a SN and then, in contrast with most approaches, remove them from the graph in order to not bias their graph algorithms.

With regard to other CMs, there are more elementary alternatives to determine the importance of a node. For example, the approaches of [Véronis 2004, Klapaftis 2007, Liu 2011, Bronselaer 2013, Moro 2014] successfully use the degree of a node, or other metric, to determine its importance in a network.

Graph Clustering/Partitioning Graph clustering is defined as the task of grouping the vertices of a graph into clusters while taking into consideration its edge structure [Schaeffer 7 08]. As previously mentioned, graph-based word sense induction relies most importantly in the graph clustering step where the actual senses of a word are inferred.

In this subsection we also consider subgraph extracting techniques which are exploited to find separated groups of words and thus induce senses. In this context we found the approaches of [Véronis 2004, Silberer 2010]. These systems make use of both the Minimum and Maximum Spanning Trees algorithms (MinST and MaxST, respectively) as a final step to disambiguate a target word given its context. Meanwhile, both [Liu 2011, Qian 4 08] use the Hypergraph Normalized Cut (HCT) approach, a hypergraph clustering method based on minimum cuts, to induce senses.

Most of the reviewed approaches employ state of the art techniques [Klapaftis 2008, Klapaftis 2010, Jurgens 2011, Hope 3 06]. Specifically, they utilize Chinese Whispers (CW) [Biemann 2006], Hierarchical Random Graphs (HRG) [Clauset 2008], Link Clustering (LC) [Ahn 2010], and MaxMax (MM) [Hope 2013] respectively.

Briefly, CW is a randomized graph-clustering method which is time-linear with respect to the number of edges and does not need a fixed number of clusters as input. It only requires a maximum amount of iterations to perform. HRG, being a hierarchical clustering algorithm, groups words into a binary tree representation, which allows to have more in-detail information about the similarity among words when compared to flat clustering algorithms. Regarding LC, instead of clustering nodes, this procedure groups edges. Thus it can identify contexts related to certain senses, instead of finding groups of words as most approaches do. Finally, MM, is able to assemble words into a fixed cluster (hard clustering) or allow them to be in several groups at the same time (soft clustering). It shares certain characteristics with CW: they are both methods that exploit similarity within the local neighborhood of nodes and both are time-linear. Nonetheless, a key difference is that CW is not deterministic, while MM is, thus MM will find always the same clustering result for the same input graph.

3.2.2 Discussion

We have covered the network attributes of several approaches on semantic related NLP tasks. A summary of these strategies is shown in Table 3.1. In this section we shortly discuss the reviewed articles from a modelization perspective as well as looking at the evolution of the approaches used to solve the word sense disambiguation and induction tasks.

Regarding WSD approaches, we see that the use of a lexical knowledge base, such as Wordnet, is pervasive in this task. Indeed, new resources, such as BabelNet, solves to some extent the fixed (no new senses are included automatically) nature of this type of resources by leveraging the always evolving knowledge of Wikipedia. Indeed, in the recent years, entity linking has emerged as a related task to WSD. It takes even more advantage from bases that combine both Wordnet and Wikipedia, such as BabelNet. On the other hand, WSI, while being a more flexible approach (language and word-usage independent, does not require human-made bases) for solving WSD, its results are tightly linked to the quality of the clustering algorithm used. With respect to the networks' modelization, we find that few approaches deal with syntactic attributes.

Table 3.1: Survey summary table.

Approach	**Network Type**				**Algorithms**			
	Semantic	Lexical	Syntactic	Heterogeneous	Edge Wts.	Graph Search	Connectivity Meas.	Graph Clust.
Veronis, 2004 [Véronis 2004]		x					x	x
Mihalcea et al., 2004 [Mihalcea 2004]	x						x	
Agirre et al., 2006 [Agirre 2006]		x				x	x	
Sinha and Mihalcea, 2007 [Sinha 2007]	x					x		
Navigli and Lapata, 2007[Navigli 2007]	x					x	x	
Tsatsaronis et al., 2007 [Tsatsaronis 7 01]	x						x	
Klapaftis and Manandhar, 2007 [Klapaftis 2007]		x			x		x	
Klapaftis and Manandhar, 2008 [Klapaftis 2008]		x			x			x
Agirre and Soroa, 2008 [Agirre 2008]	x					x		
Agirre and Soroa, 2009 [Agirre 2009]	x					x	x	
Klapaftis and Manandhar, 2010 [Klapaftis 2010]		x			x			x
Navigli and Crisafulli, 2010 [Navigli 2010]		x			x			
Silberer and Ponzetto, 2010 [Silberer 2010]		x					x	x
Di Marco and Navigli, 2011 [Di Marco 2011]		x			x			
Jurgens, 2011 [Jurgens 2011]								x
Di Marco and Navigli, 2013 [Di Marco 2013]		x			x			
Hope and Keller, 2013 [Hope 3 06]			x		x			x
Moro et al., 2014 [Moro 2014]	x						x	
Qian et al., 2014 [Qian 4 08]	x	x						x
Yeh et al., 2009 [Yeh 2009]	x						x	
Liu et al., 2011 [Liu 2011]		x					x	x
Matuschek and Gurevych, 2013 [Matuschek 3 05]	x					x		
Bronselaer and Pasi, 2013 [Bronselaer 2013]			x				x	
Kivimäki et al., 2013 [Kivimäki 2013]	x				x		x	
Saluja and Navrátil, 2013 [Saluja 2013]		x		x	x		x	
25	11	12	2	1	9	6	14	8

We believe that finding semantic similarities can be improved by adding syntactic information not only while using dependency relations but also by leveraging the constituency tree of each word. Moreover, using syntactic data along with semantic and/or lexical co-occurrences takes us into the heterogeneous network domain which has not been addressed in most of the approaches covered. Being able to design new similarity metrics that deal with different types of information opens new avenues of research in the semantic similarity domain. Finally, concerning the algorithms employed, few approaches make direct use of the graph Laplacian representation. New similarities could be defined using the Laplacian as a starting point.

Taking into account the described opportunities of research, in the following section we propose a hypergraph modelization of a linguistic network that aims to solve some limitations stated above.

3.3 Proposed Model: Fusion-Enriched Hypergraph Linguistic Network

As stated before, our model consists on two parts (and two contributions). The first one, an hypergraph model that holds different types of linguistic relations extracted from a corpus. And the second one, the combination of linguistic features in order to generate a less sparse, enriched representation.

In this section we focus on the first part, the hypergraph model. We note that for the sake of simplicity we limit ourselves to lexical and syntactic information. The model in essence holds two different networks, one for each type of relations. They are both unified by means of a hypergraph structure.

3.3.1 Hypergraph Linguistic Model

Formally, a hypergraph [Berge 1985] is a graph generalization that allows more than two vertices to be linked by a single edge. We call $\mathcal{H} = (V, E)$ a hypergraph with the vertex set V and the hyperedge set E. Let V denote a finite set of objects, and let E (the hyperarcs or hyperedges) be a group of subsets of V such that $V = \cup_{e_j \in E} e_j$. A *weighted hypergraph* is a hypergraph that has a positive number $w(e)$ associated with each hyperedge, called the *weight* of the hyperedge e. A *weighted hypergraph* is then denoted by $\mathcal{H} = (V, E, w)$. A hyperedge e is said to be *incident* with a vertex v when $v \in e$. As one can see, as in regular graph theory, the adjacency is referred to the

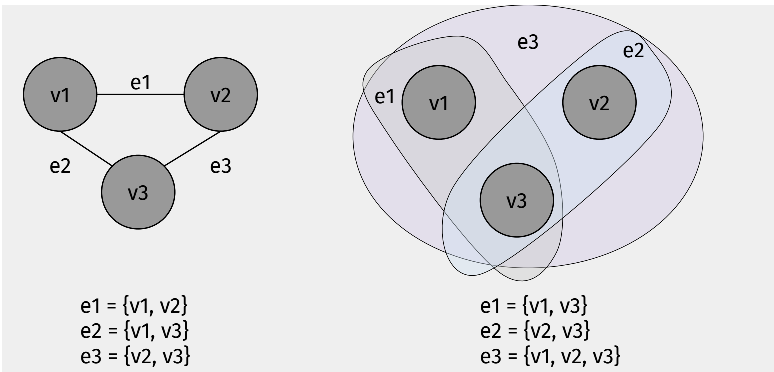


Figure 3.2: A graph (to the left) compared to a hypergraph (to the right). The edges of the hypergraph (hyperedges) may hold more than two nodes at once, relaxing the constraint of graphs' binary relations. A hypergraph can be seen as a set of n-ary sets: $E = \{\{v1, v3\}, \{v2, v3\}, \{v1, v2, v3\}\}$.

elements of the same kind (vertices vs vertices, or edges vs edges), while the incidence is referred to the elements of different kind (vertices vs edges).

Building upon previous linguistic representations [Klapaftis 2007, Liu 2011, Qian 4 08], our model is indeed based on the use of a hypergraph. Their single most important difference with regular graphs, being able to relate more than two vertices at the same type, allows for a better characterization of interactions within a set of individual elements (in our case, words) [Heintz 2014]. Indeed, our hypergraph modelization initially integrates four types of relations between tokens: sentence co-occurrence, part-of-speech tags, words' constituents data and dependency relations in a single linguistic structure. These relationships were chosen because its is relatively easy to obtain them for high-resource languages. These features can be seen as building blocks for NLP models. Extracting deeper language features would implicate relying even more on general domain systems. In any case, our goal is to arrive to more complex annotations (e.g., named entities) from the selected features and relations. Indeed, as we discussed before, different types of contexts gives us different types of similarities. Recall that a shorter lexical window approximates us to a syntactical context. That is why we decided to keep a lexical context at sentence level, so that it may complement the distributional semantic information provided by the dependency functions context as well as the phrase-constituency syntactic context. In short, we aim to cover three levels of possible semantic relatedness via three levels (in terms of the size of the neighborhood of a target word) of distributional co-occurrences :

a short range with dependency functions, a medium range with phrase constituency membership, and a longer range with sentence lexical co-occurrences. The intuition is that when solving NLP tasks, having direct access to these three semantic spaces will help to determine a more appropriate meaning's relation between words.

Construction In our case, the set of words in the corpus are the set of nodes V, and the set of hyperedges E represent the relations between nodes according to different linguistic aspects. We consider each word (i.e., each node) to exist in one of three types of hyperedges, two syntactic and one lexical co-occurrence contexts:

1. **NP**: noun phrase (NP) constituents,
2. **DEP**: dependency relations. We consider all types of dependency functions between nouns and verbs,
3. **SEN**: lexical context, in this case the window considered is the whole sentence

The part of speech information is stored implicitly with the constituent information. While these parameters are fixed in our implementation, they can easily be adapted to other configurations. For example, we may consider noun phrases and verb phrases as chunks, specific types of dependency functions, or different lexical window size.

To populate the hypergraph, given a token v, a noun phrase p, a sentence s, and a dependency function $\mathtt{dep}(h, \cdot)$, with h being the head of the relation, we consider the following rules:

- v is incident (or belongs to a hyperedge e_j of type **NP** if v appears in the same noun phrase p.
- The same condition is used with sentence hyperedges **SEN**: if v appears in a sentence s, it will be located in a hyperedge e_j of type **SEN**.
- If v participates in a dependency function $\mathtt{dep}(h, v)$ as a dependent, it belongs to a hyperedge e_j of type **DEP**.

Each hyperedge is labeled according to an identifier that allows the hypergraph to be populated while reading words from a corpus. For example, the hyperedges of the set **SEN** $= \{h_{S_1}, h_{S_2}, h_{S_3}\}$ are indeed hyperedges that represent sentences, identified by an index in this case. Hyperedges $h_{S_1}, h_{S_2}, h_{S_3}$ contain each a set of words. Additionally, the hypergraph can be represented as a $n \times m$ incidence H matrix with entries $h(i,j) = N(v_i, e_j)$ where $N(v_i, e_j)$ is the number of times $v_i \in e_j$ occurs in the corpus. This frequency values can be later converted into other weighting schemes as seen in Chapter 2. Indeed, the incidence matrix allows us to pass from the hypergraph-based model of representation into the vector-space model.

Running Example We illustrate the process of creating a sample hypergraph model with the following example phrase: *The report contains copies of the minutes of these meetings*. We tokenize the phrase, keeping all the words, and we lemmatize and parse it to obtain both constituency and dependency trees.

Constituency Tree The constituency tree of the example phrase is shown in Figure 3.3. The sentence, as well as each noun phrase (*NP*) node is identified by a number, these numbers serve as an unique identifier of the phrase chunk within the whole sentence. We can observe that this phrase is composed by five noun phrases and one verb phrase. Meanwhile, some *NP* are formed by other kind of phrases, depending on the grammar production rule used to build each one of them. Furthermore, as is usual in this kind of structures, there is a one to one relation between the number of tokens in the sentence and the number of leaves in the tree.

For clarity, in our example we only consider nouns and the first three noun phrases (from left to right), as well as the nominal subject (*nsubj*) and direct object (*dobj*) dependency relations. Thus, in total, as described below, we have three hyperedges of noun phrase type: NP_1, NP_2 and NP_3. Each of them corresponding to the noun phrases in the constituents tree.

Dependency Tree The dependencies of the example phrase are shown in Figure 3.4 as a tree structure. The relations can also be seen as tuples in Table 3.2 In these relations' examples, the head is the first token to appear followed by the dependent word. Two hyperedges are found: $nsubj_{contains}$ and $dobj_{contains}$.

Hyperdges Found From both syntactic parses and the phrase itself we build a hypergraph representation as stated before. We show below the hyperedges sets found for each type, (**NP**, **SEN**, and **DEP**), and their members. Each hyperedge is labeled with a unique identifier:

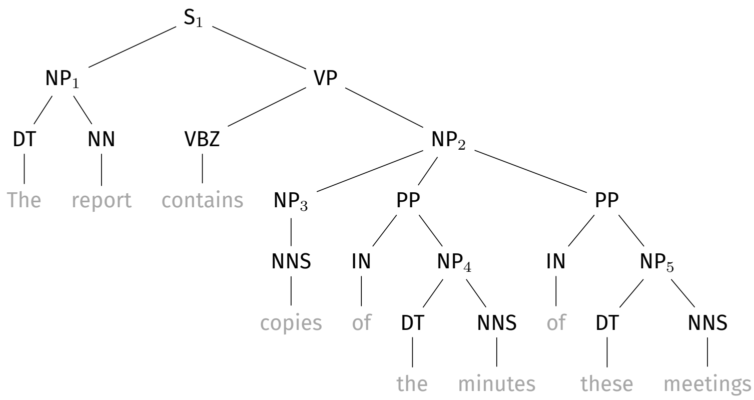


Figure 3.3: Constituency-based tree of the phrase *The report contains copies of the minutes of these meetings.*

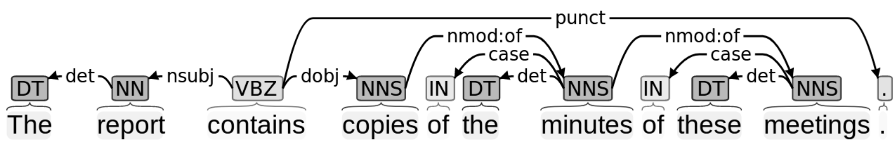


Figure 3.4: Dependency-based tree of the example phrase.

root(root, contains)	**det**(minutes, the)
det(report, The)	**nmod:of**(copies, minutes)
dobj(contains, copies)	**det**(meetings, these)
nmod:of(minutes, meetings)	**nmod**(minutes, meetings)
nsubj(contains, report)	

Table 3.2: Dependency relations of the example phrase.

		NOUN PHRASE			DEPENDENCY		SENTENCE
		NP_1 DT:NN	NP_2 NP:PP:PP	NP_3 NNS	nsubj contains	dobj contains	S_1
NN	report	1			1		1
	copies		1	1		1	1
	minutes		1				1
	meetings		1				1
VB	contains						1

Figure 3.5: Incidence matrix of the example phrase hypergraph modelization.

- **NP** $= \{NP_1, NP_2, NP_3\}$
  - $NP_1 = \{report\}$
  - $NP_2 = \{copies, minutes, meetings\}$
  - $NP_3 = \{minutes\}$
- **SEN** $= \{S_1\}$
  - $S_1 = \{report, contains, copies, minutes, meetings\}$
- **DEP** $= \{nsubj_{contains}, dobj_{contains}\}$
  - $nsubj_{contains} = \{report\}$
  - $dobj_{contains} = \{copies\}$

Incidence Matrix We can represent these hyperedges as an incidence matrix, illustrated in Figure 3.5. Looking at the table, we can infer that the word *copies* appears in two hyperedges of type **NP**: first in NP_2, which is built from a noun phrase and two prepositional phrases (*PP*). Secondly, we see that it is part of NP_3, which indicates a plural noun (*NNS*). Regarding the syntactic dependency hyperedges, the word *copies* appear in the *dobj contains* column which indicates that *copies* was the direct object of the verb *contains*. Concerning the sentence hyperedges, we see that the token *copies* appeared in the same sentence S_1 as the other four noun words.

Discussion With the short example we show the intuitive way in which we identify three different kinds of relations: lexical co-occurrence (at sentence level), dependency co-occurrence, and noun phrase co-occurrence. Looking at the incidence matrix we see the three levels of semantic relatedness we aim to represent with the three different types of context: sentence, dependency, and noun phrase level. At the same time, there is a structure within the hypergraphs. Groups of words are found to be together either directly or by means of paths traced by other nodes.

On the other hand, also looking a the matrix, we realize that it is sparse[3]. Sparsity, as we saw previously, affects the performance of knowledge discovery techniques applied to NLP tasks.

Just as the literature approaches covered before, we aim to solve semantic tasks by using the proposed linguistic resource and its relations. Yet, unlike those approaches we have three contexts and thus three levels of semantic relatedness, coupled to the n-ary relations from the hypergraph structure. Nonetheless, our model also suffers from data sparsity. We will show how to deal with this issue in the following section. The general idea is that by combining features from the different contexts we can alleviate the problem as similarities not seen in a context may complement the features from another context. The set of approaches that perform this combination are known as multimedia fusion techniques.

3.3.2 Representation Enrichment with Fusion Techniques

Multimodal fusion is a set of popular techniques used in multimedia analysis tasks. These methods integrate multiple media features, the affinities among these attributes or the decisions obtained from systems trained with said features, to obtain rich insights about the data being used and thus to solve a given analysis task [Atrey 2010]. We note that these techniques come at the price of augmenting the complexity of a given system by increasing or reducing the sparsity of a given feature matrix.

In the multimodal fusion literature we can discern two main common types of techniques: early fusion and late fusion.

Early Fusion This technique is the most widely used fusion method. The principle is simple: we take both modal features and concatenate them into a single representation matrix. More formally, we consider two matrices that represent different modality features each over the same set of individuals. To perform early fusion we concatenate them column-wise, such that we form a new matrix having the same number of

[3]Sparse for the size of this example incidence matrix. Sparsity increases as more text is included in the hypergraph.

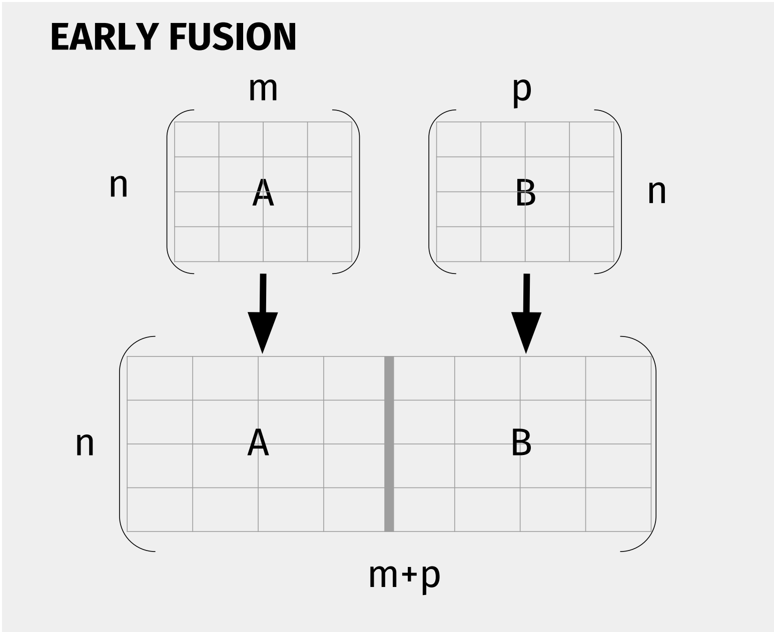


Figure 3.6: Early Fusion of feature matrices A and B. The result is the concatenation of both matrices.

lines but increasing the number of columns to the sum of the number of columns of both matrices. The matrices may also be weighted as to control the influence of each modality.

Such trivial fusion can is showed in Figure 3.6. Two matrices are used, A and B. Each one represent a different feature space. They both have the same number of rows n, they have m and p columns, respectively. After an early fusion operation, the final matrix has $m + p$ features. We will use these matrix notations to exemplify the rest of fusions.

The main advantage of early fusion is that a single unique model is fitted while leveraging the correlations among the concatenated features. The method is also easy to integrate into an analysis system. The main drawback is that we increase the representation space and may make it harder to fit models over it.

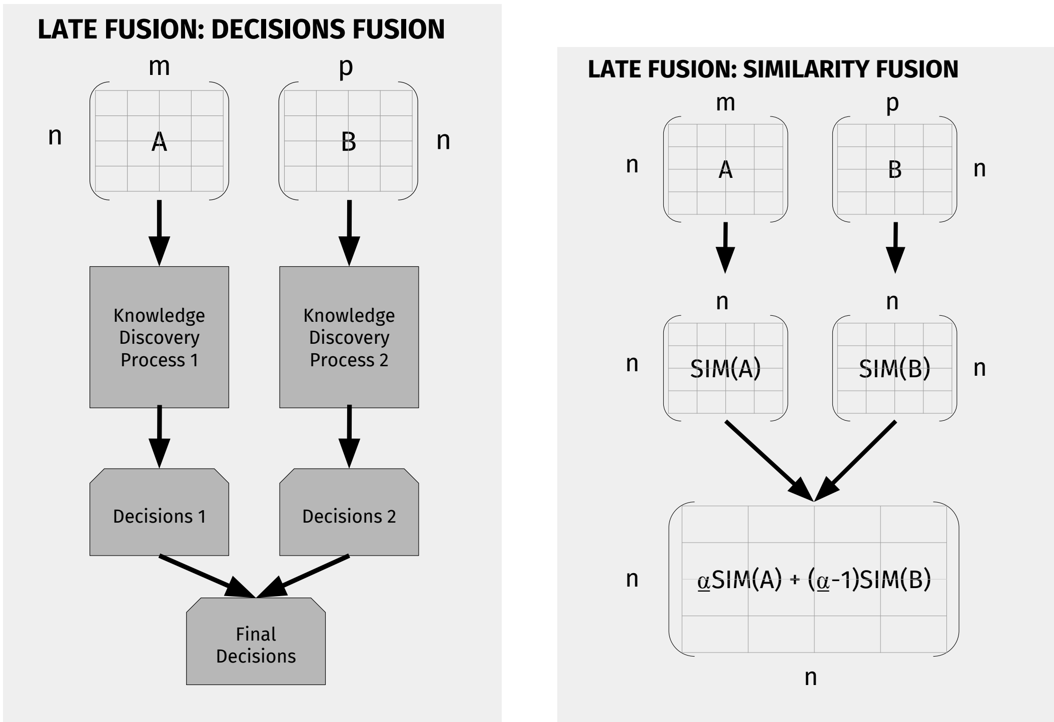
Late Fusion In contrast to early fusion, in late fusion the combination of multimodal features are generally performed at the decision level, i.e., using the output of independent knowledge discovery models trained each with a unique set of features [Clinchant 2011]. In this setting, decisions produced by each model are combined into a single final result set. The diagram in Figure 3.7a shows this combination of matrices A and B. The methods used to combine preliminary decisions usually involve one of two types: rule-based (where modalities are combined according to domain-specific knowledge) or linear fusion (e.g., weighting and then adding or multiplying both matrices together). This type of fusion is very close to the so-called ensemble methods in the machine learning literature. Late fusion combines both modalities in the same semantic space. In that sense, we may also combine modalities via an affinity representation instead of final decision sets. In other words, we can combine two modality matrices by means of their respective similarities. A final representation is then usually obtained by adding the weighted similarity matrices, as in Figure 3.7b, where two such matrices are computed, $\mathrm{SIM}(A)$, and $\mathrm{SIM}(B)$ and then combined into a single weighted matrix (determined by the parameter α).

The advantages of late fusion include the combination of features at the same level of representation (either the fusion of decisions or similarity matrices). Also, given that independent models are trained separately, we can choose which algorithm is more adequate for each type of features.

Cross-media Similarity Fusion A third type of fusion technique, cross-media similarity fusion (or simply cross fusion), introduced in [Ah-Pine 2015, Clinchant 2011], is defined and employed to propagate a single similarity matrix into a second similarity matrix. In their paper, the authors propagated information from textual media towards visual media. In our case, we transfer information among textual features. For example, to perform a cross fusion between lexical and syntactical features, we perform the following steps:

1. Compute the corresponding similarity matrices for each type of features.
2. Select only the k-nearest neighbors for each word within the lexical similarity matrix. These neighbors are to be used as lexical representatives to enrich the syntactical similarities.
3. Linearly combine both similarity matrices (lexical k-nearest lexical neighbors with the syntactical features) via a matrix product.

Figure 3.8 presents a cross fusion between matrices A and B. The similarity matrices $\mathrm{SIM}(A)$ and $\mathrm{SIM}(B)$ are obtained. The k-nearest neighbor operation is applied to matrix A, yielding a matrix $\mathrm{SIM}(A)'$ containing only the most representative similarities. Finally, cross fusion is represented by the product $\mathrm{SIM}(A)' \times \mathrm{SIM}(B)$.



(a) Fusion of decisions.

(b) Fusion of similarity matrices.

Figure 3.7: Late fusion possibilities. To the left, fusion of knowledge discovery models' decisions. To the right, fusion of similarity matrices.

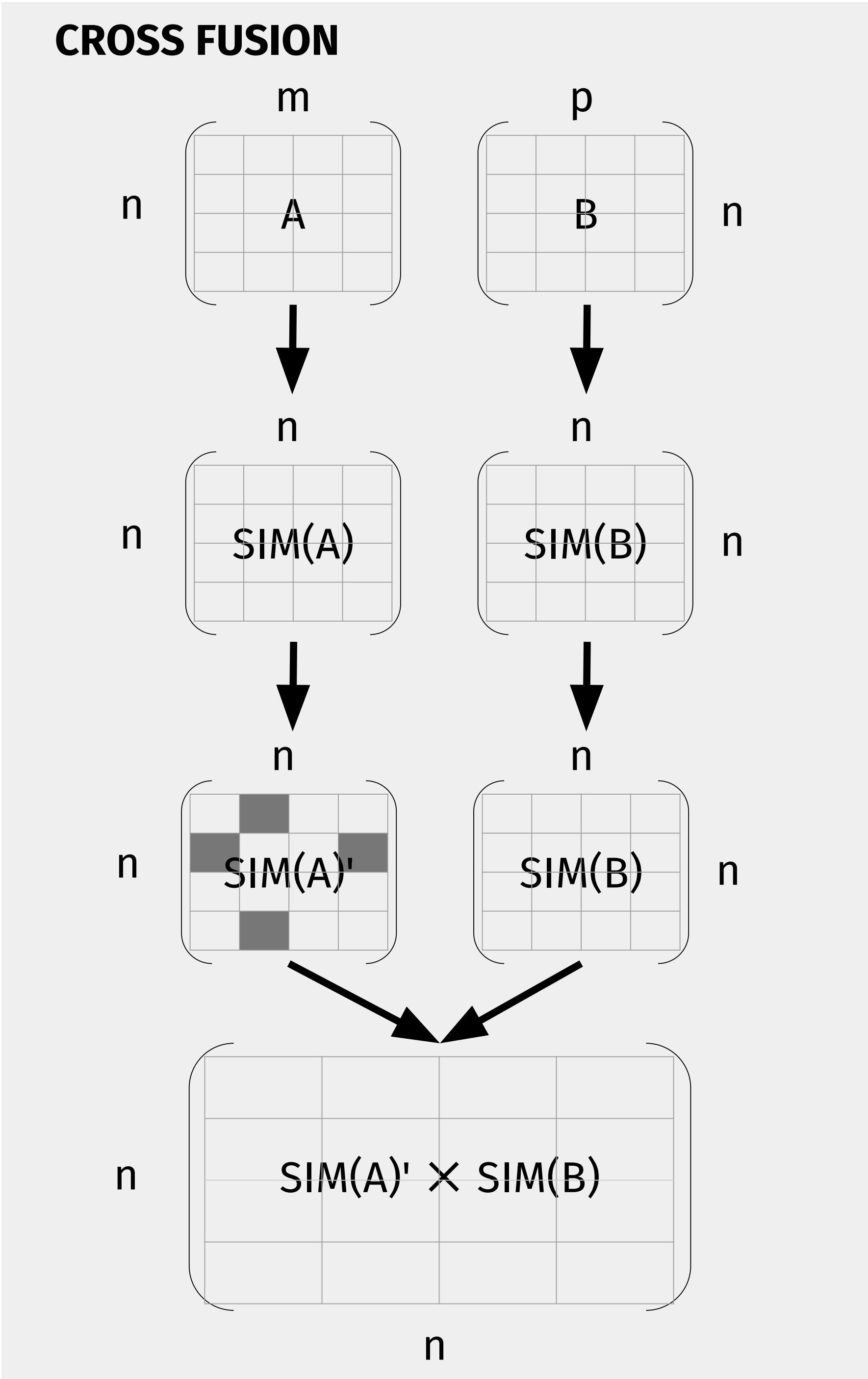


Figure 3.8: Cross fusion for matrices A and B. Similarities are computed, the best similarities are selected for A and finally a product is calculated between the similarities of B and the top similarities of A.

Cross fusion aims to bridge the semantic gap between two modalities by using the most similar neighbors as proxies to transfer valuable information from one modality onto another one. Usually, the result of a cross fusion is combined with the previous techniques, early and late fusion. In this work we perform experiment in that sense.

Hybrid Fusion We may leverage the advantages of the previous three types of fusion techniques by combining them once more in a hybrid setting. As described in [Atrey 2010, Yu 2014], the main idea is to simultaneously combine features at the feature level, i.e., early fusion, and at the same semantic space or decision level. Nonetheless, they define a specific type of hybrid fusion. In this chapter, we adopt a looser definition of hybrid fusion. That is, we perform a hybrid combination of features by leveraging the aggregation of the fusion strategies described before.

Discussion The fusion operators presented (early, late, and cross fusion) are simple and straight-forward. In total, they have two parameters to control: α in late fusion, controlling the importance for each matrix A and B; and k, controlling how many top similarities are kept in cross fusion.

As we will see in the experiments carried out in the next chapter, it is the aggregation of several of these operators that yields interesting results against the use of single features or independent fusion operators. This is in line with other relevant research [Ah-Pine 2015]. We consider early fusion, the simple concatenation, a baseline to the rest of fusion aggregations we perform.

Fusion techniques also have downsides. The operators densify the feature-space matrix but at the same time the number of dimensions grow considerably (with the early fusion operation). Additionally, to the increment of density, the number of features represent an important challenge computationally. We address these concerns on the following chapter.

Before getting into the experimentation details, in order to make our hypergraph linguistic resource concrete, we present the process to obtain such representation from a raw corpus, namely the English Wikipedia. The method we describe first extracts text from the corpus and then analyses it to create a Syntactically Annotated English Wikipedia Dump (SAEWD). From there we detail the steps we carried out to store it as the proposed language network (represented as a hypergraph incidence matrix with its complementary metadata information about the meaning of each vertex and hyperedge).

3.4 Proof of Concept: Wikipedia-based Corpus as Enriched Hypergraph Model

In order to materialize our proposed linguistic model, we need to first create a chain of applications that will extract text from a semi-structured body of text, tokenize it, parse it to extract the syntactic trees the model requires, and then store in order to be used by an NLP application. In this section we describe this process, implemented as an application that takes a corpus as input and outputs the linguistic resource we introduced in the previous section. In this practical example, we use the English Wikipedia corpus as the source for our corpus.

3.4.1 Related Parsed Dumps

The online encyclopedia Wikipedia[4] has been used as a source of valuable data as well as a common background corpus to perform experiments and compare results for diverse NLP/TM related tasks. For example, concerning the first case, in the area of Information Extraction, Wikipedia's infoboxes structured information is used in [Wu 2010] as a valuable resource to complement and improve their open IE system. Along the same line, [Charton 2010] extracted metadata from Wikipedia while leveraging its internal structure in order to produce a semantically annotated corpus. Moving on to the Information Retrieval field, features extracted from Wikipedia can also help to better predict the performance of a query [Katz 2014] in a given corpus. In the second case, as a background collection for experiments, a document-aligned version of English and Italian Wikipedia has been used to determine the quality between word's translations [Vulić 2011].

Wikipedia, being such a popular resource already has various off-the-shelf parsed snapshots (or dumps). These parsed dumps allow researchers to focus more into their approaches than into the extraction and transformation of Wikipedia's data. We briefly describe certain relevant parses found in the literature.

A relevant Wikipedia parsed dump example comes from [Jordi Atserias 2008]. Their work provides a balanced amount of syntactic and semantic information. In short, the dump includes each word's part of speech tag, their dependency relations as well as the output of three different named entity recognition parsers. Additionally, they provide a graph structure that leverages Wikipedia's internal composition along-

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

side its corresponding metadata. Nonetheless, the resource is no longer available on the original URL although it may be obtained through Yahoo's Webscope[5] datasets library. In [Flickinger 2010], they perform a deep parse analysis is performed to provide detailed syntactic and semantic information. The authors leverage a previously manually annotated portion of the English Wikipedia. They extract a grammar from this portion and also train a statistical model to automatically parse the rest of Wikipedia. Even though the parse offered is deep and rich in details, the annotation labels, as well as the corpus output format, may not be convenient and easy to use because of its complexity and particular nature. [Schenkel 2007] released a purely semantic XML parse that links WordNet concepts to Wikipedia pages. They focus greatly on cleaning and pre-treating Wikipedia. In this paper we do not focus as much into the cleaning of Wikipedia as already available tools can solve the task quite well for non-specific needs. Finally, there are certain Wikipedia dumps that offer the raw cleaned text without any extra subsequent parsing or analysis. Such is the case of the corpus made available by [Shaoul 2010]. This corpus makes use of the *WikiExtractor* script [Giuseppe Attardi 2015] to clean the Wikipedia dump.

Although the existing parses and dumps already satisfy numerous specific research needs, they have certain limitations that drove us to build our own resource: the Syntactically Annotated English Wikipedia Dump (SAEWD). Specifically, we address the following shortcomings: the lack of constituents-based tree information, the complex output formats, the limited online access and the absence of the tools used (i.e., the source code) to create the annotated corpus. In SAEWD we include the complete parse tree information for each word provided by well-known parsing tools. We store the extracted information in a simple and already existing output format. Additionally, we give open access to the parsed dump and we share our source code with the community. The code allows anyone (with programming skills) to apply our processing pipeline and build their own particular Wikipedia parse or even to parse other text collections. Finally, we present and provide a hypergraph linguistic network for fast NLP/TM experimentation. Indeed, SAEWD aims to be used as a stepping stone for a standard Wikipedia parsed version for the largest possible set of tasks in future research.

SAEWD uses widely known English language parsing tools, namely those included in the Stanford CoreNLP suite. Aside from being accessible and regularly

[5] https://webscope.sandbox.yahoo.com/

maintained, it provides a common set of labels (Universal Dependencies[6]) used by numerous NLP and TM experiments. Regarding SAEWD output's format, we believe that the file format we use, which follows that of [Jordi Atserias 2008], allows for fast reading and simple interpretation. Among other syntactical information, we provide the constituents parse branch for each word (explained in detail in Section 3.4.3). Constituent's paths, and hence chunk's production rules, have been proved useful as a complement feature to classic text representations [Sagae 9 10, Bergsma 2012, Massung 2013].

Furthermore, we propose a hypergraph linguistic representation. Over the past few years, research on the NLP domain has been focusing on novel techniques that take advantage of the characteristics of language networks to achieve new and interesting results [Mihalcea 2011]. That is why, in addition to SAEWD, we also propose, as a proof of concept, a hypergraph representation that stores certain information found in a SAEWD in a practical way that allows for fast and effortless data extraction. This hypergraph can be indeed considered as a Linguistic Network [Choudhury 2009b]. It aims to facilitate the implementation of graph-based approaches by allowing researchers to jump directly into the algorithm development stage. We use a sub-sample of the Wikipedia corpus consisting of articles related to Natural Language Processing and Text Mining.

In the following sections we describe the steps we undertook to transform a Wikipedia dump into SAEWD (Section 2), we give a detailed account of the contents of SAEWD and the format in which we stored the parsed information (Section 3), then we explain the characteristics of our proposed network structure (Section 4). Lastly, we present our final comments on the nature of the work done as well as possible future work perspectives.

3.4.2 Construction of SAEWD

The three main steps we followed to build SAEWD are presented in Figure 3.9. Briefly, we have one input, which is the Wikipedia dump and one output which is the parsed snapshot. In the following we provide a detailed description of each of the process.

We begin the construction of the parsed corpus with the Wikipedia dump XML file obtained from the Wikipedia database[7] from early November 2014. This dump

[6] http://universaldependencies.github.io/docs/

[7] https://dumps.wikimedia.org/enwiki

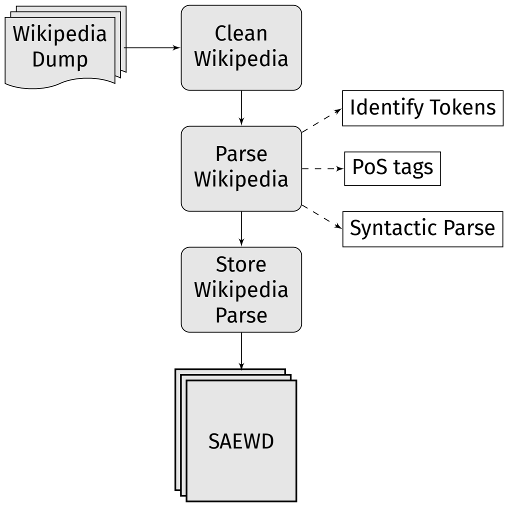


Figure 3.9: The tree steps we took to build SAEWD.

contains around 4.7 million article pages[8]. As shown in Figure 3.9, we apply the following processing steps in order to yield the final parsed version.

3.4.2.1 Cleaning Wikipedia

First, we discard Wikipedia's tables, references and lists, markup annotations and HTML tags with the *WikiExtractor* [Giuseppe Attardi 2015] script. We used this tool to clean and split the content of the original XML file into 429 folders each one containing 100 files of approximately 300 kB. These files contain a certain number of complete Wikipedia articles which is automatically determined by WikiExtractor according to the maximum possible size assigned for each file, 300 kB in our case, thus the number of articles in each file may vary. We decided to use numerous files as well as a small size to easily read their content into memory while parsing. Having multiple small files also makes it easier to handle the multi-threading aspect of our parsing tool. We kept WikiExtractor's original folder nomenclature which assigns to each one of them a sequence of letters sorted lexicographically[9]. The files containing the cleaned text is simply named *wiki_XX* where *XX* goes from 00 to 99, as we have 100 files per folder. It is important to note that the Wikipedia articles' titles themselves are not sorted in any specific way, as it was not in the interest of our research to have them ordered. Inside each cleaned file, besides the article's text, WikiExtractor keeps the original article's

[8] We kept all articles available in the Wikipedia dump.
[9] We have folders named *AA*, *AB*, *AC* and so on.

URL as well as its unique Wikipedia ID within an XML-like label that also doubles as article separator.

3.4.2.2 Parsing Wikipedia

Next, once the Wikipedia dump had been cleaned, we use the Stanford CoreNLP[10] [Manning 2014] analysis tools to parse all the file texts produced during the previous step. As a part of our processing pipeline, we first perform sentence segmentation, word tokenization and lemmatization. Below, we briefly describe each of the extracted attributes. We also exemplify them in detail in Section 3.4.3.

- PoS tagging: We obtain the grammatical category of each word, i.e., the part-of-speech tag, using the CoreNLP default tagger, the *left3words* PoS tagging model.

- Dependency parse: this attribute consists on an extracted tree that describes the types of grammatical relations between words, i.e., the dependency-based parse tree. The analysis was performed with the Stanford's *Shift-Reduce* parser. As information representation, we use the basic dependencies scheme, as we wanted to include each one of the possible dependency relations without any collapsing between them.

- Constituents parse: the output of this analysis is a rooted tree that represents the syntactic structure of a phrase. This tree is commonly known as the constituency-based parse tree. For each word, we store its complete path in the constituency tree. Specifically, we keep all the nodes of a word's own branch from the root to the word itself. We employ the Stanford Shift-Reduce parser. This path is transformed into a single line and included in SAEWD.

Finally, once the parsing process is complete, the parsed files are stored into individual files and thus there are as much parsed files as input Wikipedia cleaned files. The parsed files keep their original name plus the `parsed` extension, e.g., `wiki_00.parsed`. The structure within the files is described in Section 3.4.3.2. After parsing, we found the statistics shown in Table 3.3.

3.4.3 SAEWD Description

In this section we describe in detail the characteristics of SAEWD.

[10] http://nlp.stanford.edu/software/corenlp.shtml

Table 3.3: English Wikipedia dump statistics.

Number of tokens	1,889,769,908
Unique tokens (types)	8,761,691
Number of sentences	84,760,512
Average number of tokens per sentence	22.30

Table 3.4: Extract of a Wikipedia parsed file. The phrase shown is the parse result of the previous example sentence in Figure 3.10

FILENAME wiki_00.parsed

token	lemma	POS	constituency	head	dependency
%%#PAGE Anarchism					
⋮	⋮	⋮	⋮	⋮	⋮
%%#SEN 25 9					
A	a	DT	NP_22,S_97	3	det
great	great	JJ	NP_22,S_97	3	amod
brigand	brigand	NN	NP_22,S_97	4	nsubj
becomes	become	VBZ	VP_44,S_97	0	root
a	a	DT	NP_18,NP_20,VP_44,S_97	6	det
ruler	ruler	NN	NP_18,NP_20,VP_44,S_97	4	xcomp
of	of	IN	PP_57,NP_20,VP_44,S_97	9	case
a	a	DT	NP_18,PP_57,NP_20,VP_44,S_97	9	det
Nation	nation	NN	NP_18,PP_57,NP_20,VP_44,S_97	6	nmod

3.4.3.1 Constituency parse storage in detail

We will use an example to better explain the storage of the constituency-based parse tree. In Figure 3.10 we can see the constituency parse of the phrase *A great brigand becomes a ruler of a Nation*. On the bottom of the figure, we observe the constituent's path (or branch), of the words *brigand* and *Nation*. As in any tree structure, each leaf node has a defined path from the root node to itself. In this example, the leaf containing the noun *brigand* follows the bottom-up path *NP22→S97*. *Brigand's* parent node is a Noun Phrase (NP) node which in turn comes from the root of the tree, the Sentence node *S*. We assign to each phrase chunk an identifier (22 and 97 in this case) in order to distinguish them according to their building elements as specified by the

PoS Tag	Token	NP				DEP				SEN
		NP_22$_1$	NP_20$_1$	NP_18$_1$	NP_18$_2$	nsubj_become	xcomp_become	nmod_ruler	amod_brigand	S$_1$
NN	brigand	1				1				1
	ruler		1	1			1			1
	nation		1		1			1		1
VB	becomes									1
JJ	great	1							1	1

Table 3.5: Brief example of the linguistic network incidence matrix of the previous used phrase. On the left side, as on the top, we can see the metadata we store for each word (rows) and each column (hyperedges). We omit the rest of the words from the example phrase for brevity.

grammar rule used. In other words, a phrase chunk, e.g., a NP, a Verbal Phrase (VP), a Prepositional Phrase (PP), or other chunk defined by the grammar in CoreNLP, may be built from different types of PoS tags. Thus, again from Figure 3.10, we see that the sentence *S97* is built both from a NP and a VP chunk. In a similar way, the noun phrase *NP18* is produced by a determinant (DT) and a noun (NN), while *NP22* is generated by a determinant, an adjective (JJ) and a noun. The identification digits are obtained from the hash code that represents each chunk object inside our Java application. For each phrase-chunk tree node, we keep the last two significative figures produced by the `hashCode`[11] Java method.

As another example, the noun *Nation* has the following bottom-up constituency path: *NP18*→*PP57*→*NP20*→*VP44*. Indeed, the string `NP_18,PP_57,NP_20,VP_44,S_97`, originating from the previously described path, is the information we keep about the constituency parse for each token in the Wikipedia dump.

3.4.3.2 Annotation scheme

To store the parsed text we use a scheme inspired by that used in [Jordi Atserias 2008]. The format can be considered as a regular tsv file (i.e., the entries are separated by tab spaces) with additional metadata tags. An extract from a parsed file can be observed in Table 3.4.

The file includes two headers: the first one simply indicates the name of the current parse file; the second one contains the names that describe each column. The tags and columns our parsed dump contains are the following:

- Metadata tags:

[11] Java `hashCode` function description: https://en.wikipedia.org/wiki/Java_hashCode%28%29

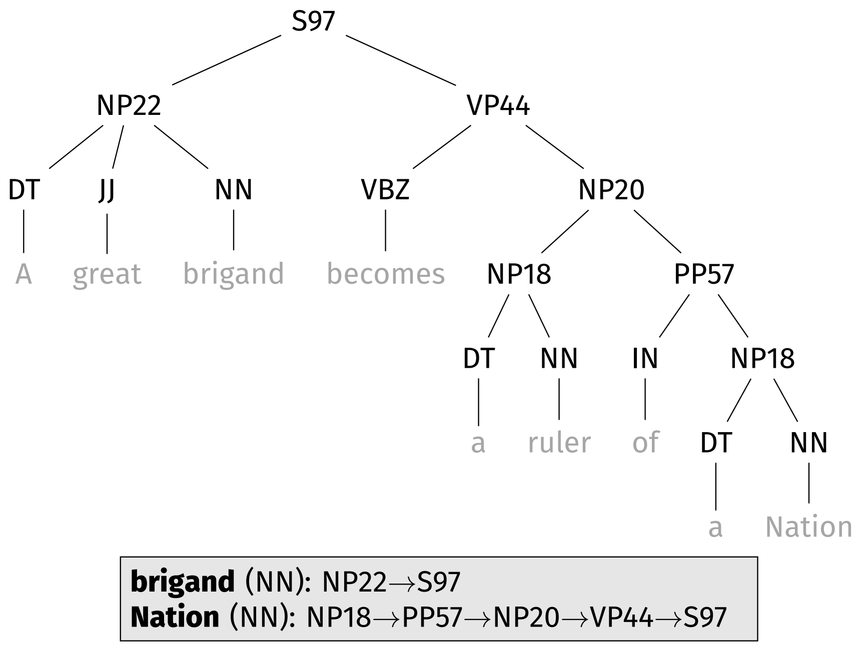


Figure 3.10: Constituency tree for the phrase *A great brigand becomes a ruler of a Nation*. On the bottom, we can see the bottom-up path stored for the words *brigand* and *Nation*.

1. FILENAME: indicates the original file used to extract the current parse,
2. %%#PAGE: denotes a new Wikipedia article, as well as its title,
3. %%#SEN: marks the beginning of a new sentence. It is followed by two integers: (1) the number of the current sentence, and (2), the number of tokens in the sentence.

- Parse columns for each token:

  1. Token: the token itself,
  2. Lemma: the token the canonical form,
  3. POS: its part of speech tag,
  4. Constituency: the bottom-up constituency path described before,
  5. Head: the head index of the dependency relation the current token belongs to,
  6. Constituency: the name of the grammatical relationship this token participates in as a dependent.

Using the example phrase introduced before (Table 3.4), the token *becomes* has *become* as lemma, it is a verb, thus it has *VBZ* as PoS tag, its constituency path is

VP_44,S_97, so it belongs to the verb phrase VP44 which in turn comes from sentence S97. Finally, *becomes*, being the main verb, is in this case the grammatical root of the sentence and its head is by convention determined as zero.

3.4.3.3 Discussion

Concerning the computation time, SAEWD takes around 40 hours to be produced using a general purpose laptop (Intel i7 4700MQ with 4 cores, 8 GB and Linux Mint 17 as operative system). Most of the time is taken by the parsing step.

We verified the consistency of the corpus built by analyzing a sample of 20 Wikipedia articles. The output of CoreNLP and the information contained in the corpus match.

3.5 Conclusion

In this chapter we analyzed the state of the art of linguistic network-based approaches to semantic similarity task from a graph-centric point of view. We reviewed the techniques in terms of its graph characteristics, from their structure to the algorithms employed. Among the literature covered, certain non-explored research paths were identified, namely the lack of syntactic data on the networks employed, and therefore, a homogeneous network nature that only allows for relations of a unique type.

We addressed these paths with the proposition of a hypergraph linguistic model that is able to hold heterogeneous language information. We believe that this structure allows the integration multiple kinds of information and has vast potential in terms of which algorithms it can be used with. The three levels of context we integrated in the model (sentence lexical co-occurrence, dependency function co-occurrence, and constituent-membership co-occurrence) aim to cover distinct levels of semantic relatedness. We noted the challenges of dealing both with textual data sparsity and leveraging the heterogeneity of the hypergraph. To alleviate both concerns, we propose the use of fusion functions, introduced also in this chapter. The structure of the hypergraph is also an important characteristic that we can use to find groups of semantically related words within a corpus. Finally, we presented a materialization of a corpus, a portion of the English Wikipedia, as the linguistic network we proposed.

In the following section we set to solve two natural language processing challenges using the model described above as a linguistic resource. Specifically, we address the tasks of word sense induction and disambiguation, and the named entity recognition. Both tasks are located on the computational semantics sub-domain of NLP. By making use of the network we want to test the effectiveness of using different types of linguistic features.

Chapter 4

Linguistic Model Applications: NLP Semantic Tasks

Abstract. *This chapter contains the experiments we performed as applications of the proposed model. We address the use of the heterogeneous contexts, the structure of the network and the sparsity of textual data. First, we propose a method to solve word sense induction and disambiguation. This task is an elementary semantic NLP task useful to build larger systems. In the first section of this chapter, we leverage the network presented previously and propose an algorithm that takes into account the structure of our model. Based on the real-world property of the network, we induce senses by grouping words that share a similar sense. We then assign these senses to target words. We show that our method improves on similar network-based methods. Furthermore, we analyze how each type of context behave in terms of performance for each target word tested.*

Secondly, in the second section of the chapter, we explore the use of well-known multi-modal fusion techniques to solve two prominent Natural Language Processing tasks. Specifically, we focus on solving Named Entity Recognition and Word Sense Induction and Disambiguation by applying feature-combination methods that have already shown their efficiency in the multi-media analysis domain. We present a series of experiments employing fusion techniques in order to combine textual linguistic features. Our results show that the combination of textual features indeed improves the performance compared to single feature and the trivial feature concatenation.

Contents

4.1 Leveraging the Network Structure

4.1.1 Introduction

In this chapter we employ the model introduced before to propose a solution to both Word Sense Disambiguation (WSD) and Induction (WSI) tasks. The intuition of our method is not original, as we will see, but still we address concerns that are not very well studied, namely the use of heterogeneous context features to solve semantic tasks while reducing the number of parameters compared to similar approaches. Indeed, we also analyze the difference provided by different types of contexts. Our method is evaluated with two important datasets, that of Semeval 2007 and 2010.

We begin by giving a description and and recalling the art for both tasks. Specifically, we identify what type of language network is used and how. We then present the characteristics of our proposed method. Finally we describe and discuss the obtained results.

Word sense disambiguation, or WSD, is the task of examining a word in different contexts (we refer to a word in a particular context as an instance) and determining which is the sense being used in each one of the contexts analyzed. Usually a list of senses from where to chose the correct one is given as an input. When this list is not available, WSD then becomes a different, complementary task: word sense induction, or WSI. WSI also analyses tokens of a target word context but before assigning a sense to each of its instances, it first generates a list of possible senses from where to select from.

Word Sense Disambiguation Recall that the general process used to solve WSD follows a two-step algorithm. Given an input text with a words to disambiguate, and their corresponding context, WSD systems accomplish the following:

1. Link target words (usually nouns, without stop-words and functional words) with their corresponding sense (or synset in the case of Wordnet-like dictionaries) and extract their vertices and edges into a new, smaller SN.

2. Apply a node ranking technique, usually a random walk based method, and select, for each ambiguous word in the input text, its top ranking synset node as the correct sense.

The differences in the methods, in general boil down to three aspects. Firstly, the type of semantic relation implied by the edges of the network. Secondly, the function used to assign a weight to each edge, as well as its meaning. On the other hand, a common characteristic of these approaches is that, as said before, they all rely on semantic networks and thus are constrained by the limitations of such resources, mainly their static content and human annotated nature. We find that the use of a lexical knowledge base, such as Wordnet, is pervasive in this task. Indeed, new resources, such as BabelNet, solves to some extent the fixed (no new senses are included automatically) nature of this type of resources by leveraging the ever-evolving knowledge of Wikipedia.

Word Sense Induction We believe that in order to solve WSD in a truly end-to-end unsupervised way, one would need to automatically find a list of senses for a word without the help of pre-built semantic networks. Unsupervised WSI is performed as follows:

Given an input document with a set of target words, coupled with a set of contexts (a target word in a unique context is called an instance), the goal is to discover a list of senses for each target word and then assign each instance in the document with an automatically generated sense. The common four steps used are the following:

1. Build a LCN, assigning tokens as nodes and establishing edges between them if they co-occur in a given context (usually if they both appear in the same sentence, paragraph or fixed window of words).

2. Determine the weights for each edge according to a frequency metric.

3. Apply a graph clustering algorithm. Each cluster found will represent a sense of the polysemous target word.

4. Match target word instances with the clusters found (the senses) by using the word context. Specifically, assign a sense to each instance by looking at the tokens in the context. This step is actually the word sense disambiguation task.

As with WSD, while LCNs used are mostly the same among approaches, there are certain moving parts that make up the difference between WSI approaches. The most common differences that arise are:

- The clustering algorithm to find senses in the LCN graph.
- The technique used to match context words to clusters.
- The weight used in the graph edges.

WSI, while being a more flexible approach (language and word-domain independent, does not require human-made knowledge bases), its results are tightly linked to the quality of the clustering algorithm used on the language network built.

With respect to the networks' structure, we find that few approaches include syntactic attributes into their model. We believe that finding semantic similarities can be improved by adding syntactic information not only while using dependency relations but also by leveraging the constituency tree of each word. Moreover, using syntactic data along with semantic and/or lexical co-occurrences takes us into the heterogeneous network domain which has not been addressed in most of the approaches covered. Being able to design new similarity metrics that deal with different types of information opens new avenues of research in the semantic similarity domain.

4.1.2 Proposed Method

Formally, the objective of WSI/WSD is the following: given a document d with a set T of target words $tw \in T$ and the set C with contexts for each target word ct_{tw}. Specifically, each paragraph represents the context of a target word. A target word in a specific context is also called an instance. The goal is first to solve the WSI task, that is, automatically determine a list of senses for a given tw, and then assign one meaning from this list to each of its instances, the WSD task.

Our method follows is inspired on previous approaches from [Véronis 2004] and [Klapaftis 2007]. In Hyperlex, the graph-based method presented in [Véronis 2004], the main intuition is that co-occurrence networks have small-world properties and thus it is possible to detect and isolate important heavily-connected nodes, call "hubs". These hubs, and their connected nodes represent a sense in itself.

Hyperlex performs WSI and WSD using a weighted lexical co-occurrence network. The process is performed for each target word in the document. As a first step, they build a graph by defining the vertices (the target word node is removed) as the tokens found in the co-occurring context of a target word. The edges link two words co-occurring together. Each edge is assigned a weight that decreases as the association frequency of the words increases. The second step consists on iteratively finding the hubs and removing them, along with their adjacent nodes, from the target word graph. Again, the intuition of the method is that these isolated hubs, and their adjacent words, represent a sense of the analyzed word. The third and final step carries out the disambiguation. A new graph is created by adding the target word to the co-occurrence graph. Zero-weighted edges are added between each hub and the target word. A minimum spanning tree is then calculated and the sense component found to have the closest set of nodes is chosen as the target word sense.

The second approach, *UOY*, described in [Klapaftis 2007], relies itself on the small-world intuition presented by Hyperlex to find hubs and its adjacent nodes to represent senses. In short, these methods, as ours, exploit the real-world characteristics of linguistic networks by theorizing that there are certain few high-degree nodes (called hubs) that carry an important role in the network and therefore may represent, coupled with their neighbors, a sense for a given target word. Particularly, UOY considers bigrams and trigrams that co-occur in a paragraph as hyperedges. Under a frequent-itemset setting, they determine important hyperedges given their *support* and their *confidence* values. Then, the clustering of words takes place by finding the hubs and considering them as sense carriers only if they satisfy a threshold mainly set upon their containing-hyperedge confidence value. Finally, once the senses are identified, each target word instance (represented by a context) is assigned to a sense according to the sum of confidences of the hyperedge appearing on said context.

In our method, we generate a network for each tw and consider that the high-degree nodes inside this network may represent a tw sense. Figure 4.1 shows an overview of the process. Also, in Algorithm 1 we show the general flow of our approach. We detail the steps taken alongside the corresponding line in the algorithm below.

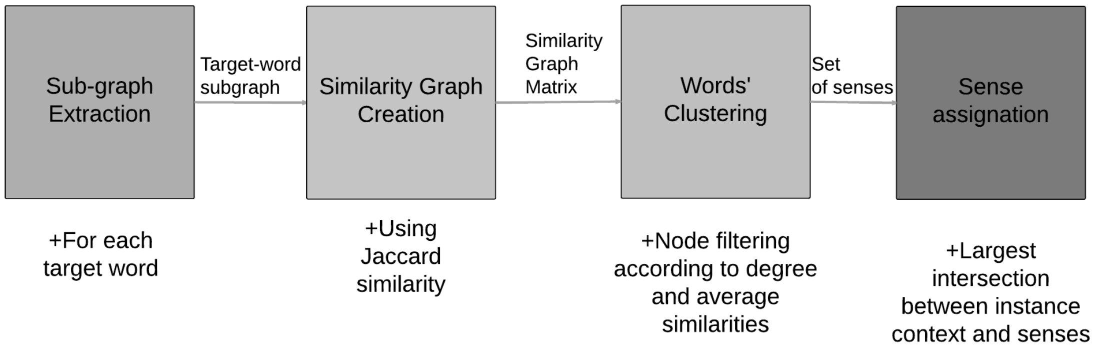


Figure 4.1: Block diagram of the WSI/WSD method proposed.

Creation of the linguistic network In order to find senses from the contexts of a target word, the first step in our procedure is to build a linguistic graph G_H from a background corpus. As described in previous sections, the dependency and constituency trees are used to build the hypergraph: words are depicted by nodes, and they may exist inside any of the three different types of hyperedges defined (sentence, noun phrase or dependency contexts). If any hyperedge is repeated through the corpus, we increment a counter and keep the number of apparitions instead of adding redundant columns to the hypergraph incidence matrix.

At each step, that is, for each tw in the test input document, we extract a subgraph G_{tw} from G_H that contains all the words that appear together with tw (line 2), whether by lexical or syntactic co-occurrence. The tw is removed from G_{tw}. In this approach we focus specifically on dependency relations and lexical co-occurrence.

We note that for the syntactic co-occurrence, that is, the dependency relations between words, we apply two strategies: when dealing with a noun target word, we use the co-occurrent relations between said noun and other words having a similar head dependency token. On the other hand, when dealing with verbs, we select the co-occurrent words having said verb as head of the dependency relation.

Computing the similarity between nodes In order to computationally treat G_{tw}, we first induce a bipartite graph $B_{tw} = (U, W, E)$ from G_{tw} (line 3). The set of left nodes U represent words and the set of right nodes W depicts the membership to a given hyperedge. Thus, we have as many nodes in W as we had hyperedges in G_H.

We compute the Jaccard index between each node $n_{i,j} \in U$ as $\mathrm{Jaccard}(i,j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$ in order to build a $|U| \times |U|$ similarity matrix S_{tw} (line 4). We induce from

S_{tw} a new filtered hypergraph incidence matrix F_{tw} (line 5), which contains word nodes as rows and columns as hyperedges. Each of these hyperedges represent a set of words that are deemed similar between them according to their Jaccard index value, which must be equal or higher than an assigned threshold th_1 .

Clustering words together Once the incidence matrix F_{tw} is built we can proceed to induce senses for a target word by clustering words (vertices) together. First, we calculate the degree of each node $n_i \in F_{tw}$. The degree of a node is simply the number of hyperedges it is incident in. Nodes are sorted in descending order and evaluated one by one. We consider the top c-nodes as sense hub candidates (line 6). We accept or reject a node $n \in F_{tw}$ as a sense carrying word according to one condition. As shown from line 11 to 17 in the pseudo-code, we set a minimum limit to the average of the Jaccard similarities between each pair of neighbors of node $n \in F_{tw}$ within each hyperedge n belongs to. Formally, for a node n, we define the average Jaccard measure as:

$$AvgJaccard(n) = \frac{1}{|hedges(n)|} \sum_{h \in hedges(n)} \frac{\sum_{\substack{i \in h \\ j \in h; i \neq j}} Jaccard(i, j)}{|h| + 1}$$

where $hedeges(n)$ is the set of hyperedges n is incident in and its cardinality is defined as $|hedges(n)|$. $|h|$ is the number of nodes in hyperedge h.

The Jaccard similarity measure allows us to easily determine the neighbors of each node in the current bipartite hypergraph representation. As each node is joined to a sentence or dependency node, calculating the Jaccard similarity amounts to determining the level of co-occurrence between each word according to a specific type of hyperedge (represented as a node from the other graph partition) while taking into account the total number of hyperedges the words participate in. We differentiate specifically from the previously described method, UOY, in that in the case of that system, the weighting of the hyperedges is done by computing the average confidence metric of each hyperedge. In this regard, the Jaccard similarity is more flexible with respect to the confidence metric, as the confidence requires in the numerator the number of contexts (paragraphs in UOY's case) shared by all the members of the hyperedge, whereas the Jaccard measure takes pairs of members individually and thus is less strict in the apparition of all the elements of the hyperedge in the contexts. Given the nature of the features used (lexical and syntactical dependencies), we fix our thresholds in a manual but simpler way by defining percentiles and taking the value of the threshold directly, without having to change it according to the characteristics of the data.

If node n satisfies both thresholds th_1 and th_2, it is deemed as a sense purveyor and all its neighbors (words that appear in the same hyperedges as n) are conflated into a single set representing a tw sense. This new sense is added to SoS_{tw} (line 17). The sense set is then removed from F_{tw}.

The process is repeated until no more nodes satisfy both boundaries. When the process is complete, we obtain a set of senses SoS_{tw} where each set contains words that ideally represent a unique meaning for each target word.

Sense assignation The assignation of a sense consists in looking at each tw instance represented by a context ct and simply determining which sense s in SoS_{tw} shares the highest amount of words with ct. The sense s is thus assigned to that instance. If two senses in SoS_{tw} share the same amount of words with ct, one of them is randomly chosen. This operation is repeated for each instance of each target word.

4.1.3 Experiments and Evaluation

4.1.3.1 Datasets

We trained and evaluated our system on two datasets: Semeval-2007 (task 2) [Agirre 2007a] and Semeval-2010 (Task 14) [Manandhar o 07]. The Semeval-2007 task consisted in the induction and disambiguation of a single set of 100 words, 65 verbs and 35 nouns, each target word having a set of contexts where the word appear. On the other hand, the Semeval-2010 task consisted also on 100 words, with 50 being verbs and 50 being nouns. This time, a training set from which the senses of a word have to be induced is provided. In our experiment, for the Semeval-2010 dataset, we induce the senses from the training set and disambiguate the target words within the test set.

We apply a light pre-treatment, consisting on token lemmatization and we remove all words that appear less than four times. Concerning the individual graphs of each target word, we work only with nouns and if the extracted graph has fewer than 100 nodes, we do not apply any filtering (we keep all the extracted words). We do this in order to avoid empty solutions.

Algorithm 1: Pseudo-code of our WSI/WSD network-based approach

```
Input: A set tw_set = {tw_1, tw_2, ..., tw_n} of target words
Input: A background linguistic network G_H
Input: Filtering thresholds th_1, th_2
Output: A set SoS_tw of senses for each target word
foreach target word tw in tw_set do
    G_tw = extract_subgraph(G_H, tw);
    B_tw = induce_bipartite_graph(G_tw);
    S_tw = sim_matrix(B_tw);
    F_tw = induce_hypergraph(S_tw, th1);
    candidate_hubs = sort(degree(F_tw))[:100];
    SoS_tw = [ ];
    foreach candidate_hub in candidate_hubs do
        candidate_hyperedges = get_hyperedges(candidate_hub, F_tw);
        candidate_avg_jaccard = 0;
        foreach hyperedge in candidate_hyperedges do
            candidate_avg_jaccard += get_avg_jaccard(hyperedge);
        end
        if candidate_jaccard > th2 then
            SoS_tw.add(get_words(candidate_hyperedges));
            F_tw = F_tw \ candidate_hyperedges;
    end
    return SoS_tw
end
```

4.1.3.2 Implementation

The objective of this experiment is to show the complementarity of both lexical and syntactic co-occurrence information while solving WSI and WSD tasks while using the method described in the previous subsection. To that end we build two independent systems: **LEX**, which uses exclusively lexical co-occurrence hyperedges, and **DEP**, which employs only syntactic dependency hyperedges. Each type of hyperedge has its own network characteristics as mentioned before. Sentence hyperedges tend to have a much smaller number of words than those of the dependency category. This make sense as sentences usually contain less than 30 words, meanwhile a dependency hyperedge may contain up to hundreds of words (several words may share the same dependency relation). Taking this into consideration we set different threshold values for **LEX** and for **DEP**. First, we consider only the top 100 nodes as candidate sense hubs. Secondly, we do not set the thresholds' values directly but instead we set up a percentile value for the Jaccard similarity ($\mathtt{th}_1$) and for the average Jaccard similarity ($\mathtt{th}_2$). This is a practical solution to the changing nature of the network model according to the features being employed. We experimentally found the best values for each threshold used.

4.1.4 Results and Discussion

Both Semeval-2007 and Semeval-2010 tasks are evaluated by an unsupervised and supervised set of measures.

Semeval-2007 In the case of Semeval-2007, the unsupervised evaluation assumed the induced senses as clusters of examples. These clusters are compared to the sets of examples tagged with the given gold standard word senses (classes), and evaluated using the *F-Score* measure for clusters.

The supervised setting maps the induced senses to manually-defined gold standard senses, and use a mapping produced by the organizers to tag the test corpus with gold standard tags. The mapping is automatically produced by the organizers, and the resulting results evaluated according to the usual precision and recall measures for supervised word sense disambiguation systems.

In Table 4.1 we present the unsupervised evaluation results for our models as well as for some other systems. We include the F-Score measure as performance metric. Three baselines are included. The best one, one cluster per word, or *1c1word* was not beaten during the competition. The second was a random assignation of senses to each instance. Finally, the third and easiest to beat baseline assigned one cluster per instance 1c1instance.

In this table, as in the rest of the tables presented in this section, the columns show the results for all the words, for the nouns exclusively as well as for the verbs. The final column indicates the number of induced clusters per system. It is important to consider this value as the unsupervised metrics are biased towards systems with less number of induced clusters and thus to the 1c1word baseline.

We can appreciate that both our methods surpass the baselines and the system described before *UoY(2007)*. The best system of the competition, *UBC-AS* used also co-occurrence graphs and applied a random-walk based clustering algorithm over the vertices' similarity matrix. Still, our system induced a larger amount of senses, while retaining a competitive F-Score value. We also note that in this evaluation **DEP**, the system using only co-occurrent dependency relations outperformed the lexical co-occurrence only system **LEX**.

Moving onto the supervised results for Semeval-2007, in Table 4.2 we show the results obtained concerning the Recall performance metric. In this table we include the competing system *I2R*, based on an Information Bottleneck based clustering algorithm, which obtains the best results according to all the words and nouns. Both our systems **DEP** and **LEX** beat the baseline of assigning the most frequent sense to an instance (*MFS*). More interestingly, *DEP* was able to beat the MFS verb baseline, something that was not achieved during the competition. As was the case before, our systems beat UoY(2007).

As a way of determining where does both systems complement each other, in Figure 4.2 and Figures 4.3 and 4.4 we show the unsupervised F-Score value for nouns and verbs respectively (we split the verbs in two figures for visibility). We can see that, as the previous result tables indicated, **DEP** did better overall. Nonetheless, and what is most interesting in these figures, is that there are certain words, nouns and verbs, that obtain better scores using **LEX** instead of **DEP** and vice versa. For example, the nouns *area*, *future*, and *state* are better treated by **SEN**, according to this measure, even if by a small margin. On the other hand, with respect to the verbs, the differences between performance are more important. The system **SEN** does better while finding senses and assigning them to the verbs *avoid*, *fix*, and *work*. This information will be useful during the design of hybridization techniques between feature of our hypergraph structure.

FS (%)	**all**	**nouns**	**verbs**	**#cl**
1c1word	78.9	80.7	76.8	1.00
UBC-AS	78.7	80.8	76.3	1.32
DEP	74.9	80.2	69.0	3.27
LEX	61.4	62.6	60.1	4.26
UoY(2007)	56.1	65.8	45.1	9.28
Random	37.9	38.1	37.7	19.7
1c1instance	9.5	6.6	12.7	48.51

Table 4.1: Unsupervised F-Score (FS) for the Semeval 2007 test set

SR (%)	**all**	**nouns**	**verbs**	**#cl**
I2R	81.6	86.8	75.7	3.08
LEX	79.4	82.5	75.9	4.26
DEP	79.1	81.5	76.4	3.27
MFS	78.7	80.9	76.2	1
UoY(2007)	77.7	81.6	73.3	9.28

Table 4.2: Supervised Recall (SR) on the Semeval 2007 test set

Semeval-2010 In Semeval-2010, two unsupervised evaluation metrics are introduced: *V-Measure* and *paired F-Score*. Briefly, the V-Measure assesses the quality of a clustering solution by measuring the degree to which each cluster consists of instances principally belonging to a single gold standard class, or homogeneity; and to completeness, the level with which each gold standard class consists of instances primarily assigned to a single cluster.

The paired F-Score evaluation transforms the clustering problem into a classification one. Two instance pairs sets are generated, the first one coming from the system induced clusters, by including pairs of the instances found in each cluster. The second set of instance pairs is built from the gold standard classes. It contains pairs of the instances found in each class. We can then define *Precision* and *Recall*. Precision is computed as the ratio of the number of common instance pairs between the two sets

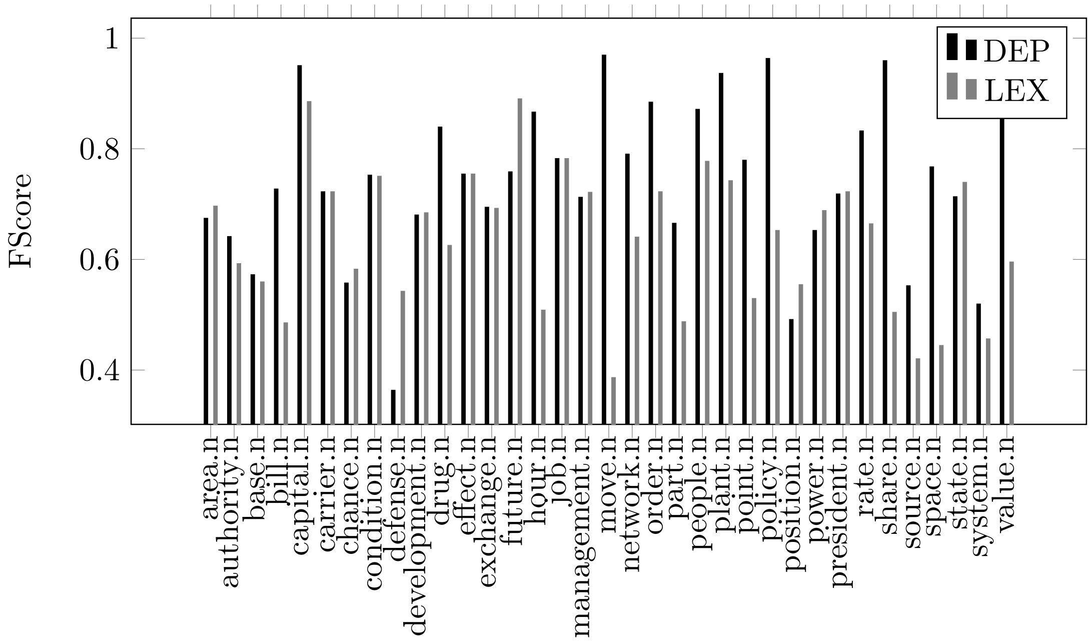


Figure 4.2: Unsupervised F-Score results for the nouns of the Semeval 2007 test set.

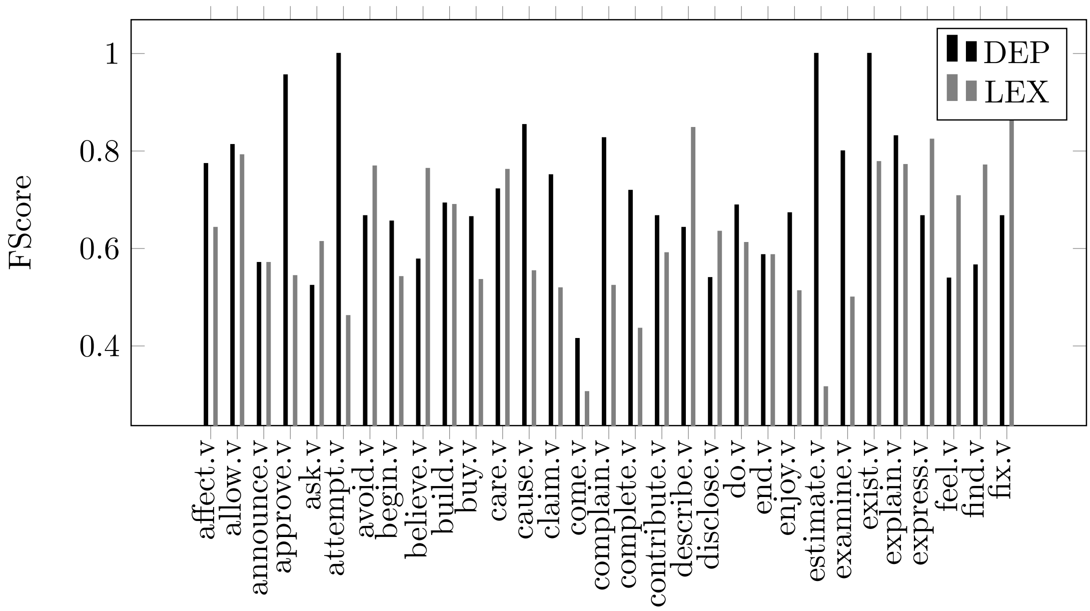


Figure 4.3: Unsupervised F-Score results for the first half of verbs of the Semeval 2007 test set.

divided by the total number of pairs in the clustering solution. Recall is the count of common instance pairs between the two sets divided by the total number of pairs in

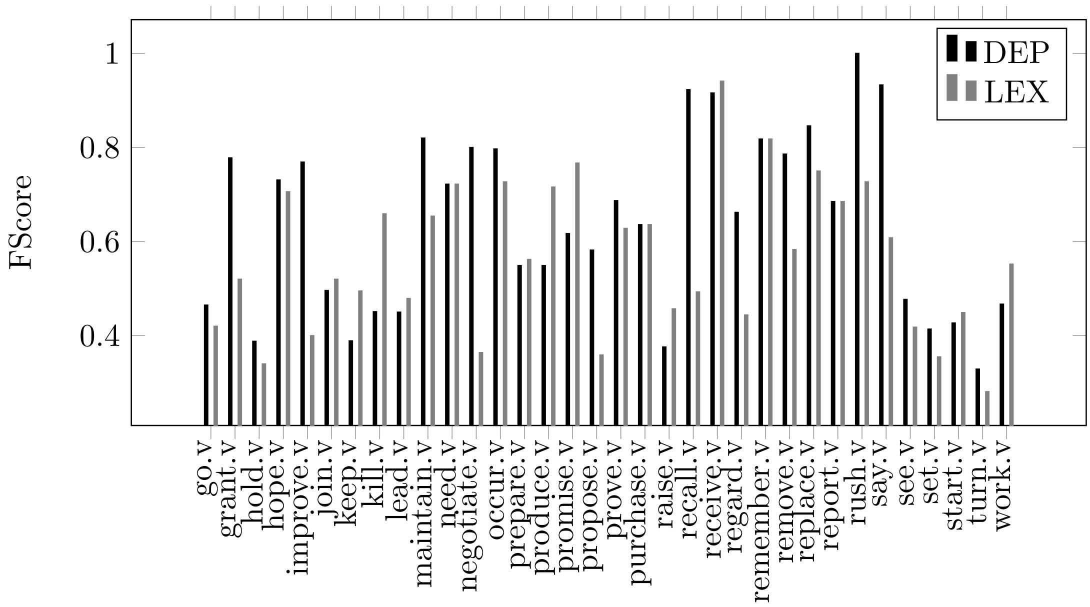


Figure 4.4: Unsupervised F-Score results for the second half of verbs of the Semeval 2007 test set.

the gold standard. The paired F-Score is the harmonic mean of both quantities.

Concerning the supervised evaluation, it follows explicitly that of Semeval-2007, using recall as the main performance measure as in [Manandhar o 07, Van de Cruys 2011, Pedersen 2010].

In Table 4.3 we present our systems compared to the baseline and other methods using during Semeval 2010. V-Measure is a metric well-known for favoring systems producing a higher number of clusters [Van de Cruys 2011, Pedersen 2010]. Thus, it is considered not a very reliable metric. We have included it to have a global insight about the performance of our method. We remark that only **LEX** performed better than both baselines, random assignation of senses (Random) and using the most frequent sense (MFS). Also, we note that we included only the best performing systems, in this case $NMF_{\texttt{lib}}$ and *Hermit*. The former did not participate on the competition but was developed later. We include it henceforth to illustrate how systems variate from one position to another depending the metric used to assess the performance. The latter was the best method on this metric from the task challenge. It is important to notice that other systems exist between Hermit and **LEX**. They were not included for the sake of clarity.

The second unsupervised measure, Paired F-Score, can be seen in Table 4.4. In this

VM (%)	all	nouns	verbs	#cl
Hermit	16.2	16.7	15.6	10.78
$\text{NMF}_{\texttt{lib}}$	11.8	13.5	9.4	4.80
LEX	11.6	8.8	11.9	10.5
Random	4.4	4.2	4.6	4.00
DEP	3.5	3.9	2.8	2.75
MFS	0.0	0.0	0.0	1.00

Table 4.3: Unsupervised V-Measure (VM) on the Semeval 2010 test set

case both systems presented performed better than the random baseline. Any system presented was able to beat the MFS baseline. We note that **DEP** does much better compared to **LEX** concerning verbs, namely 58% vs. 28% F-Score. Still, the results are low considering the best results of the competition, 63% from Duluth, although again, it generates a number of senses very similar to the MFS baseline. Both our systems induce a considerable amount of clusters while keeping a descent F-Score.

Finally, we in Table 4.5 we show the supervised Recall results of Semeval-2010. The best performing algorithm shown is $\text{NMF}_{\texttt{lib}}$. During the competition, UoY(2010) was the best method. It is a graph-based algorithm which shares the name with the UoY system presented in Semeval-2007, but it is a different approach.

Concerning our systems, in this evaluation they seem to perform the best, or in a comparable level, to the top methods. We find that in general our systems seem to perform better on the Semeval-2007 dataset. Discovering the reason could shed light into improving the performance on the Semeval-2010 test set. Given the results, it seems like a combination of features (syntactic plus lexical) in a single algorithm could yield better results.

As a side note, it is argued that supervised Recall is not a very robust metric as the supervised method within tends to converge towards the most frequent sense. In this regard, several researchers [Van de Cruys 2011, Pedersen 2010] have voiced their concerns about the quality of the current WSI/WSD evaluation metrics as well as the need of new, more robust techniques to properly evaluate these systems.

FS (%)	all	nouns	verbs	#cl
MFS	63.5	57.0	72.4	1.00
Duluth-WSI-SVD-Gap	63.3	57.0	72.4	1.02
DEP	53.6	50.1	58.7	2.75
$\text{NMF}_{\texttt{lib}}$	45.3	42.2	49.8	5.42
LEX	38.4	46.7	28.5	10.5
Random	31.9	30.4	34.1	4.00

Table 4.4: Unsupervised Paired F-Score (FS) for the Semeval 2010 test set

SR (%)	all	nouns	verbs
$\text{NMF}_{\texttt{lib}}$	62.6	57.3	70.2
UoY(2010)	62.4	59.4	66.8
LEX	59.8	55.8	67.4
DEP	59.3	53.9	67.2
MFS	58.7	53.2	66.6
Random	57.3	51.5	65.7

Table 4.5: Supervised recall (SR) for Semeval 2010 test set (80% mapping, 20% evaluation)

4.1.5 Conclusion

We proposed a WSI/WSD method based on the contexts from the hypergraph model presented in the previous chapter. We show how using the inner links within the hypergraph structure we can group words that represent senses and then assign them to target words. Our method distinguishes from similar works in two main aspects: the definition of similarity used, the reduced number of parameters that are needed, the use of diverse types of contexts to solve the task. We show that our method beats said similar approaches. Also, we discovered the behavior of syntactic contexts in comparison to lexical contexts at word-level. Indeed, lexical contexts seem to perform better. This is in line with other works on distributional representations [Kiela 2014].

4.2 Combining Heterogeneous Features

4.2.1 Introduction

Named Entity Recognition (NER) and Word Sense Induction and Disambiguation (WSI/ WSD) requires textual features to represent the similarities between words to discern between different words' meanings. NER goal is to automatically discover, within a text, mentions that belong to a well-defined semantic category. The classic task of NER involves detecting entities of type Location, Organization, Person and Miscellaneous. The task is of great importance for more complex NLP systems, e.g, relation extraction, opinion mining. Two common solutions to NER are one of the following: via matching patterns created manually or extracted semi-automatically; or by training a supervised machine learning algorithm with large quantities of annotated text. The latter being the currently more popular solution to this task.

As stated before, both tasks rely on features extracted from text. Usually, these representations are obtained from the surrounding context of the words in the input corpus. Mainly, two types of representations are used. According to their nature we call these features lexical and syntactical. The first type requires no extra information than that contained already in the analyzed text itself. It consists merely on the tokens surrounding a word, i.e., those tokens that come before and after within a fixed window. The second type, syntactical features, is similar to the lexical representation in that we also consider as features the tokens that appear next to the corpus' words. Nonetheless, it requires a deeper degree of language understanding. In particular, these features are based on part of speech tags, phrase constituents information, and syntactical functionality between words, portrayed by syntactical dependencies. Likewise, specific features, particular to one task are also be employed.

Most of the approaches in the literature dealing with these tasks use each of these features independently or stacked together, i.e., different feature columns in an input representation space matrix. In the latter case, features are combined without regards to their nature.

The main intuition of the present work is that word similarities may be found at different levels according to the type of features employed. In order to exploit these similarities, we look into multimedia fusion methods. In order to better perform an analysis task, these techniques combine multimodal representations, their corresponding similarities, or the decisions coming from models fitted with these features.

In our experiments, we try to mutually complement independent representations by utilizing said fusion techniques to combine (or fuse) features in the hope of improving the performance of the tasks at hand, specially compared to the use of features independently.

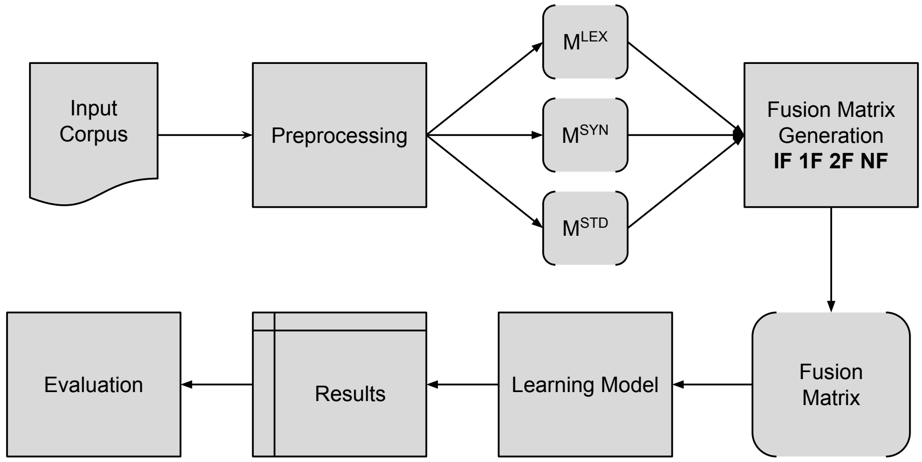
Fusion techniques have previously shown their efficiency, mainly on image-related tasks, where there is a need to model the relation between images and text extracts. Here, in order to apply multimedia fusion techniques, we consider that the textual features come from different modalities. The main contribution of this work is to assess the effectiveness of simple yet untested techniques to combine classical and easy to obtain textual features. As a second contribution, we propose a series of feature combination and recombination to attain better results.

The rest of the chapter is organized as follows: in section **??**, we go into further details about fusion techniques. We introduce the fusion operators that we use in our experiments in section 4.2.2. Then, in section 4.2.3 we show the effectiveness of the presented methods by testing them on NER and WSI/WSD and their respective datasets. Finally, in section 4.2.5 we present our conclusions and future directions to explore. A more general overview on fusion for multimedia analysis can be found in [Atrey 2010].

We consider the first three types of fusion techniques described in the Chapter 3.3.2 (early fusion, late fusion and cross fusion) as the building blocks to the experiments we conduct. While we work with a single modality, i.e., textual data, we consider the different kinds of features extracted from it as distinct modalities. Our intuition being that the semantic similarities among words in these different spaces can be combined in order to exploit the latent complementarity between the lexical and syntactical representations. The fusion should therefore improve the performance of the NLP tasks at hand, NER and WSI/WSD.

Our first goal is to assess the effectiveness of the classic fusion methods and then, as a second goal, to propose new combinations that yield better outcomes in terms of performance than the simpler approaches. The new combinations are found empirically. Nonetheless, as we will show, their effectiveness replicates to different datasets and NLP tasks. As will be clear in the next sections, the ways in which we tackle the tasks are, by themselves, mostly not original. We focus specially on the generation of new representation spaces. We use well-proven methods to solve both WSI/WSD and NER, so that comparisons with other systems may make more sense.

Figure 4.5: Steps followed on our experiments. First the corpus is preprocessed, then features are extracted from the text. A fusion matrix is generated, which in turn is used as input to a learning algorithm. Finally, the system yields its results and to be analyzed.



4.2.2 Proposed Method

Our first goal is to assess the effectiveness of the classic fusion methods and then, as a second goal, to propose new combinations that yield better outcomes in terms of performance than the simpler approaches. The new combinations are found empirically. Nonetheless, as we will show, their effectiveness replicates to different datasets and NLP tasks.

In the present subsection we address the core of the work performed in this chapter. We formally describe the fusion techniques we employ in the next section. Also, we delineate the procedure followed in our experiments.

The experiments we carry on consist in generating fusion matrices that will serve as input to a learning algorithm in order to solve NER and WSI/WSD. These input feature matrices are based upon lexical, syntactical, or other types of representation. The procedure can be sen in Figure 4.5.

4.2.2.1 Fusion Strategies

We begin by presenting a formal definition of the fusion techniques employed and described in the previous sections. We define (weighted) early fusion, late fusion and cross fusion as follows:

Early Fusion

$$E(A, B) = \textbf{hstack}(A, B) \tag{4.1}$$

Weighted Early Fusion

$$wE_{\alpha}(A, B) = \textbf{hstack}(\alpha \cdot A, (1 - \alpha) \cdot B) \tag{4.2}$$

Late Fusion

$$L_{\beta}(A, B) = \beta \cdot A + (1 - \beta) \cdot B \tag{4.3}$$

Cross fusion

$$X_{\gamma}(A, B) = \mathbf{K}(A, \gamma) \times B \tag{4.4}$$

Parameters A and B are arbitrary input matrices. They may initially represent, for example, the lexical (M^{LEX}) or syntactical based (M^{SYN}) features matrix, or their corresponding similarity matrices, S^{LEX} and S^{SYN}, respectively. In a broader sense, matrices A and B may represent any pair of valid[1] fusion matrices.

In early fusion, $E(A, B)$, the matrices A and B are combined together via a concatenation function **hstack** which joins both of them column-wise. Weighted early fusion represents the same operation as before with an extra parameter: α, which controls the relative importance of each matrix. In the following, we refer to both operations as early fusion. When α is determined, we refer to weighted early fusion.

Regarding late fusion $L_{\beta}(A, B)$, the β parameter determines again the importance of the matrix A, and consequently also the relevance of matrix B.

In cross fusion $X_{\gamma}(A, B)$, the $\mathbf{K}(\cdot)$ function keeps the top-γ closest words (columns) to each word (lines) while the rest of the values are set to zero.

Using the previously defined operators, we distinguish four levels of experiments:

1. **Single Features**: in this phase we consider the modalities independently as input to the learning methods. For instance, we may train a model for NER using only the lexical features matrix M^{LEX}.

2. **First Degree Fusion**: we consider the three elementary fusion techniques by themselves (early fusion, late fusion, cross fusion) without any recombination. These experiments, as well as those from the previous level, serve as the baselines we set to surpass in order to show the efficacy of the rest of the fusion

[1]Valid in terms of having compatible shapes while computing a matrix sum or multiplication.

approaches. As an example, we may obtain a representation matrix by performing an early fusion between the lexical matrix and the syntactical features matrix: $E(M^{LEX}, M^{SYN})$. In this level we distinguish two types of cross fusion: Cross Early Fusion (XEF) and Cross Late Fusion (XLF). The first one combines a similarity matrix with a feature matrix: $X(S^{LEX}, M^{SYN})$. The second one joins a similarity matrix with a similarity matrix: $X(S^{SYN}, S^{LEX})$.

3. **Second Degree Fusion**: we recombine the outputs of the previous two levels with the elementary techniques. This procedure then yields a recombination of "second-degree" among fusion methods. We introduce the four types of second degree fusions in the following list. Each one is illustrated with an example:

    (a) Cross Late Early Fusion (XLEF): $X(X(S^{STD}, S^{SYN}), M^{STD})$

    (b) Cross Early Early Fusion (XEEF: $X(S^{STD}, X(S^{STD}, S^{SYN}))$

    (c) Early Cross Early Fusion (EXEF): $E(M^{STD}, X(S^{LEX}, M^{STD}))$

    (d) Late Cross Early Fusion (LXEF): $L(M^{STD}, X(S^{STD}, M^{STD}))$

4. **N-Degree Fusion**: in this last level we follow a similar approach to the previous level by combining the output of the second-degree fusion level multiple times (more than two times, or $N > 2u$) with other second-degree fusion outputs. Again, in this level we test the following two fusion operations:

    (a) Early Late Cross Early Fusion (ELXEF): $E(M^{STD}, L(M^{STD}, X(S^{STD}, M^{STD})))$

    (b) Early ELXEF (EELXEF): $E(M^{LEX}, E(E(M^{STD}, L(M^{STD}, X(S^{STD}, M^{STD}))), L(M^{LEX}, X(S^{SYN}, M^{LEX}))))$

4.2.2.2 Feature Matrices

In the previous subsection we presented the fusion operators used in our experiments. Below we detail the three types of features used to describe the words of the corpus tested.

Lexical Matrix (LEX) For each token in the corpus, we use a lexical window of two words to the left and two words to the right, plus the token itself. Specifically, for a target word w, its lexical context is $(w_{-2}, w_{-2}, w, w_{+1}, w_{+2})$. This type of context features is typical for most systems studying the surroundings of a word, i.e., using a distributional approach [Levy 2014a].

Syntactical Matrix (SYN) Based on the syntactic features used in [Levy 2014a, Panchenko 2017], we derive contexts based on the syntactic relations a word participates in, as well as including the part of speech (PoS) of the arguments of these relations. Formally, for a word w with modifiers $m_1, \dots, m_k$ and their corresponding PoS tags $p_1^m, \dots, p_k^m$; a head h and its corresponding PoS tag p_h, we consider the context features $(m_1, p_{m_1}, lbl_1), \dots, (m_k, p_{m_k}, lbl_k), (h, p_h, lbl\_inv_h)$. In this case, lbl and lbl_{inv} indicate the label of the dependency relation and its inverse, correspondingly. Using syntactic dependencies as features should yield more specific similarities, closer to synonymy, instead of the broader topical similarity found through lexical contexts.

NER Standard Features Matrix (STD) The features used for NER are based roughly on the same as those used in [Daume 2006, Balasuriya 2009]. The feature set consists of: the word itself, whether the word begins with capital letter, prefix and suffix up to three characters (within a window of two words to the left and two words to the right), and the PoS tag of the current word. These features are considered to be standard in the literature. We note that the matrix generated with these features is exclusively used in the experiments regarding NER.

4.2.2.3 Learning Methods

We use supervised and unsupervised learning methods for NER and WSI/WSD respectively. On the one hand, for NER, as supervised algorithm, we use an averaged structured perceptron [Collins 2002, Daume 2006] to determine the tags of the named entities. We considered Logistic Regression and linear SVM. We chose the perceptron because of its performance and the lower training time.

On the other hand, for WSI/WSD, specifically for the induction part, we applied spectral clustering, as in [Goyal 2014], on the input matrices in order to automatically discover senses (a cluster is considered a sense). Regarding disambiguation, we trivially assign senses to the target word instances according to the number of common words in each cluster and the context words of the target word. In other words, for each test instance of a target word, we select the cluster (sense) with the maximum number of shared words with the current instance context.

4.2.3 Experiments and Evaluation

We experiment with four levels of fusion: Single Features (SF), First-degree Fusion (1F), Second-degree Fusion (2F) and N-degree Fusion (NF). The representation matrices for NER come from lexical context features M^{LEX}, syntactical context features M^{SYN} or standard features M^{STD}. On the other hand, experiments on WSI/WSD exclusively employ matrices M^{LEX} and M^{SYN}.

Our first goal is to compare the efficiency of the basic multimedia fusion techniques applied to single-modality multi-feature NLP tasks, namely NER and WSI/WSD. A second goal is to empirically determine a fusion combination setting able to leverage the complementarity of our features.

To this end, we evaluate the aforementioned 4 fusion levels. We note that the fusion combinations in the third and fourth level (2F and NF) are proposed based on the results obtained in the previous levels. In other words, in order to reduce the number of experiments, we restrict our tests to the best performing configurations. This is due to the large number of possible combinations (an argument to a fusion operation may be any valid output of a second fusion operation).

4.2.3.1 Named Entity Recognition

Pre-processing As is usual when preprocessing text before performing named entity recognition, [Ratinov 2009], we normalize tokens that include numbers. For example, the token 1980 becomes *DDDD* and 212-325-4751 becomes *DDD*-*DDD*-*DDDD* . This allows a degree of abstraction to tokens that contain years, phone numbers, etc. We do not normalize punctuation marks.

Features The linguistic information we use are extracted with the Stanford's CoreNLP parser [Manning 2014]. Again, the features used for these experiments on NER are those described before: lexical, syntactic and standard features, i.e., M^{LEX}, M^{SYN}, and M^{STD}, respectively.

Test Datasets We work with three corpus coming from different domains:

(1) CoNLL-2003 (CONLL): This dataset was used in the language-independent named entity recognition CoNLL-2003 shared task [Sang 2003]. It contains selected news-wire articles from the Reuters Corpus. Each article is annotated manually. It is divided in three parts: training (*train*) and two testing sets (*testa*

and *testb*). The training part contains 219,554 lines, while the test sets contain 55,044 and 50,350 lines, respectively. The task was evaluated on the *testb* file, as in the original task.

(2) WikiNER (WNER): A more recent dataset [Balasuriya 2009] of selected English Wikipedia articles, all of them annotated automatically with the author's semi-supervised method. In total, it contains 3,656,439 words.

(3) Wikigold (WGLD): Also a corpus of Wikipedia articles, from the same authors of the previous corpus. Nonetheless, this was annotated manually. This dataset is the smaller, with 41,011 words. We used this corpus to validate human-tagged Wikipedia text. These three datasets are tagged with the same four types of entities: Location, Organization, Person and Miscellaneous.

Evaluation Measures We evaluate our NER models following the standard CoNLL-2003 evaluation script. Given the amount of experiments we carried on, and the size constraints, we report exclusively the total F-measure for the four types of entities (Location, Organization, Person, Miscellaneous). WNER and WGLD datasets are evaluated on a 5-fold cross validation.

Results We present in this subsection the results obtained in the named entity recognition task, while employing the 4 levels of fusion proposed in the previous section.

In contrast to other related fusion works [Ah-Pine 2015, Clinchant 2011, Gialampoukidis 2016], we do not focus our analysis on the impact of the parameters of the fusion operators. Instead, we focus our analysis on the effect of the type of linguistic data being used and how, by transferring information from one feature type to another, they can be experimentally recombined to generate more complete representations.

Regarding the fusion operators' parameters, we empirically found the best configuration for β, from late fusion $L_\beta(A, B) = \beta \cdot A + (1 - \beta) \cdot B$, is $\beta = 0.5$. This implies that an equal combination is the best linear fusion for two different types of features.

In respect of the γ parameter, used in cross fusion $X_\gamma(A, B) = \mathbf{K}(A, \gamma) \times B$, we set $\gamma = 5$. This indicates that just few high quality similarities attain better results than utilizing a larger quantity of lower quality similarities.

Table 4.6: NER F-measure results using the Single Features over the three datasets. These values serve as a first set of baselines.

$\mathcal{A}$	Single Features		
	CONLL	**WNER**	**WGLD**
M^{STD}	77.41	77.50	59.66
M^{LEX}	69.40	69.17	52.34
M^{SYN}	32.95	28.47	25.49

Single Features Looking at Table 4.6, we see that the best independent features, in terms of F-measure come from the standard representation matrix M^{STD}. This is not surprising as these features, simple as they may be, have been used and proved extensively in the NER community. On the other hand, M^{LEX} performs relatively well, considering it only includes information contained in the dataset itself. Nevertheless, this representation that this kind of lexical context features are the foundation of most word embedding techniques used nowadays. While we expected better results from the syntactical features M^{SYN}, as they are able to provide not only general word similarity, but also functional, getting close to synonymy-level [Levy 2014a], we believe that the relatively small size of the datasets do not provide enough information to generalize

First Degree Fusion In Table 4.7 we present the First Degree fusion level. The best performance is obtained by trivially concatenating the representation matrices. This baseline proved to be the toughest result to beat. Late fusion does not perform well in this setting, still, we see further on that by linearly combining weighted representation matrices, we can add information to an already strong representation. Finally, regarding the cross fusion techniques, cross early and late fusion, we see that they depend directly on the information contained in the similarity matrices. We note that, as is the case on single features, the combinations with matrix S^{STD} yield almost always the best results. While these fusion techniques by themselves may not offer the best results, we see below that by recombining them with other types of fusion we can improve the general performance of a representation.

Table 4.7: NER F-measure results using first degree fusion (1F). B is either indicated on the table or specified as follows. Looking at EF, $\hat{b}_{EF} = E(M^{SYN}, M^{STD})$. In XEF, b^{*}_{XEF} takes the matrix from the set $\{M^{LEX}, M^{STD}\}$ which yields the best performing result. In XLF, $\hat{b}^{*}_{XLF}$ corresponds to the best performing matrix in $\{S^{LEX}, S^{SYN}\}$. These configurations serve as the main set of baseline results.

A	B	CONLL	WNER	WGLD
		Early Fusion		
M^{LEX}	M^{SYN}	72.01	70.59	59.38
M^{LEX}	M^{STD}	78.13	79.78	61.96
M^{SYN}	M^{STD}	77.70	78.10	60.93
M^{LEX}	$\hat{b}_{EF}$	**78.90**	**80.04**	**63.20**
		Late Fusion		
S^{LEX}	S^{SYN}	**61.65**	58.79	44.29
S^{LEX}	S^{STD}	55.64	**67.70**	48.00
S^{SYN}	S^{STD}	50.21	58.41	**49.81**
		Cross Early Fusion		
S^{LEX}	M^{STD}	49.90	**70.27**	**62.69**
S^{SYN}	M^{STD}	47.27	51.38	48.53
S^{STD}	b^{*}_{XEF}	**52.89**	62.21	50.15
		Cross Late Fusion		
S^{LEX}	S^{STD}	27.75	**59.12**	38.35
S^{SYN}	b^{*}_{XLF}	36.87	40.92	39.62
S^{STD}	b^{*}_{XLF}	**41.89**	52.03	**39.92**

Second Degree Fusion The second degree fusion techniques presented in Table 4.8 show that the recombination of cross fusion techniques gets us closer to the early fusion baseline. With the exception of cross late early fusion, the rest of the recombination schemes yield interesting results. First, in cross early fusion, the best results, for the most part, are obtained while using the S^{LEX} matrix combined with the output of $E(M^{LEX}, M^{STD})$, which is still far from the baseline values. Concerning, EXEF, we get already close to surpass the baselines with the M^{STD} matrix, except for the CONLL dataset. In LXEF, even though the cross fusion $X(S^{SYN}, M^{LEX})$ is not the best performing, we found experimentally that by combining it with M^{LEX} through a late fusion, it gets a strong complementary representation. Our intuition in this case was to complement M^{LEX} with itself but enriched with the S^{SYN} information. In the N-degree fusion results we discover that indeed this propagation of information helps us beat the baselines we set before.

N-degree Fusion Finally, the last set of experiments are shown in Table 4.9. Using a recombination of fusion techniques, a so-called hybrid approach, we finally beat the baselines (single features and early fusion) for each dataset. We note that the best configuration made use of a weighted early fusion with $\alpha = 0.95$. This indicates that the single feature matrix, M^{LEX} is enriched a small amount by the fusion recombination, which is enough to improve the results of said baselines. In CONLL, the early fusion (see Table 4.7) baseline being 78.13, we reached 78.69, the lowest improvement of the three datasets. Regarding the Wikipedia corpus, in WNER, we went from 79.78 to 81.75; and in WGLD, from 61.96 to 67.29, the largest improvement of all.

In the next section we transfer the knowledge gained in this task to a new one, word sense induction and disambiguation.

4.2.3.2 Word Sense Induction and Disambiguation

Having learned the best fusion configuration from the previous task, in these experiments we set to test if the improvements achieved can be transfered into another NLP task, namely Word Sensed Induction and Disambiguation (WSI/WSD). As preprocessing, we simply remove stopwords and tokens with less than three letters.

Features We use the same set of features from the previous task, without the standard NER features, that is, those represented by M^{STD}, as they are specifically designed to tackle NER.

Test Dataset The WSI/WSD model is tested on the dataset of the SEMEVAL-2007 WSID task [Agirre 2007b]. The task was based on a set of 100 target words (65 verbs and 35 nouns), each word having a set of instances, which are specific contexts where the word appear. Senses are induced from these contexts and applied to each one of the instances.

Evaluation Measures Being an unsupervised task, the evaluation metrics of WSI/WSD are debated in terms of quality [de Cruys 2011]. We consider supervised recall and unsupervised F-measure, as in the competition original paper [Agirre 2007b]. The first one maps the output of a system to the true senses of the target words' instances and the second one measures the quality of the correspondence between the automatically found clusters and the senses. We consider that the number of senses found by the system is also a rather good indicator of performance: the best competition baseline assigns the most frequent sense to each target word (this baseline is called MFS), thus this baseline system would have an average of 1 sense (cluster) per word. A system that goes near this average may be indeed not resolving the task efficiently but finding the MFS trivial solution. Consequently, to show that we do not fall in the MFS solution, we display in our results the average number of clusters.

4.2.4 Results and Discussion

Word sense induction and disambiguation results are found in Table 4.10. Again, we aim to surpass the baseline of the single features and early fusion. We experimentally set $\beta = 0.90$ and $\gamma = 50$. In this task, in late fusion, when the first matrix is deemed more relevant than the second one, the performance is higher. This may be due to the fact that, in this task, the feature matrices rows contain types (that is, each line represent an unique word), and thus they are more dense, which may entail more noisy data. By reducing the relevance of the second matrix in late fusion, we are effectively attenuating the less important information. Regarding $\gamma = 50$, again due to the denser characteristic of the matrices, there is a larger quantity of true similar words that are useful to project information into another matrix, through cross fusion.

The WSI/WSD results are shown in Table 4.10. In the following paragraph, we will discuss these result all at once. Due to the page limit constraint, we omit certain configurations that do not yield interesting results either by converging to the MFS solution (1 sense found per target word) or because the performance shown by those configurations is not interesting.

Regarding Single Features, M^{LEX} comes on top of M^{SYN} again. Nonetheless, M^{SYN} is much closer in terms of performance, and as expected, it is actually higher with regards to verbs.

On the 1F level, we see that the early fusion technique in this task does not surpass the independent features representation. Our intuition is that the similarities of both matrices seem to be correlated. In cross early fusion, the best result is obtained by $X(S^{LEX}, M^{LEX})$, regarding the unsupervised F-measure. This configuration already beats our baselines, improving both noun and verb results on the unsupervised evaluation, improving the supervised recall of nouns, and staying on the same level considering all words. Also, it produces more senses than the MSF average number of senses (1 sense per target word), which is good but not indicative of results correctness. Regarding cross late fusion, given the average number of clusters produced, it seems that both results converge towards the MFS, therefore we do not consider these results.

Beginning with the fusion recombinations, in level 2F, both cross late early fusions yield average results. In cross early cross early fusion, the early fusion of M^{LEX} with $X(S^{LEX}, M^{LEX})$ yields very similar results than $X(S^{LEX}, M^{LEX})$. The next natural step is to test this fusion via a linear combination, with a late fusion. The result obtained confirmed the intuition of enriching a single feature matrix with another weighted-down matrix to improve the performance. Indeed, we consider that $L(M^{LEX}, X(S^{LEX}, M^{LEX}))$ gets the best results in terms of all-words supervised recall and the second best all-words unsupervised F-measure (we do not consider solutions that are too close to the MFS baseline).

We test the same configurations as in NER, within the NF level, to try and improve our results. Nonetheless, in general, they do not overcome the best result found previously. In general, we found that the recombination fusion techniques work in terms of improving the performance of the tasks addressed.

In the following, we discuss how each type of feature affects the performance of individual classes in NER or the senses discovered in WSI/WSD.

4.2.5 Conclusion

We presented a comparative study of multimedia fusion techniques applied to two NLP tasks: Named Entity Recognition and Word Sense Induction and Disambiguation. We also proposed new fusion recombinations in order to complement the information contained in the single representation matrices. In order to accomplish this goal, we built upon basic fusion techniques such as early and late fusion, as well as cross media fusion to transfer quality information from one set of features to another.

We found that by taking a strong feature, in our case lexical context, M^{LEX}, and enriching it with the output of rather complex fusion combinations, we can improve the performance of the tasks addressed. The enrichment has to give more relevance to the strong feature matrix, by selecting the right parameters.

While there is an improvement, we do note that fusion techniques augment the complexity of the tasks at hand by enlarging the feature space or making it more dense.

In that sense, more intelligent ways of finding the most appropriate fusion must be researched. This is indeed one of our future work paths: determining an optimal fusion path from single features to a N-degree fusion recombination. Coupled with this, the automatic determination of the parameters is still ongoing research in the multimedia fusion community. Consequently, we believe that efficiently determining both parameters and fusion combinations is the general domain of our future work. Another route we would like to explore is testing these techniques on other tasks and with datasets from different domains, in order to assert its effectiveness.

Table 4.8: NER F-measure results using second degree fusion (2F). In XLEF, a^* corresponds to the best performing matrix in the set $\{X(S^{STD}, S^{LEX}), X(S^{LEX}, S^{STD}), X(S^{STD}, S^{SYN})\}$. For XEEF, $\hat{b}_{XEEF} = E(M^{LEX}, M^{STD})$. In EXEF, b^*_{EXEF} takes the best performing matrix from $\{X(S^{SYN}, M^{LEX}), X(S^{LEX}, M^{LEX}), X(S^{LEX}, M^{STD}), X(S^{SYN}, M^{LEX}), X(S^{SYN}, M^{STD})\}$. Finally, in LXEF, $\hat{b}_{LXEF}$ takes the best possible matrix from $\{X(S^{LEX}, M^{STD}), X(S^{SYN}, M^{STD}), X(S^{SYN}, M^{LEX})\}$.

A	B	**Cross Late Early Fusion**		
		CONLL	**WNER**	**WGLD**
$\hat{a}$	M^{STD}	37.69	59.44	**41.71**
$\hat{a}$	M^{LEX}	**38.31**	**58.73**	41.56
$\hat{a}$	M^{SYN}	29.31	52.06	34.91
		Cross Early Early Fusion		
		CONLL	**WNER**	**WGLD**
S^{STD}	$\hat{b}_{XEEF}$	**54.34**	**64.20**	39.59
S^{LEX}	$\hat{b}_{XEEF}$	49.71	71.84	**45.14**
S^{SYN}	$\hat{b}_{XEEF}$	47.54	53.77	43.32
		Early Cross Early Fusion		
		CONLL	**WNER**	**WGLD**
M^{STD}	b^*_{EXEF}	49.58	**77.32**	**61.69**
M^{LEX}	b^*_{EXEF}	49.79	66.22	53.54
M^{SYN}	b^*_{EXEF}	**51.53**	70.94	53.70
		Late Cross Early Fusion		
		CONLL	**WNER**	**WGLD**
M^{STD}	$\hat{b}_{LXEF}$	54.82	**75.70**	**54.73**
M^{LEX}	$\hat{b}_{LXEF}$	**56.53**	62.27	52.39

Table 4.9: F-measure results using N-degree fusion (NF). In ELXEF, $\hat{\mathfrak{b}}_{\text{ELXEF}} = L(M^{\text{LEX}}, X(S^{\text{SYN}}, M^{\text{LEX}}))$. For EELXEF, $\hat{\mathfrak{b}}_{\text{EELXEF}} = E(E(M^{\text{STD}}, L(M^{\text{LEX}}, X(S^{\text{SYN}}, M^{\text{LEX}}))), L(M^{\text{LEX}}, X(S^{\text{STD}}, M^{\text{LEX}})))$ for CONLL and $\hat{\mathfrak{b}}_{\text{EELXEF}} = E(E(M^{\text{STD}}, L(M^{\text{STD}}, X(S^{\text{SYN}}, M^{\text{STD}}))), L(M^{\text{LEX}}, X(S^{\text{SYN}}, M^{\text{LEX}})))$ for WNER and WGLD. The best result is obtained in EELXEF when $\alpha = 0.95$.

A	B	**Early Late Cross Early Fusion**		
		CONLL	**WNER**	**WGLD**
M^{STD}	$\hat{\mathfrak{b}}_{\text{ELXEF}}$	67.16	79.45	62.37
		Early Early Late Cross Early Fusion		
		CONLL	**WNER**	**WGLD**
M^{LEX}	$\hat{\mathfrak{b}}_{\text{EELXEF}}$	65.01	78.02	62.34
$M^{\text{LEX}}_{\alpha=0.95}$	$\hat{\mathfrak{b}}_{\text{EELXEF}}$	**79.67**	**81.79**	**67.05**
EF Baseline		78.90	80.04	63.20

Table 4.10: Supervised Recall and Unsupervised F-measure for the Semeval 2007 corpus. We also display the average number of clusters found by each fusion configuration.

Method	Recall (%)			FM (%)			# cl
	all	noun	verb	all	noun	verb	
Single Features							
M^{LEX}	79.20	82.10	75.80	72.70	76.90	67.90	4.13
M^{SYN}	79.10	81.60	76.20	69.30	69.40	69.20	4.47
Early Fusion							
$E(M^{LEX}, M^{SYN})$	78.70	81.11	76.10	74.00	76.66	71.11	4.46
Cross Early Fusion							
$X(S^{LEX}, M^{LEX})$	79.20	82.30	75.70	76.20	79.60	72.50	3.63
$X(S^{LEX}, M^{SYN})$	78.30	80.90	75.30	74.60	75.10	73.90	3.08
$X(S^{SYN}, M^{LEX})$	78.60	80.90	76.10	78.90	80.70	76.90	1.08
$X(S^{SYN}, M^{SYN})$	78.90	81.40	76.10	73.70	77.70	70.00	2.72
Cross Late Fusion							
$X(S^{SYN}, S^{LEX})$	78.70	80.90	76.20	78.90	80.80	76.80	1.01
$X(S^{LEX}, S^{SYN})$	78.80	80.90	76.06	78.70	80.50	76.80	1.33
Cross Late Early Fusion							
$X(X(S^{LEX}, S^{SYN}), M^{LEX})$	78.40	80.40	76.10	70.00	68.70	71.40	3.11
$X(X(S^{LEX}, S^{SYN}), M^{SYN})$	78.90	81.80	75.60	75.20	77.40	72.80	3.16
Early Cross Early Fusion							
$E(M^{LEX}, X(S^{LEX}, M^{LEX}))$	79.20	82.40	75.70	76.00	79.50	72.10	3.57
$E(M^{SYN}, X(S^{LEX}, M^{LEX}))$	78.30	80.50	75.80	75.20	75.40	75.00	1.95
Late Cross Early Fusion							
$L(M^{SYN}, X(S^{LEX}, M^{SYN}))$	78.60	81.10	75.80	67.80	71.40	63.80	4.22
$L(M^{LEX}, X(S^{LEX}, M^{LEX}))$	79.50	82.80	75.70	76.09	79.10	72.70	3.96
Early Late Cross Early Fusion							
$E(M^{LEX}, L(M^{SYN}, X(S^{LEX}, M^{SYN})))$	78.50	81.40	75.40	74.20	78.20	69.80	4.26
$E(M^{LEX}, L(M^{LEX}, X(S^{LEX}, M^{LEX})))$	79.50	82.70	75.90	75.80	78.50	72.70	3.99

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Linguistic Networks are useful methods to understand the nature of our language. In the literature, they are generally used to comprehend either the dynamics of words and other textual units within language, and to solve practical NLP tasks. Nonetheless, no mater the objective, they are usually based on the distributional hypothesis, that is, words will be found in similar contexts if they tend to be semantically related.

Distributional models are based on several parameters, such as the size of the context to be used, their linguistic type (either syntactic, lexical, etc.), the weight that affects each context-word co-occurrence, as well as determining how the semantic relatedness is computed. Indeed, most of the linguistic networks in the literature deal with a single type of contexts, either lexical or syntactical.

On the other hand, text data representations, described through contexts in a distributional framework, are sparse by nature: the large majority of the entries in a co-occurrence matrix are zero. This translates to greatly sparse features' matrices which represent problems for knowledge discovery methods as they do not have much information about words because each one of them is has a very low number of features. We considered the lack of heterogeneity and data sparsity two open challenges in textual representations.

To treat these concerns, on this thesis we proposed three contributions. The first one is a linguistic network. The second and third are practical: a method based on graph structure to find groups of related words. The third one, a method to generate denser matrix representations by combining different feature spaces.

The linguistic model we proposed unifies language networks by means of a hypergraph structure. We consider three different types of co-occurrence contexts in order to represent three distinct levels of semantic relatedness. These contexts are based on lexical and syntactic co-occurrences, which are unified with a hypergraph. This union yields words that are related by any of the three contexts and thus creating more links among words.

The second method leverages the structure of th hypergraph to find communities of words using contexts independently. These groups are found based on the intuition that words tend to group together around a single hub word which represents, broadly, the general semantic topic of these words.

The third method deals with both sparsity while making use of the heterogeneous features at the same time. By using fusion techniques, we mix syntactic and lexical context to produce more dense feature matrices.

In order to evaluate the methods and intuitions proposed, we performed experiments on two semantic tasks: WSI/WSD and NER.

Concerning WSI/WSD, we tested our hypergraph-based model over the Semeval 2007 and 2010 corpora. Using the free-scale presumption we found communities of words describing senses by using sentence-level lexical contexts and raw frequencies to weight the co-occurrences. Jaccard similarity was chosen to measure the relatedness among words. These parameters were defined experimentally. Also, we found that contrary to what we expected initially, the contexts defined by syntactic-based co-occurrences perform worse than lexical contexts. Additionally, we analyzed the differences of the two contexts in terms of performance for each word in the Semeval 2007 test corpus. In general, it seems that verbs are better off with syntactical contexts while nouns are best represented by lexical contexts.

With regard to our fusion technique method, we tested it over both WSI/WSD and NER tasks. We created new representation matrices that showed improvement in performance. In order to get to these improvements, which are consistent in the whole ensemble of datasets tested, we performed a high level of fusion aggregation. Not unlike relevant literature. Once again, lexical, and ad-hoc, features, proved more useful that syntactic features. We estimate this is due to the fact that syntactic features require larger corpora to actually populate the relations between words using dependency functions. Our experiments show that reducing the sparsity by combining heterogeneous features can ameliorate over using independent features and over the trivial feature concatenation. For all our experiments, while our results can be regarded as "baseline" performances, we do stay in the same ball-park of similar task-tailored methods.

5.2 Future Work

The work we present still has several research paths to be explored. The hypergraph model itself could utilize different contexts, going further than syntactic or lexical contexts, for example using morphological or even phonological contexts for words or other utterances. Even more, the constituent-based contexts are surely open for improvement: trying more literature approaches or devising intelligent ways to leverage the information provided by this syntactic parse.

The choice of weights and similarity measures is also open for exploration. While in my research experience, no great improvement comes from changing for one or the other, it is still interesting to understand in which task what parameters are the most appropriate. This is still an open research question in the domain. The hypergraph could be better leveraged by using hypergraph-specific methods, mainly through spectral analysis.

Regarding the network-based algorithm for WSI/WSD, a deeper errors' analysis would deep a larger glimpse on the behavior of nouns and verbs according to the context.

Concerning fusion techniques, a more principled way to determine what type of context with what type of fusion operation would indeed reduce the need for exploring the whole space of possibilities. Finally, comparing said methods with other well-established dimension reduction approaches would be interesting to understand the trade-offs of lower performance versus dimension reduction, while focusing on not-so-large corpora. Indeed, if the new wave of distributional representations, or word embeddings, has a shortcoming is that empirically it does not perform as well on smaller corpus. This may represent an avenue of opportunity to methods such as feature fusion functions.

Bibliography

[Aggarwal 2012] Charu C. Aggarwal and ChengXiang Zhai, editors. Mining text data. Springer, 2012. (Cited on pages 3 and 27.)

[Agirre 2006] Eneko Agirre, David Martínez, Oier López de Lacalle and Aitor Soroa. *Two Graph-based Algorithms for State-of-the-art WSD*. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, pages 585–593, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. (Cited on pages 42, 43 and 46.)

[Agirre 2007a] Eneko Agirre and Aitor Soroa. *Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 7–12, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on page 76.)

[Agirre 2007b] Eneko Agirre and Aitor Soroa. *Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 7–12, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on pages 95 and 96.)

[Agirre 2008] Eneko Agirre and Aitor Soroa. *Using the Multilingual Central Repository for Graph-Based Word Sense Disambiguation*. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). European Language Resources Association (ELRA), may 2008. (Cited on pages 40, 43 and 46.)

[Agirre 2009] Eneko Agirre and Aitor Soroa. *Personalizing PageRank for Word Sense Disambiguation*. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, EACL '09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. (Cited on pages 40, 42, 43 and 46.)

[Ah-Pine 2015] Julien Ah-Pine, Gabriela Csurka and Stéphane Clinchant. *Unsupervised Visual and Textual Information Fusion in CBMIR Using Graph-Based Methods*. vol. 33, no. 2, pages 9:1–9:31, 2015. (Cited on pages 31, 55, 58 and 92.)

[Ahn 2010] Yong-Yeol Ahn, James P Bagrow and Sune Lehmann. *Link communities reveal multiscale complexity in networks*. vol. 466, no. 7307, pages 761–764, 2010. (Cited on page 45.)

[Atrey 2010] Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El-Saddik and Mohan S. Kankanhalli. *Multimodal fusion for multimedia analysis: a survey*. vol. 16, no. 6, pages 345–379, 2010. (Cited on pages 53, 58 and 86.)

[Balasuriya 2009] Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy and James R. Curran. *Named Entity Recognition in Wikipedia*. In Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources, People's Web '09, pages 10–18, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. (Cited on pages 90 and 92.)

[Baroni 2010] Marco Baroni and Alessandro Lenci. *Distributional memory: A general framework for corpus-based semantics*. Computational Linguistics, vol. 36, no. 4, pages 673–721, 2010. (Cited on pages 18, 19 and 20.)

[Baroni 2014] Marco Baroni, Georgiana Dinu and Germán Kruszewski. *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.* In ACL (1), pages 238–247, 2014. (Cited on page 30.)

[Bender 2013] Emily M Bender. *Linguistic fundamentals for natural language processing: 100 essentials from morphology and syntax*. Synthesis Lectures on Human Language Technologies, vol. 6, no. 3, pages 1–184, 2013. (Cited on page 16.)

[Bengio 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Jauvin. *A neural probabilistic language model*. Journal of machine learning research, vol. 3, no. Feb, pages 1137–1155, 2003. (Cited on page 30.)

[Berge 1985] C. Berge. Graphs and hypergraphs. Elsevier, 1985. (Cited on page 47.)

[Bergsma 2012] Shane Bergsma, Matt Post and David Yarowsky. *Stylometric Analysis of Scientific Articles*. In Proceedings of the 2012 Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT '12, pages 327–337, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. (Cited on page 61.)

[Biemann 2006] Chris Biemann. *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. Association for Computational Linguistics, 2006. (Cited on page 45.)

[Bird 2006] Steven Bird. *NLTK: the natural language toolkit*. In Proceedings of the COLING/ACL on Interactive presentation sessions, pages 69–72. Association for Computational Linguistics, 2006. (Cited on page 18.)

[Booth 1955] Andrew Donald Booth. *Machine translation of languages, fourteen essays*. 1955. (Cited on page 12.)

[Brin 8 04] Sergey Brin and Lawrence Page. *The Anatomy of a Large-scale Hypertextual Web Search Engine*. vol. 30, no. 1-7, pages 107–117, 1998-04. (Cited on page 43.)

[Bronselaer 2013] Antoon Bronselaer and Gabriella Pasi. *An approach to graph-based analysis of textual documents*. In Proceedings of the 8th conference of the European Society for Fuzzy Logic and Technology, EUSFLAT-13, Milano, Italy, September 11-13, 2013, 2013. (Cited on pages 40, 44 and 46.)

[Bruni 2014] Elia Bruni, Nam-Khanh Tran and Marco Baroni. *Multimodal distributional semantics.* J. Artif. Intell. Res.(JAIR), vol. 49, no. 2014, pages 1–47, 2014. (Cited on pages 14 and 32.)

[Charton 2010] Eric Charton and Juan-Manuel Torres-Moreno. *NLGbAse: A Free Linguistic Resource for Natural Language Processing Systems*. In Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), May 2010. (Cited on page 59.)

[Choudhury 2009a] Monojit Choudhury and Animesh Mukherjee. *The Structure and Dynamics of Linguistic Networks*. In Niloy Ganguly, Andreas Deutsch and Animesh Mukherjee, editors, Dynamics On and Of Complex Networks, Modeling and Simulation in Science, Engineering and Technology, pages 145–166. BirkhÃd'user Boston, 2009. (Cited on pages 23, 24 and 25.)

[Choudhury 2009b] Monojit Choudhury and Animesh Mukherjee. *The Structure and Dynamics of Linguistic Networks*. In Niloy Ganguly, Andreas Deutsch and Animesh Mukherjee, editors, Dynamics On and Of Complex Networks, Modeling and Simulation in Science, Engineering and Technology, pages 145–166. Birkh?user Boston, 2009. (Cited on page 61.)

[Church 1990] Kenneth Ward Church and Patrick Hanks. *Word Association Norms, Mutual Information, and Lexicography*. Comput. Linguist., vol. 16, no. 1, pages 22–29, March 1990. (Cited on page 20.)

[Clark 2010] Alexander Clark, Chris Fox and Shalom Lappin. The handbook of computational linguistics and natural language processing. Wiley-Blackwell, 2010. (Cited on pages 2, 4, 17, 20 and 22.)

[Clark 2015] Stephen Clark. *Vector space models of lexical meaning*. Handbook of Contemporary Semantic Theory, The, pages 493–522, 2015. (Cited on page 22.)

[Clauset 2008] Aaron Clauset, Cristopher Moore and Mark EJ Newman. *Hierarchical structure and the prediction of missing links in networks*. vol. 453, no. 7191, pages 98–101, 2008. (Cited on page 45.)

[Clinchant 2011] Stéphane Clinchant, Julien Ah-Pine and Gabriela Csurka. *Semantic combination of textual and visual information in multimedia retrieval*. In ICMR, page 44. ACM, 2011. (Cited on pages 54, 55 and 92.)

[Collins 2002] Michael Collins. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP ’02, pages 1–8. Association for Computational Linguistics, 2002. (Cited on page 90.)

[Collobert 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu and Pavel P. Kuksa. *Natural Language Processing (almost) from Scratch*. CoRR, vol. abs/1103.0398, 2011. (Cited on pages 14 and 30.)

[Daume 2006] Harold Charles Daume III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, 2006. AAI3337548. (Cited on pages 15 and 90.)

[de Cruys 2011] Tim Van de Cruys and Marianna Apidianaki. *Latent Semantic Word Sense Induction and Disambiguation*. In ACL, pages 1476–1485. The Association for Computer Linguistics, 2011. (Cited on page 96.)

[De Marneffe 2006] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning*et al.* *Generating typed dependency parses from phrase structure parses*. In Proceedings of LREC, volume 6, pages 449–454. Genoa Italy, 2006. (Cited on page 18.)

[De Marneffe 2008] Marie-Catherine De Marneffe and Christopher D Manning. *Stanford typed dependencies manual*. Technical report, Technical report, Stanford University, 2008. (Cited on page 18.)

[De Saussure 1916] Ferdinand De Saussure. *Course in general linguistics (trans. Roy Harris)*. London: Duckworth, 1916. (Cited on page 12.)

[Di Marco 2011] Antonio Di Marco and Roberto Navigli. *Clustering Web Search Results with Maximum Spanning Trees*. In Proceedings of the 12th International Conference on Artificial Intelligence Around Man and Beyond, AI*IA'11, pages 201–212, Berlin, Heidelberg, 2011. Springer-Verlag. (Cited on pages 38, 42 and 46.)

[Di Marco 2013] Antonio Di Marco and Roberto Navigli. *Clustering and diversifying web search results with graph-based word sense induction*. vol. 39, no. 3, pages 709–754, 2013. (Cited on pages 42 and 46.)

[Estrada 2005] Ernesto Estrada and Juan A Rodriguez-Velazquez. *Complex networks as hypergraphs*. 2005. (Cited on page 38.)

[Fabre 2015] Cécile Fabre and Alessandro Lenci. *Distributional Semantics Today Introduction to the special issue*. Traitement Automatique des Langues, vol. 56, no. 2, pages 7–20, 2015. (Cited on page 13.)

[Ferret 2010] Olivier Ferret. *Testing Semantic Similarity Measures for Extracting Synonyms from a Corpus.* In LREC, volume 10, pages 3338–3343, 2010. (Cited on page 22.)

[Firth 1957] J. R. Firth. *A synopsis of linguistic theory 1930-55.* vol. 1952-59, pages 1–32, 1957. (Cited on page 12.)

[Flickinger 2010] Dan Flickinger, Stephan Oepen and Gisle Ytrestol. *WikiWoods: Syntacto-Semantic Annotation for English Wikipedia*. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), May 2010. (Cited on page 60.)

[Gabrilovich 2007] Evgeniy Gabrilovich and Shaul Markovitch. *Computing semantic relatedness using wikipedia-based explicit semantic analysis.* In IJcAI, volume 7, pages 1606–1611, 2007. (Cited on pages vii and 31.)

[Garten 2010] Yael Garten, Adrien Coulet and Russ B Altman. *Recent progress in automatically extracting information from the pharmacogenomic literature*. Pharmacogenomics, vol. 11, no. 10, pages 1467–1489, 2010. (Cited on page 1.)

[Gialampoukidis 2016] Ilias Gialampoukidis, Anastasia Moumtzidou, Dimitris Liparas, Stefanos Vrochidis and Ioannis Kompatsiaris. *A hybrid graph-based and non-linear late fusion approach for multimedia retrieval*. In CBMI, pages 1–6. IEEE, 2016. (Cited on page 92.)

[Giuseppe Attardi 2015] Giuseppe Attardi. *WikiExtractor*. `https://github.com/attardi/wikiextractor`, 2015. (Cited on pages 60 and 62.)

[Goldberg 2013] Yoav Goldberg and Jon Orwant. *A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books.* In * SEM@ NAACL-HLT, pages 241–247, 2013. (Cited on page 19.)

[Goyal 2014] Kartik Goyal and Eduard H. Hovy. *Unsupervised Word Sense Induction using Distributional Statistics*. In COLING, pages 1302–1310. ACL, 2014. (Cited on page 90.)

[Guille 2016] Adrien Guille, Edmundo-Pavel Soriano-Morales and Ciprian-Octavian Truica. *Topic modeling and hypergraph mining to analyze the EGC conference history*. In 16ème Journées Francophones Extraction et Gestion des Connaissances, EGC 2016, 18-22 Janvier 2016, Reims, France, pages 383–394, 2016. (Cited on page 10.)

[Han 2009] Jiawei Han. *Mining Heterogeneous Information Networks by Exploring the Power of Links*. In Discovery Science, volume 5808 of *Lecture Notes in Computer Science*, pages 13–30. Springer Berlin Heidelberg, 2009. (Cited on page 27.)

[Harris 1954] Zellig Harris. *Distributional structure*. vol. 10, no. 23, pages 146–162, 1954. (Cited on page 11.)

[Heintz 2014] Benjamin Heintz and Abhishek Chandra. *Beyond graphs: toward scalable hypergraph analysis systems*. vol. 41, no. 4, pages 94–97, 2014. (Cited on page 48.)

[Hope 2013] David Hope and Bill Keller. *MaxMax: a graph-based soft clustering algorithm applied to word sense induction*. In Computational Linguistics and Intelligent Text Processing, pages 368–381. Springer, 2013. (Cited on page 45.)

[Hope 3 06] David Hope and Bill Keller. *UoS: A Graph-Based System for Graded Word Sense Induction*. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 689–694. Association for Computational Linguistics, 2013-06. (Cited on pages 42, 44 and 46.)

[Jaccard 1908] P. Jaccard. Nouvelles recherches sur la distribution florale. Bulletin de la Société vaudoise des sciences naturelles. Impr. Réunies, 1908. (Cited on page 23.)

[Jordi Atserias 2008] Massimiliano Ciaramita Jordi Atserias Hugo Zaragoza and Giuseppe Attardi. *Semantically Annotated Snapshot of the English Wikipedia*. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). European Language Resources Association (ELRA), May 2008. (Cited on pages 59, 61 and 65.)

[Jurafsky 2009] Dan Jurafsky and James H. Martin. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd edition. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2nd édition, 2009. (Cited on pages 2, 3, 4, 14, 15, 16, 18 and 20.)

[Jurafsky 2017] Dan Jurafsky and James H. Martin. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd edition. 2017. (Cited on pages 20, 21 and 28.)

[Jurgens 2011] David Jurgens. *Word Sense Induction by Community Detection*. In Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing, TextGraphs-6, pages 24–28, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. (Cited on pages 38, 44 and 46.)

[Katz 2014] Gilad Katz, Anna Shtock, Oren Kurland, Bracha Shapira and Lior Rokach. *Wikipedia-based query performance prediction*. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pages 1235–1238. ACM, 2014. (Cited on page 59.)

[Kiela 2014] Douwe Kiela and Stephen Clark. *A systematic study of semantic vector space model parameters*. In Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC) at EACL, pages 21–30, 2014. (Cited on pages 19, 20, 22 and 84.)

[Kivimäki 2013] Ilkka Kivimäki, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Cédrick Fairon, Hugues Bersini and Marco Saerens. *A graph-based approach to skill extraction from text*. page 79, 2013. (Cited on pages 41, 42, 43, 44 and 46.)

[Klapaftis 2007] Ioannis P. Klapaftis and Suresh Manandhar. *UOY: A Hypergraph Model for Word Sense Induction & Disambiguation*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 414–417, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on pages 38, 44, 46, 48, 72 and 73.)

[Klapaftis 2008] Ioannis P. Klapaftis and Suresh Manandhar. *Word Sense Induction Using Graphs of Collocations*. In Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence, pages 298–302. IOS Press, 2008. (Cited on pages 38, 40, 44 and 46.)

[Klapaftis 2010] Ioannis P. Klapaftis and Suresh Manandhar. *Word Sense Induction & Disambiguation Using Hierarchical Random Graphs*. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10, pages 745–755, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Cited on pages 40, 42, 44 and 46.)

[Lebret 2013] Rémi Lebret, Joël Legrand and Ronan Collobert. *Is deep learning really necessary for word embeddings?* Technical report, Idiap, 2013. (Cited on page 14.)

[Lee 2001] Daniel D Lee and H Sebastian Seung. *Algorithms for non-negative matrix factorization*. In Advances in neural information processing systems, pages 556–562, 2001. (Cited on page 29.)

[Lehecka 2015] Tomas Lehecka. *Handbook of Pragmatics 2015*. chapter 2. John Benjamins, 2015. (Cited on page 12.)

[Levy 2014a] Omer Levy and Yoav Goldberg. *Dependency-Based Word Embeddings*. In ACL (2), pages 302–308. The Association for Computer Linguistics, 2014. (Cited on pages 15, 18, 19, 32, 89, 90 and 93.)

[Levy 2014b] Omer Levy and Yoav Goldberg. *Neural word embedding as implicit matrix factorization*. In Advances in neural information processing systems, pages 2177–2185, 2014. (Cited on pages 14, 15 and 30.)

[Levy 2015] Omer Levy, Yoav Goldberg and Ido Dagan. *Improving distributional similarity with lessons learned from word embeddings*. Transactions of the Association for Computational Linguistics, vol. 3, pages 211–225, 2015. (Cited on pages 20, 21, 29 and 30.)

[Lin 1997] Dekang Lin. *Using Syntactic Dependency As Local Context to Resolve Word Sense Ambiguity*. In Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics, EACL '97, pages 64–71, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. (Cited on pages 18 and 19.)

[Liu 2011] Haishan Liu, Paea Le Pendu, Ruoming Jin and Dejing Dou. *A Hypergraph-based Method for Discovering Semantically Associated Itemsets*. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, ICDM '11, pages 398–406. IEEE Computer Society, 2011. (Cited on pages 39, 44, 46 and 48.)

[Liu 2012] Bing Liu and Lei Zhang. *A survey of opinion mining and sentiment analysis*. In Mining text data, pages 415–463. Springer, 2012. (Cited on page 4.)

[Loudcher 2015] Sabine Loudcher, Wararat Jakawat, Edmundo-Pavel Soriano-Morales and Cécile Favre. *Combining OLAP and information networks for bibliographic data analysis: a survey*. Scientometrics, vol. 103, no. 2, pages 471–487, 2015. (Cited on page 10.)

[Manandhar 0 07] Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach and Sameer Pradhan. *SemEval-2010 Task 14: Word Sense Induction & Disambiguation*. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 63–68. Association for Computational Linguistics, 2010-07. (Cited on pages 76 and 82.)

[Manning 1999] Christopher D Manning, Hinrich Schütze*et al.* Foundations of statistical natural language processing, volume 999. MIT Press, 1999. (Cited on page 20.)

[Manning 2008] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Introduction to information retrieval. Cambridge University Press, New York, NY, USA, 2008. (Cited on page 22.)

[Manning 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky. *The Stanford CoreNLP Natural Language Processing Toolkit*. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60, 2014. (Cited on pages 63 and 91.)

[Massung 2013] S. Massung, Chengxiang Zhai and J. Hockenmaier. *Structural Parse Tree Features for Text Representation*. In Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on, pages 9–16, 2013. (Cited on page 61.)

[Matuschek 3 05] Michael Matuschek and Iryna Gurevych. *Dijkstra-WSA: A Graph-Based Approach to Word Sense Alignment*. vol. 1, pages 151–164, 2013-05. (Cited on pages 43 and 46.)

[Mihalcea 2004] Rada Mihalcea, Paul Tarau and Elizabeth Figa. *PageRank on Semantic Networks, with Application to Word Sense Disambiguation*. In Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. (Cited on pages 40, 42, 43 and 46.)

[Mihalcea 2011] Rada F. Mihalcea and Dragomir R. Radev. Graph-based natural language processing and information retrieval. Cambridge University Press, 1st édition, 2011. (Cited on pages 23, 24, 25 and 61.)

[Mikolov 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado and Jeff Dean. *Distributed representations of words and phrases and their compositionality*. In Advances in neural information processing systems, pages 3111–3119, 2013. (Cited on pages 14, 15 and 30.)

[Molino 2017] Piero Molino. *Word Embeddings Past, Present and Future*, 06 2017. AI With The Best AIWTB 2017. (Cited on pages vii, 13 and 28.)

[Moro 2014] Andrea Moro, Alessandro Raganato and Roberto Navigli. *Entity Linking meets Word Sense Disambiguation: a Unified Approach*. vol. 2, pages 231–244, 2014. (Cited on pages 40, 43, 44 and 46.)

[Murphy 2012] Kevin P. Murphy. Machine learning: A probabilistic perspective. The MIT Press, 2012. (Cited on page 43.)

[Nadeau 2007] David Nadeau and Satoshi Sekine. *A survey of named entity recognition and classification*. Lingvisticae Investigationes, vol. 30, no. 1, pages 3–26, 2007. (Cited on page 4.)

[Nastase 2015] Vivi Nastase, Rada Mihalcea and Dragomir R Radev. *A survey of graphs in natural language processing*. Natural Language Engineering, vol. 21, no. 5, pages 665–698, 2015. (Cited on pages 24 and 36.)

[Navigli 2007] Roberto Navigli and Mirella Lapata. *Graph Connectivity Measures for Unsupervised Word Sense Disambiguation*. In Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, pages 1683–1688, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. (Cited on pages 38, 40, 42, 43 and 46.)

[Navigli 2010] Roberto Navigli and Giuseppe Crisafulli. *Inducing Word Senses to Improve Web Search Result Clustering*. EMNLP '10, pages 116–126, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Cited on pages 38, 42 and 46.)

[Nida 1979] Eugene A Nida. A componential analysis of meaning: An introduction to semantic structures, volume 57. Walter de Gruyter GmbH & Co KG, 1979. (Cited on page 12.)

[Nivre 2016] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo

Pyysalo, Natalia Silveira*et al. Universal Dependencies v1: A Multilingual Treebank Collection.* In LREC, 2016. (Cited on page 18.)

[Niwa 1994] Yoshiki Niwa and Yoshihiko Nitta. *Co-occurrence vectors from corpora vs. distance vectors from dictionaries*. In Proceedings of the 15th conference on Computational linguistics-Volume 1, pages 304–309. Association for Computational Linguistics, 1994. (Cited on page 20.)

[Ozkan 2010] Derya Ozkan, Kenji Sagae and Louis-Philippe Morency. *Latent mixture of discriminative experts for multimodal prediction modeling*. In Proceedings of the 23rd International Conference on Computational Linguistics, pages 860–868. Association for Computational Linguistics, 2010. (Cited on page 31.)

[Padó 2003] Sebastian Padó and Mirella Lapata. *Constructing semantic space models from parsed corpora*. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, pages 128–135. Association for Computational Linguistics, 2003. (Cited on page 19.)

[Panchenko 2017] A. Panchenko, S. Faralli, S. P. Ponzetto and C. Biemann. *Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017). The Association for Computer Linguistics, 2017. (Cited on pages 18, 19 and 90.)

[Pedersen 2010] Ted Pedersen. *Duluth-WSI: SenseClusters applied to the sense induction task of SemEval-2*. In Proceedings of the 5th international workshop on semantic evaluation, pages 363–366. Association for Computational Linguistics, 2010. (Cited on pages 82 and 83.)

[Pennington 2014] Jeffrey Pennington, Richard Socher and Christopher D Manning. *Glove: Global vectors for word representation.* In EMNLP, volume 14, pages 1532–1543, 2014. (Cited on page 30.)

[Périnet 2015a] Amandine Périnet. *Analyse distributionnelle appliquée aux textes de spécialité : réduction de la dispersion des données par abstraction des contextes*. Theses, Université Paris 13 ; Laboratoire d'Informatique Médicale et d'Ingénieurie des Connaissances en e-Santé, March 2015. (Cited on pages 13, 14, 18 and 28.)

[Périnet 2015b] Amandine Périnet and Thierry Hamon. *Analyse distributionnelle appliquée aux textes de spécialité - Réduction de la dispersion des données par abstraction des contextes*. TAL, vol. 56, no. 2, 2015. (Cited on page 27.)

[Phan 2008] Xuan-Hieu Phan, Le-Minh Nguyen and Susumu Horiguchi. *Learning to classify short and sparse text & web with hidden topics from large-scale data collections*. In Proceedings of the 17th international conference on World Wide Web, pages 91–100. ACM, 2008. (Cited on page 28.)

[Qian 4 08] Tao Qian, Donghong JI, Mingyao Zhang, Chong Teng and Congling Xia. *Word Sense Induction Using Lexical Chain based Hypergraph Model*. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1601–1611. Dublin City University and Association for Computational Linguistics, 2014-08. (Cited on pages 38, 44, 46 and 48.)

[Ratinov 2009] Lev-Arie Ratinov and Dan Roth. *Design Challenges and Misconceptions in Named Entity Recognition*. In CoNLL, pages 147–155. ACL, 2009. (Cited on pages 28 and 91.)

[Sagae 9 10] Kenji Sagae and Andrew S. Gordon. *Clustering Words by Syntactic Similarity Improves Dependency Parsing of Predicate-Argument Structures*. In International Conference on Parsing Technologies (IWPT-09), 2009-10. (Cited on page 61.)

[Sahlgren 2006] Magnus Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm University, 2006. (Cited on pages 15, 18, 28 and 32.)

[Sahlgren 2008] Magnus Sahlgren. *The distributional hypothesis*. Italian Journal of Disability Studies, vol. 20, pages 33–53, 2008. (Cited on pages vii, 12, 13 and 15.)

[Saluja 2013] Avneesh Saluja and Jirı Navrátil. *Graph-Based Unsupervised Learning of Word Similarities Using Heterogeneous Feature Types*. page 29, 2013. (Cited on pages 41, 42, 44 and 46.)

[Sang 2003] Erik F. Tjong Kim Sang and Fien De Meulder. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. In CoNLL, pages 142–147. ACL, 2003. (Cited on page 91.)

[Schaeffer 7 08] Satu Elisa Schaeffer. *Survey: Graph Clustering*. vol. 1, no. 1, pages 27–64, 2007-08. (Cited on page 44.)

[Schenkel 2007] Ralf Schenkel, Fabian M. Suchanek and Gjergji Kasneci. *YAWN: A Semantically Annotated Wikipedia XML Corpus*. In Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 7.-9. März 2007, Aachen, Germany, pages 277–291, 2007. (Cited on page 60.)

[Schuster 2016] Sebastian Schuster and Christopher D Manning. *Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks.* In LREC, 2016. (Cited on page 18.)

[Schütze 1993] Hinrich Schütze and Jan Pedersen. *A vector model for syntagmatic and paradigmatic relatedness*. In Proceedings of the 9th Annual Conference of the UW Centre for the New OED and Text Research, pages 104–113. Oxford, 1993. (Cited on page 13.)

[Shaoul 2010] C. Shaoul and C. Westbury. *The Westbury Lab Wikipedia Corpus*. `http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`, 2010. (Cited on page 60.)

[Silberer 2010] Carina Silberer and Simone Paolo Ponzetto. *UHD: Cross-lingual Word Sense Disambiguation Using Multilingual Co-occurrence Graphs*. In Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval '10, pages 134–137, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Cited on pages 40, 43, 44 and 46.)

[Sinha 2007] Ravi Sinha and Rada Mihalcea. *Unsupervised Graph-basedWord Sense Disambiguation Using Measures of Word Semantic Similarity*. ICSC '07, pages 363–369, 2007. (Cited on pages 40, 42 and 46.)

[Soriano-Morales 2016a] Edmundo-Pavel Soriano-Morales, Julien Ah-Pine and Sabine Loudcher. *Hypergraph Modelization of a Syntactically Annotated English Wikipedia Dump*. In Proceedings of the Tenth International Conference on Language

Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), may 2016. (Cited on page 10.)

[Soriano-Morales 2016b] Edmundo-Pavel Soriano-Morales, Julien Ah-Pine and Sabine Loudcher. *Using a Heterogeneous Linguistic Network for Word Sense Induction and Disambiguation*. In Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING), 2016. (Cited on page 10.)

[Soriano-Morales 2017] Edmundo-Pavel Soriano-Morales, Julien Ah-Pine and Sabine Loudcher. *Fusion Techniques for Named Entity Recognition and Word Sense Induction and Disambiguation*. In International Conference on Discovery Science (DS 2017), Lecture Notes in Computer Science, page To appear, 2017. (Cited on page 10.)

[Sugiyama 2016] Masashi Sugiyama. Introduction to statistical machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016. (Cited on page 1.)

[Tsatsaronis 7 01] George Tsatsaronis, Michalis Vazirgiannis and Ion Androutsopoulos. *Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri*. 2007-01. (Cited on pages 40, 42, 43 and 46.)

[Turney 2010] Peter D. Turney and Patrick Pantel. *From Frequency to Meaning: Vector Space Models of Semantics*. CoRR, vol. abs/1003.1141, 2010. (Cited on pages 12, 13, 19 and 20.)

[Van de Cruys 2011] Tim Van de Cruys and Marianna Apidianaki. *Latent Semantic Word Sense Induction and Disambiguation*. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 1476–1485, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. (Cited on pages 30, 82 and 83.)

[Véronis 2004] Jean Véronis. *HyperLex: lexical cartography for information retrieval*. vol. 18, no. 3, pages 223 – 252, 2004. (Cited on pages 38, 44, 46 and 72.)

[Vulić 2011] Ivan Vulić, Wim De Smet and Marie-Francine Moens. *Identifying word translations from comparable corpora using latent topic models*. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 479–484. Association for Computational Linguistics, 2011. (Cited on page 59.)

[Wu 2010] Fei Wu and Daniel S. Weld. *Open Information Extraction Using Wikipedia*. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Cited on page 59.)

[Yeh 2009] Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre and Aitor Soroa. *WikiWalk: Random Walks on Wikipedia for Semantic Relatedness*. In Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing, TextGraphs-4, pages 41–49, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. (Cited on pages 40, 44 and 46.)

[Yu 2014] Shoou-I Yu, Lu Jiang, Zexi Mao, Xiaojun Chang, Xingzhong Du, Chuang Gan, Zhenzhong Lan, Zhongwen Xu, Xuanchong Li, Yang Cai*et al. Informedia@ trecvid 2014 med and mer*. In NIST TRECVID Video Retrieval Evaluation Workshop, volume 24, 2014. (Cited on page 58.)