UNIVERSITÉ DE LYON - LUMIÈRE LYON 2

ECOLE DOCTORALE INFORMATIQUE ET MATHÉMATIQUES

T H È S E

pour obtenir le grade de

DOCTEUR EN INFORMATIQUE

présentée par

Edmundo-Pavel Soriano-Morales

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

préparée au sein du laboratoire ERIC

sous la direction de Sabine Loudcher et Julien Ah-Pine

Soutenue publiquement le devant le jury :

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 types of techniques are regularly used exclusively to combine multimedia 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 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 five different open-access 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: first, we show that using fusion-enriched

heterogeneous features, coming from our proposed linguistic network, we outperform the performance of single features' systems and other basic baselines. We note that using single fusion operators is not efficient compared to using a combination of them in order to obtain a final space representation. We show that the features added by each combined fusion operation are important towards the models predicting the appropriate classes. We test the enriched representations on both WSI/WSD and NER tasks. Secondly, we address the WSI/WSD task with our network-based proposed method. While based on previous work, we improve it by obtaining better overall performance and reducing the number of parameters needed. We also discuss the use of either lexical or syntactic networks to solve the 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

Résumé

Mots clés. Traitement automatique du langage naturel, réseaux linguistiques, représentation de mots, techniques de fusion, reconnaissance d'entités nommées, induction et désambiguïsation du sens des mots.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

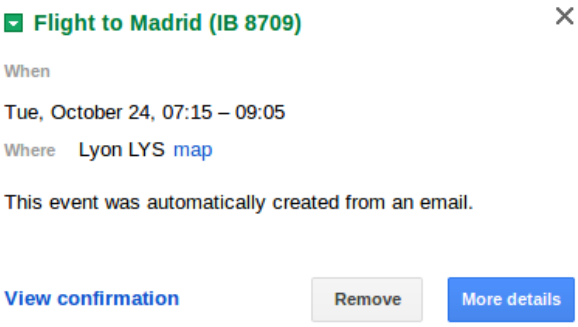
Contents

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/

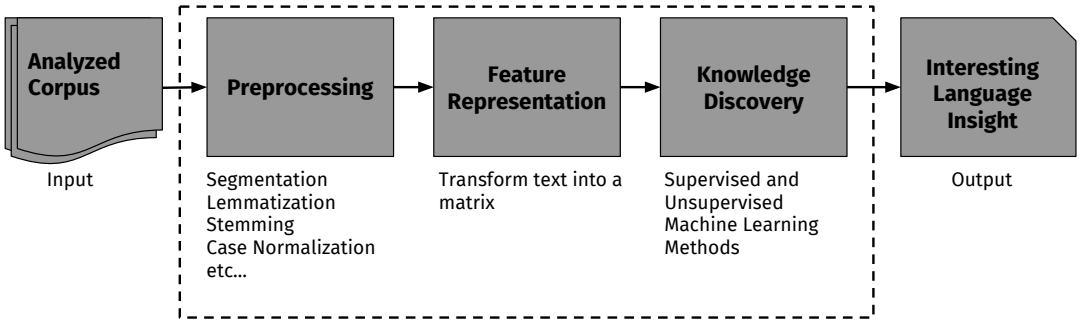


Figure 1.2: Typical steps of Natural Language Processing applications.

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

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 as efficient as possible using parallelized solutions.

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. We present this contribution in Chapter 3, particularly in section 3.3.

Contribution During the last decade, graphs have been used to model textual data given its ability to naturally describe the dynamics and structured of text. We propose to represent a linguistic resource in the form of a heterogeneous language network. This model then can be used as a comprehensive data source to address 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. 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 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 co-occurrence 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 co-occurrences 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 co-occurrences.

Towards addressing both challenges previously described we pose the following questions: **how to alleviate data sparsity on textual data?** Concerning combining linguistic features, **how can we produce a single textual representation (one unified feature space) 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 of 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, 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. This contribution is discussed in the linguistic representation introduced in Chapter 3, Section 3.3 and in its application in Chapter 4, Section 4.2.

1.2.3 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

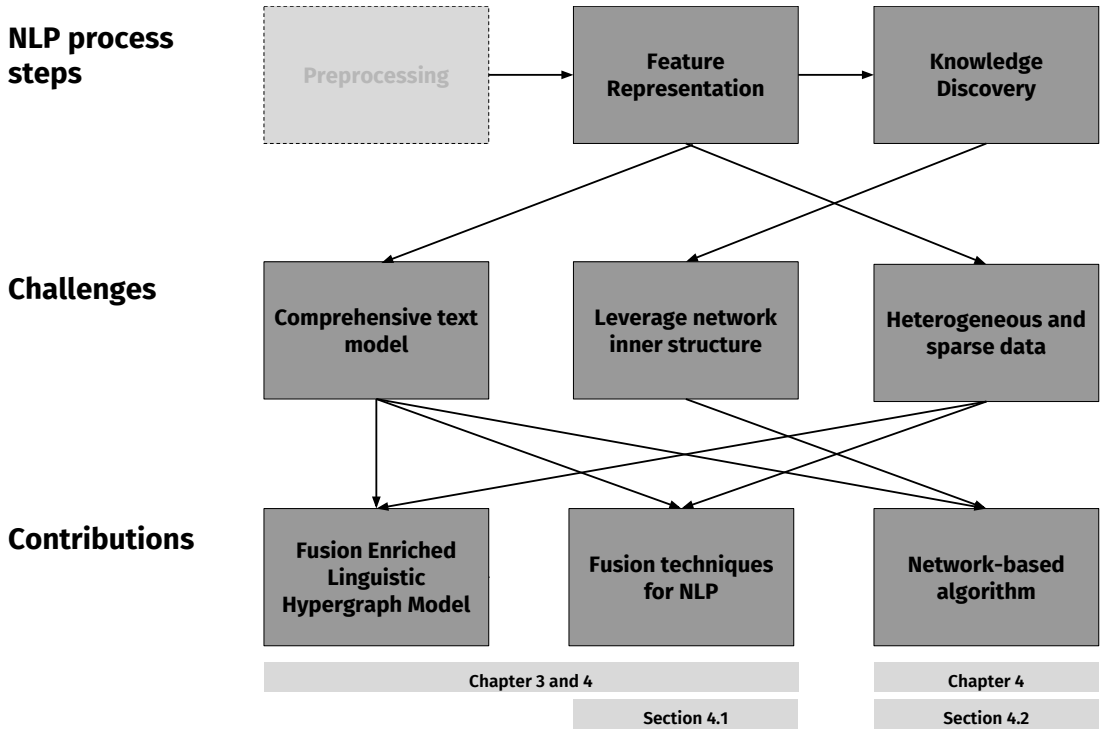


Figure 1.3: Block diagram of the NLP steps of interest, the challenges we address, and the contributions we propose.

proposed linguistic model and go beyond the classic homogeneous co-occurrences 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 improve the overall performance compared to other similar graph-based techniques.

This contribution is presented as an application, to solve word sense disambiguation, in Chapter 4, section 4.3.

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 which contextualizes our first and second contributions. We present and define a novel structure to hold language information based on a fusion enriched hypergraph linguistic model. Initially, as the first set of characteristics of the model, we discuss its characteristics and the intuitions behind its conception: the choice of the structure, the role of nodes and heterogeneous edges and the type of features stored. Then, as the second important set of properties of the proposition, we introduce a set of techniques to make use of heterogeneous relations while dealing with sparsity in order to produce an unique enriched representation. Namely, we present the feature fusion techniques. These methods are integral part of our contributions. Finally, we present a concrete application of our hypergraph model, an instantiation of the model based on 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, we 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) rinedefers 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 places the DH 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 we should use to determine similarities, as well as what does the meaning of the contextual relations between words implies. 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 grammati-

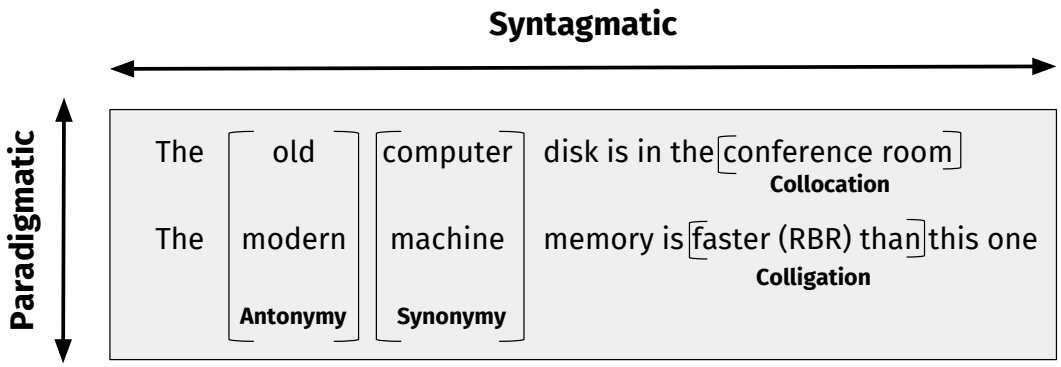


Figure 2.2: Examples of syntagmatic and paradigmatic contexts and their corresponding semantic relationships. Based on the examples by [Sahlgren 2008, Molino 2017].

cal category) [Lehecka 2015]. On the other hand, paradigmatic relations associate words according to whether they appear or not in the same context. Put differently, these are words that tend to not appear together while sharing the same context. These type of relations define classic semantic relations such as synonymy-antonymy or hypernymy-hiponymy. Coupling both the DH with structuralist theory gives birth to the more fine-grained 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), we can say that they share a paradigmatic context, as they tend to not occur together, either we write old *or* modern, while sharing the same context (either *computer* or *machine*). In this case, that leads to an antonym semantic relation. Something similar happens with the 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), which is *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 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.

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

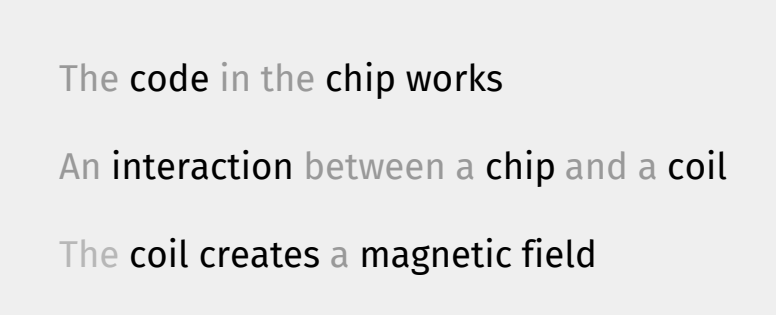


Figure 2.3: Example text. Functional words are grayed-out.

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 [Daumé III 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 synonymy. 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.

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

2.1.2 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: *PP*), 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:

[1] The nomenclature used is the Penn Tree Bank annotation.

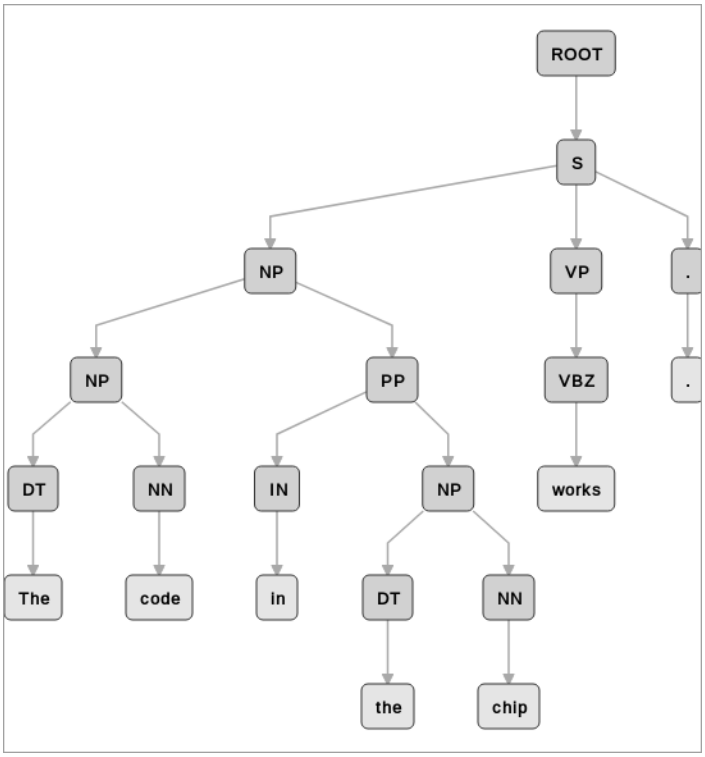


Figure 2.4: Constituents tree parse of the phrase *The code in the chip works.*

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.

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 the non-functional words that co-occur with each word in a given constituent, in this case a noun phrase (NP).

Dependency Tree While a parse tree represent the units existing within a specific group of syntactically related words, 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: $(\texttt{head}, \texttt{relation}, \texttt{dependent})$ 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: $(\texttt{conj} : \texttt{and}, \texttt{coil}, \texttt{head})$. 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 de-

[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].

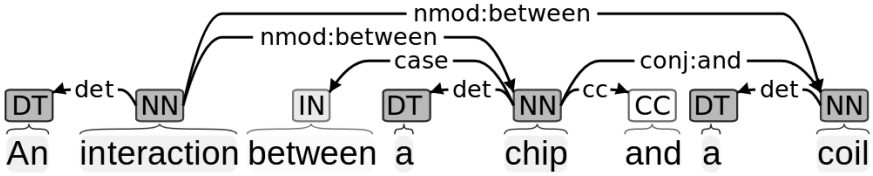


Figure 2.5: Dependency tree parse of the phrase *An interaction between a chip and a coil.*

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

cidedly 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 contexts 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

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

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.

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:

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

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:

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

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:

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

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:

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

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:

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

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: $sim_{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. 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

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

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 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 2.1.

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

[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.

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.

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

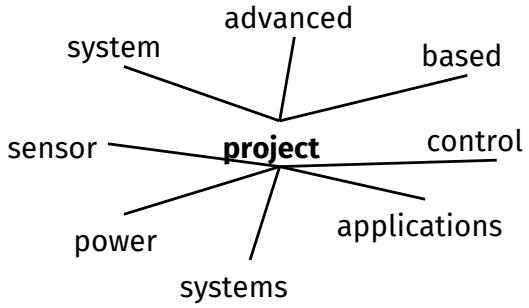


Figure 2.6: Lexical network for the word *project*.

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 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.

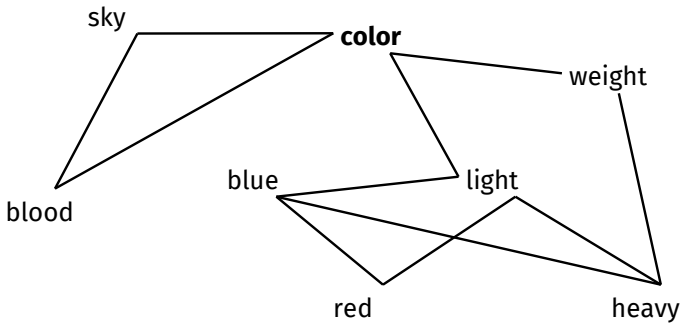


Figure 2.7: Syntactic Network of the word *color*.

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.

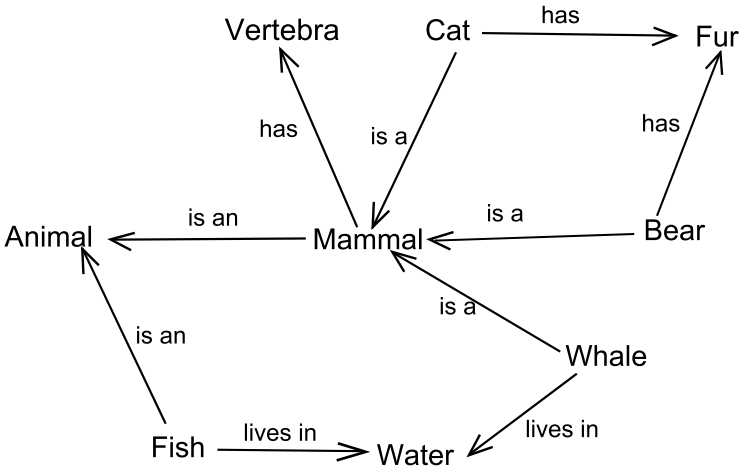


Figure 2.8: Semantic Network of the word *mammal*.

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 the 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

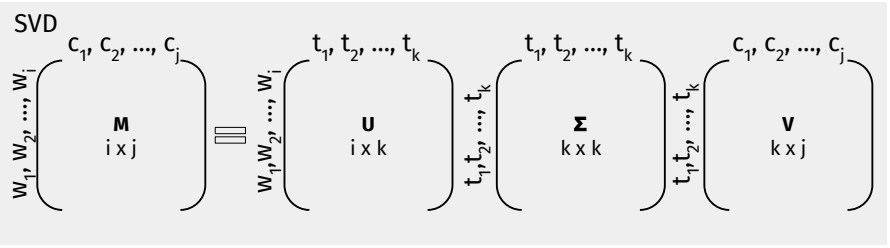


Figure 2.9: Factorization of the matrix M using SVD. Three matrices are generated: U, Σ, and V. U is the one that really interest us, it contains $|W|$ words and $k, k < C$ columns.

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 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]

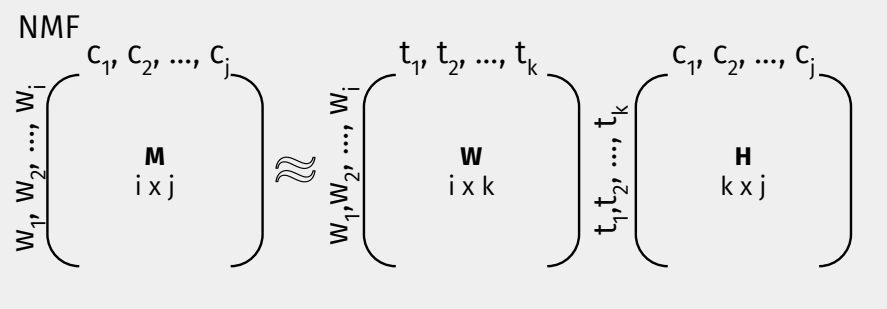


Figure 2.10: Factorization of the matrix M using NMF. Two matrices are generated: W and H. W contains the words in a k reduced space. H contains the contexts in the reduced space.

(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 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].

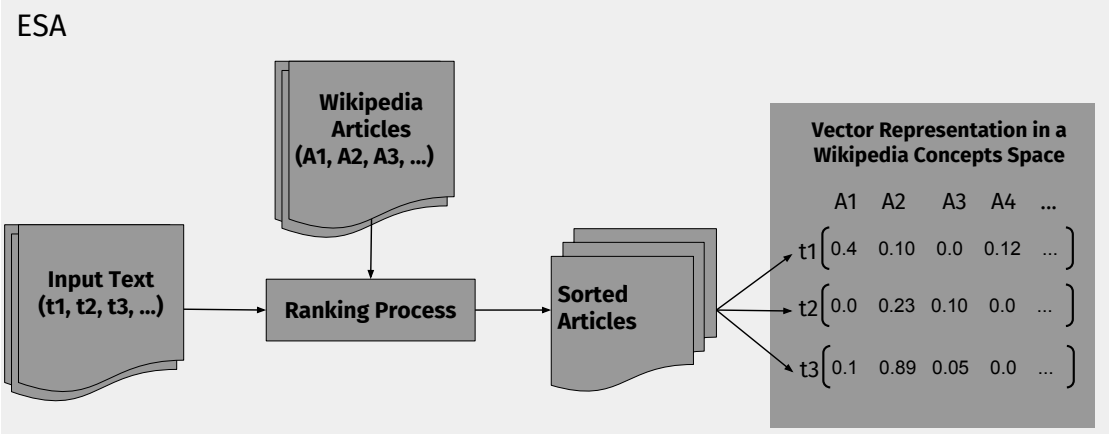


Figure 2.11: Block description of the ESA process by [Gabrilovich 2007]. The method ranks Wikipedia articles according to an input document. The document is then described by a weighted vector, where each dimension is a Wikipedia article, or concept.

Enriched Explicit Representations While implicit representations modify the interpretation of the columns of the co-occurrence matrix so that they are no longer 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, which makes dealing with them may represent a computational challenge. On the other hand, vectors based on non-English versions may suffer from the reduction of textual data, as the 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 multimedia

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

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 powerful 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 multimedia 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 Conclusion

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 interchange 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].

These contexts need to be represented computationally in order to perform some Using co-occurrence matrices, we can 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.

In the following chapter we describe our two first propositions which address the issue of using heterogeneous information to represent a term and alleviating the data sparsity that comes with such types of textual representations.

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.

The first two contributions of this thesis are contained in a fusion enriched hypergraph linguistic model proposition. The model consists on two components 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 see the block diagram of the ensemble of our propositions on Figure 3.1. There, we can observe our enriched linguistic model proposition, comprising our first two propositions, which is the focus of this chapter, as well as the instantiation of said model using a Wikipedia based corpus.

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. The three of them are addressed with our propositions. In the following chapter, we test the practicality of our proposal with well-established tasks and related evaluation corpora, which we use as benchmark input data in our experiments.

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).

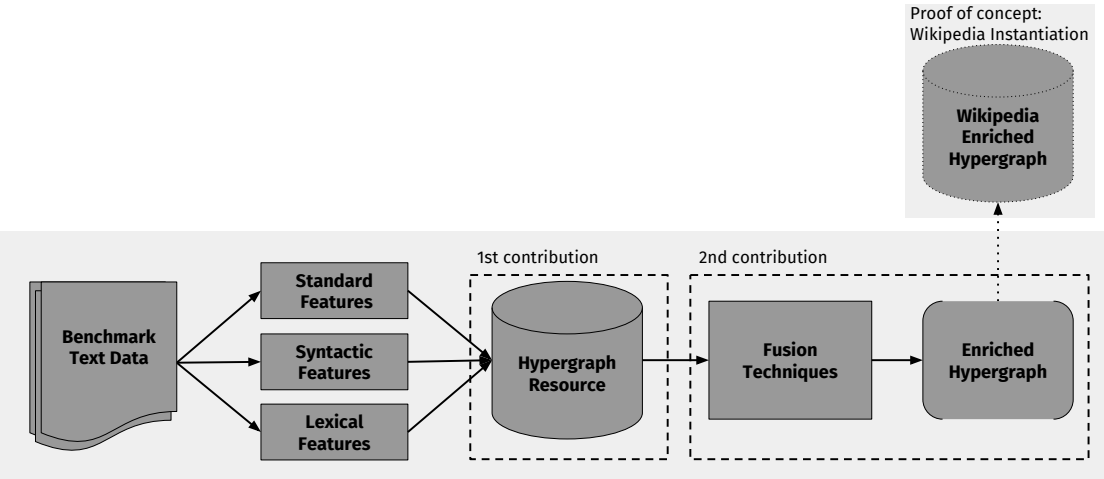


Figure 3.1: Modular description of the first two contributions of this work: an enriched hypergraph resource.

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 in 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 com-

plementary 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 2014] represent, as we do, 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 that sense, in [Qian 2014] they make use of this type of representation to solve the task of word sense induction. Briefly, in this task we have to determine a set of senses for a given target word in a corpus, according to its context (a context here being usually a paragraph where the target word occurs). In their paper, given an input document with several context instances for each target word, they first extract lexical chains (set of semantically related words) from the contexts using a topic-modeling based technique. Secondly, a hypergraph is built where the vertices represent words and the hyperedges link two or more words if they exist in the same lexical chain. Thirdly, the hypergraph

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

is clustered and groups of words are found which are considered to be the senses of the target word. Lastly, to assign these senses to each target word instance, they consider each sense as a vector, whose dimensions are each word in the corresponding cluster and its weight determines its level of co-occurrence with the target word. The sense assignation is done by determining the similarity between sense vectors and a vectorial representation of each target word instance.

Generally, 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 2007, 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 2007] 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 2007] 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], breadth-first 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 2007, 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, re-

spectively) as a final step to disambiguate a target word given its context. Meanwhile, both [Liu 2011, Qian 2014] 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 always find the same clustering result for the same input graph.

Another common clustering approach to automatically induce word senses [Goyal 2014, Song 2016] is spectral clustering [Shi 2000]. Some advantages of this method include that it is simple to implement and often outperforms other clustering methods as well as being able to work directly with graph and non-graph data [Luxburg 2007]. The algorithm consists in projecting the normalized Laplacian of an affinity (or similarity) matrix (adjacency matrix in the case of graph data) on its k first eigenvectors. The lower-space projection allows for an easier separation, for example with k-means, in order to find the group membership of each original data point.

Indeed, the Laplacian of an affinity matrix can be used to represent several properties of the inherent structure within the data. Specifically, in the case of a graph, it tells us about the number of connected components through its set of eigenvalues, specially the eigenvectors associated to the smallest eigenvalues, which synthetically

represent the data.

The normalized Laplacian of an affinity (symmetric and positive) matrix $W \in \mathbb{R}^{n \times n}$, with $w_{ij} = w_{ji} \geqslant 0$, is defined as:

$$\mathcal{L} = I - D^{-1/2} W D - 1/2 \tag{3.1}$$

where I is the identity matrix and D is the degree matrix of W. D is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal. As W may not be an adjacency matrix, we define the degrees of each row in the matrix as: $d_i = \sum_{j=1}^{n} w_{ij}$.

Given a symmetric and positive similarity matrix $W \in \mathbb{R}^{n \times n}$, and a number of desired clusters k, the steps required to perform spectral clustering are:

1. Obtain the normalized Laplacian $\mathcal{L}$ as indicated in Equation 3.1.

2. Obtain the first k eigenvectors $u_{1 \ldots k}$ of $\mathcal{L}$.

3. Store said eigenvectors as columns in a matrix $V \in \mathbb{R}^{n \times k}$. This matrix is akin to a lower-dimension projection of the original similarity matrix W.

4. Cluster the points in V_i with k-means. The clusters found and their members correspond to the cluster of the spectral algorithm.

Special attention must be given to the input matrix W as it must be indeed a symmetric affinity matrix in order to ensure the proper behavior of this method [Luxburg 2007, Goyal 2014].

Supervised Sequential Classification In order to determine the correct set of tags that identify the words within a phrase, sequential NLP tasks such as NER, are usually solved by using supervised structured algorithms. In particular, we focus on the structured perceptron by [Collins 2002] sequence labeling algorithm. The general intuition of this method is introduced below. We chose this method as it is relatively simple and performs as well as more complex probabilistic approaches (e.g., conditional random fields, hidden Markov models) [Daumé III 2006, Daumé III 2012].

Like the well-known perceptron algorithm, the structured perceptron is an online method that learns to classify one example at a time. The goal of this algorithm is to determine a label for each structured input. Formally, for an input $x \in \mathcal{X}$, we want to predict a class $\hat{y} \in \mathcal{Y}$ that maximizes the following product:

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 2007]	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 2014]	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

$$\hat{y} = \arg\max_{\hat{y} \in \mathcal{Y}} w \cdot \Phi(x, \hat{y}) \tag{3.2}$$

where $\Phi(x, \hat{y})$ is the vector of representing the feature space of input x with respect to each of the possible labels $\mathcal{Y} = \{1, \ldots, K\}$. Indeed, $\Phi(x, \hat{y})$ indicates a certain level of compatibility between input x and each label y.

We learn w as a weight vector which considers each feature in regard to each class. As with the classic perceptron, w is updated if the predicted $\hat{y}$ is different to the true label y, as follows:

$$w \leftarrow w + \Phi(x, y) - \Phi(x, \hat{y}) \tag{3.3}$$

The algorithm for the structured perceptron is shown in Algorithm 1. For each input, we obtain its predicted class and determine whether it is equal to the true label. If it is not the case, we update the weight vector.

Algorithm 1: Training phase of the Structured Perceptron

Input: Data $x \in \mathcal{X}$
Input: True labels $y \in \mathcal{Y}$
Input: Max number of iterations $MaxIteration$
Output: A vector of lerned weights w

```
1 for Iteration = 1 ... MaxIterations do
2     foreach (x, y) ∈ X, Y do
3         ŷ = arg max_{ŷ∈Y} w · Φ(x, y)
4         if ŷ ≠ y then
5             w ← w + Φ(x, y) − Φ(x, ŷ)
6         end
7     end
8 end
9 return w
```

The main issue with this algorithm is the computation of $\arg\max_{\hat{y} \in \mathcal{Y}} w \cdot \Phi(x, \hat{y})$ in line 3 of Algorithm 1. As said before, we are looking for the sequence of labels $\hat{y}$ that maximize this product. This implies a very large search over all the possibilities that grows exponentially with the number of possible tags $\mathcal{Y}$. If we consider a phrase with L words, and a tagset of k labels, the total number of possibilities to explore would be

L^k. For example, for a phrase with 10 words, and 5 labels, we would need to explore a space of $10^5 = 100,000$ possibilities. This exploration then tends to be computationally unfeasible for a large number of phrases and/or larger number of words per phrase.

In order to address this problem, another familiar algorithm is employed: the Viterbi decoder. Briefly, this method calculates the optimal set of labels for each input, by determining the maximum path (or sequence of labels) through a lattice of tags' possibilities, one step at a time. As we search the maximum path of labels taking into account only the current word and the last at a time, we reduce the search space to LK^2. Again, for a phrase of 10 words and 5 labels as before, we would be searching over $10 \times 5^2 = 250$ possible sequences of tags.

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. 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 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 2014], our model is indeed based on the use of a hypergraph. Its 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

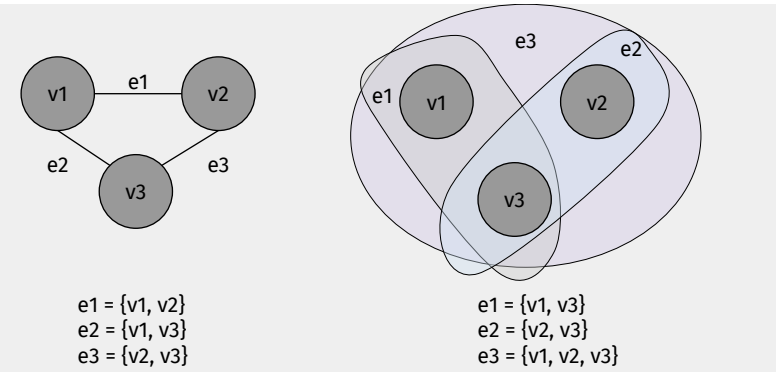


Figure 3.2: A graph (to the left) compared to a hypergraph (to the right). The edges of the hypergraph (hyperedges) may hold more than two nodes at once, relaxing the constraint of graphs' binary relations. A hypergraph can be seen as a set of n-ary sets: $E = \{\{v1, v3\}, \{v2, v3\}, \{v1, v2, v3\}\}$.

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 it 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 give us different types of similarities. Recall that a smaller lexical window gives us co-occurrence relations that approach functional relatedness, such as those found with syntactic contexts. 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 $dep(h, \cdot)$, 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 $dep(h, v)$ 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 $\mathbf{SEN} = \{h_{S_1}, h_{S_2}, h_{S_3}\}$ 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 $dobj_{contains}$.

Hyperedges 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:

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

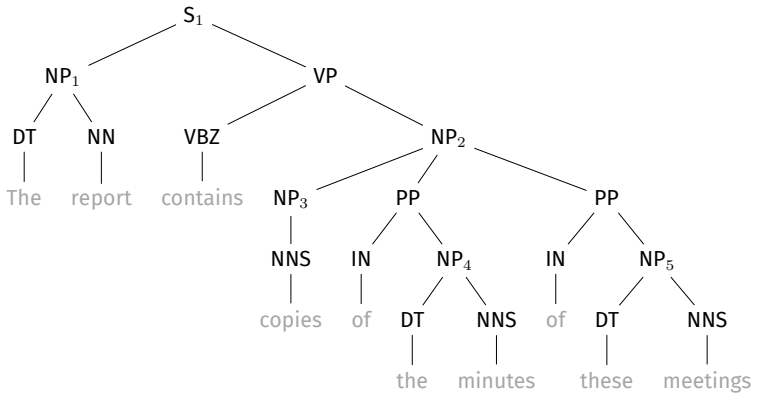


Figure 3.3: Constituency-based tree of the phrase *The report contains copies of the minutes of these meetings.*

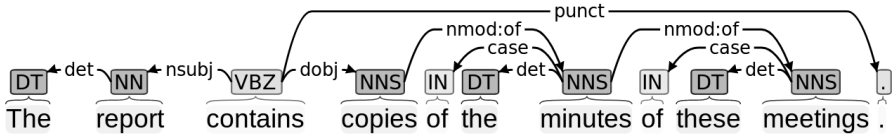


Figure 3.4: Dependency-based tree of the example phrase.

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.

  - $S_1 = \{report, contains, copies, minutes, meetings\}$

- **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.

With the short example presented 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, we realize that the incidence matrix is relatively sparse and this it will inevitably increase as more text is included in the hypergraph. 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

The second part of our proposed method deals with the fusion of textual features. Namely, we combine the features that describe terms into a single representation space. This new space aims to address two issues that arise while working with textual data: effectively using information coming from different linguistic levels (e.g., lexical, syntactic, semantic) while alleviating the sparsity typical of textual representations.

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. A third and fourth type of fusion methods, cross-media fusion and hybrid fusion are also employed in multimedia analysis tasks.

These four fusion operators naturally address the issue of dealing with heterogeneous data as they all mix one way or another the feature columns from each of two representations. Regarding alleviating sparsity, the intuition is that by combining matrices either by summing or element-wise multiplying them, the resulting matrix will have a denser structure. For example, by summing two matrices with the same shape, such as two term-term similarity matrices, we obtain a resulting matrix that contains the similarities of both feature spaces. In the same sense, when multiplying two matrices we combine them while also obtaining a denser output matrix. Nonetheless,

both sum and multiplication result depends evidently on the nature of the matrices employed. Two of the fusion techniques mentioned above, late fusion and cross-media fusion use sum and multiplication as they main matrix operator. What is more, both of them contemplate the use of similarity matrices as at least one of their inputs. Being similarity matrices, they tend to be dense and thus the resulting sum or product will be more dense than the original sparse representation, while complementing and enriching the space with other types of information. We present an example of this intuition in the following section.

We describe the four of them in the following paragraphs. The notation used is first introduced as follows. The fusion functions are binary, they all take two inputs, parameters A and parameter B which define arbitrary single-modality matrices. For example, both matrices A and B may represent a lexical M^L, syntactical based M^S, or other type representation spaces M^T. On the other hand, they may also describe their respective similarity (square) matrices, S^L and S^S. In a broader sense, matrices A and B may represent any pair of valid[3] term-feature matrices.

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 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 is shown in Figure 3.6. In this example, two matrices are used, M^L and M^s. Each one represents 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. Formally, the early fusion function is defined as:

$$E(A, B) = \textbf{hstack}(A, B) \tag{3.4}$$

As stated before, the matrices A and B are combined together via a concatenation function **hstack** which joins both of them column-wise. In order for this operator to work, both matrices must have the same n number of rows.

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

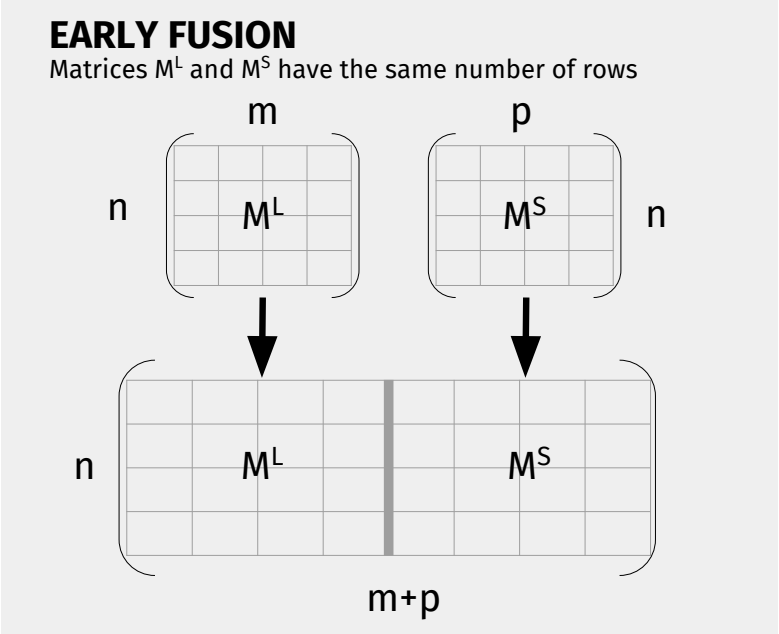


Figure 3.6: Early Fusion of feature matrices A and B. The result is the concatenation of both matrices.

During the concatenation, we may also apply a weight to matrices A and B such as:

$$E_\alpha(A, B) = \textbf{hstack}(\alpha \cdot A, (1 - \alpha) \cdot B) \tag{3.5}$$

Indeed, this 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 simply as early fusion. When α is determined (and indicated as a subscript), we refer to weighted early fusion. Otherwise, there is no weighting scheme applied to the operation.

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 particular type of fusion is very close to machine learning ensemble methods.

Indeed, 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 may combine two modalities by means of their respective similarity matrices. In this case, a representation is obtained by adding two similarity matrices, as in Figure 3.7b. In the figure, we use the equal-sized matrices A′ and B′, which in this case are the similarity matrices of inputs A and B. Both matrices are then summed to produce a single $n \times n$ weighted matrix (determined by a parameter β).

More formally, we define the late fusion function as:

$$L_\beta(A, B) = \beta \cdot A + (1 - \beta) \cdot B \tag{3.6}$$

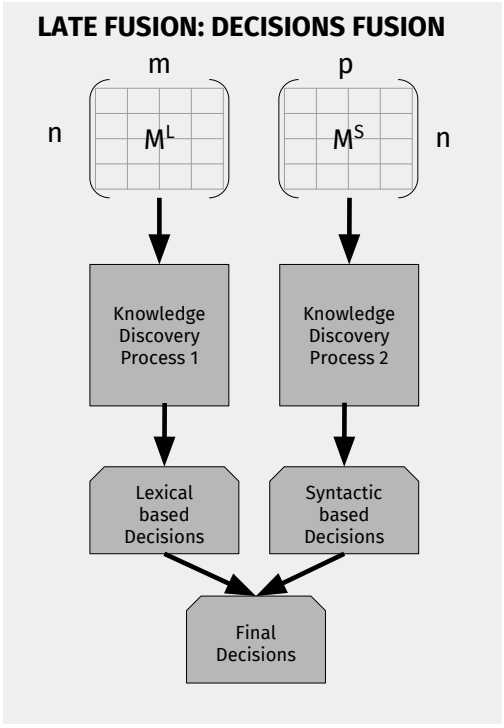
In this function, the extra parameter β affects the influence of matrix A, and consequently also the relevance of matrix B. As we are summing two matrices, for this operator both A and B must be of the same size.

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.

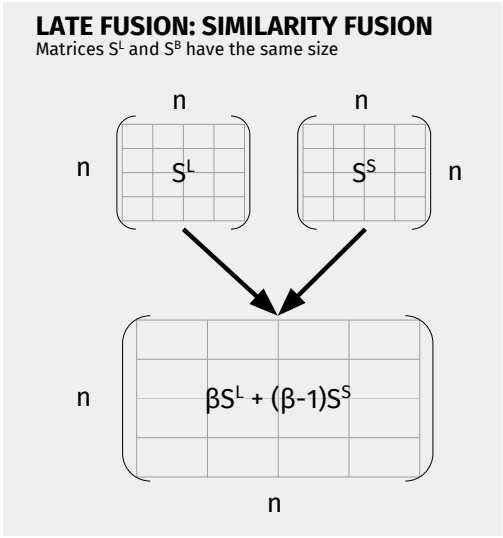
Cross-media Fusion A third type of fusion technique, cross-media fusion (or simply cross fusion), introduced in [Clinchant 2011, Ah-Pine 2015], is defined and employed to propagate a single similarity matrix into a second similarity matrix. In their paper, the authors propagated information from textual 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. Keep only the nearest neighbors for each word within the lexical similarity matrix while assigning zero to the rest. These selected neighbors are to be used as lexical representatives to enrich the syntactical similarities.



(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.

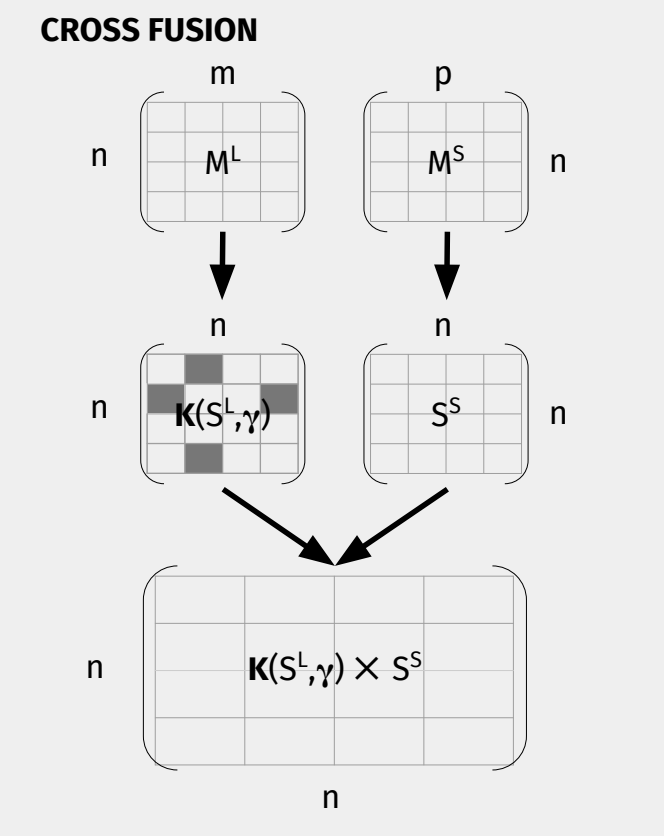


Figure 3.8: Cross fusion for matrices A and B. Similarities are computed, the best similarities are selected for A and finally a product is calculated between the similarities of B and the top similarities of A.

3. Linearly combine both similarity matrices via a matrix product.

Figure 3.8 shows this cross fusion example, between matrices M^L and M^S. First, their corresponding similarity matrices S^L and S^S are first obtained. Secondly, a top-nearest neighbors operation is applied, $\mathbf{K}(S^L, \gamma)$, yielding a matrix which contains only the most representative similarities, determined by the parameter γ. Finally, the cross fusion representation is obtained by computing the product $\mathbf{K}(S^L, \gamma) \times S^S$.

Formally, we define the cross fusion function as:

$$X_\gamma(A, B) = \mathbf{K}(A, \gamma) \times B \tag{3.7}$$

In this case, the $\mathbf{K}(\cdot)$ function takes the top-γ closest words (columns) to each word (lines) while the rest of the values are set to zero. As noted before, matrices A and B may be two similarity matrices. While A is always required to be a square filtered

similarity matrix, B may be also a plain term-feature matrix, as we describe in the following paragraphs. The sole requirement is that the number of columns of the result of the operation $\mathbf{K}(\cdot)$ should be equal to the number of rows of B.

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.

Having said that, here we distinguish three levels of hybrid fusion (aside from the use of single features, or SF henceforth, independently) that we employ in our experiments during the Chapter 4.

1. **First Degree Fusion (1F)**: we consider the three elementary fusion techniques described before (early fusion, late fusion, cross fusion) by themselves. This level of fusion serve as the baseline we set to surpass in order to show the efficacy of the representation feature space found through fusion techniques. As an example, we may obtain a first degree representation matrix by performing an early fusion between the lexical matrix and the syntactical features matrix: $E(M^L, M^S)$.

   We note that we distinguish two types of cross fusion operators: **Cross Feature Fusion** (X_F) and **Cross Similarity Fusion** (X_S). The former combines a similarity matrix with a feature matrix, e.g., $X_F(S^L, M^S)$. The latter joins a similarity matrix with another similarity matrix, for example $X_S(S^S, S^L)$. The intuition behind cross feature fusion X_F is that the rich information from the first input matrix can be transferred directly to a representation without the need of obtaining its similarity matrix beforehand. We denote them *feature* and *similarity* to refer to the fact that the first one uses simply a feature matrix and the second requires some knowledge from the data, in this case the similarity between terms.

2. **Second Degree Fusion (2F)**: in this level we begin with the recombination of the outputs of the previous two levels. Namely, this procedure yields a combination of "second-degree" among fusion methods. Indeed, we introduce four types of second degree fusions employed in the following list. Each one is illustrated with an example:

   (a) **Cross Feature Early Fusion** ($X_F EF$): consists on the cross feature fusion (X_F) of two inputs, a similarity matrix, and the output of an early fusion. For example the operation $X_F(S^T, E(M^L, M^S))$ implies the X_F of the similarity matrix S^T with the early fusion of matrices M^L and M^S.

   (b) **Cross Feature Cross Similarity Fusion** ($X_F X_S F$): entails the cross feature fusion (X_F) of two elements, the output of a cross similarity fusion (X_S), and a term-feature matrix. For example, the operation $X_F(X_S(S^T, S^S), M^T)$ is the cross feature fusion (X_F) of a cross similarity fusion (X_S), the late having similarity matrices S^T and S^S as inputs, and a standard features matrix M^T.

   (c) **Early Cross Feature Fusion** ($EX_F F$): this operation consists on the early fusion of a feature matrix with the output of a cross similarity fusion. As an example, the operation $E(M^T, X_F(S^L, M^T))$ computes the early fusion of matrix M^T with the result of a X_F with S^L and M^T as operands.

   (d) **Late Cross Feature Fusion** ($LX_F F$): this fusion implies the late fusion of a feature matrix with the output of a cross feature fusion. For example, the fusion $L(M^T, X_F(S^T, M^T))$ describes a late fusion between the feature matrix M^T and the cross feature fusion among S^T and M^T.

3. **Higher-Degree Fusion (HF)**: 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 (that is, more than two times) with other second-degree fusion outputs. In this level we test the following two fusion operations:

   (a) **Early Late Cross Feature Fusion** ($ELX_F F$): As an example for this fusion, the operation $E(M^T, L(M^S, X_F(S^T, M^T)))$ implies the combination of three fusion operations. From left to right, first we compute the early fusion (first operation) of matrix M^T, with the result of a late fusion (second operation) between feature matrix M^S and the result of a cross feature fusion, itself having as input matrices S^T and M^T. Indeed, we perform three operations,

an early fusion, a late fusion and a cross feature fusion, thus the name Early Late cross feature fusion of this operator.

(b) **Triple Early Double Late Cross Feature Fusion**[4] ($\mathtt{EEELX_FLX_F}$): althoug it seems complex, this fusion scheme merely consists on the early fusion of the last two operators: $\mathtt{LX_FF}$ and $\mathtt{ELX_FF}$, with another feature matrix. As an example, the operator $\mathtt{E}(M^L, \mathtt{E}(\mathtt{E}(M^T, \mathtt{L}(M^T, \mathtt{X_F}(S^T, M^T))), \mathtt{L}(M^L, \mathtt{X_F}(S^S, M^L))))$ entails the early fusion of matrix M^L with the result of the early fusion of $\mathtt{ELX_FF}$ with $\mathtt{LX_FF}$.

The fusion operators presented (early, late, and cross fusion) are simple and straight-forward. In total, there are three parameters to control: α and β in early and late fusion, respectively. They both control the relevance of each matrix $\mathtt{A}$ and $\mathtt{B}$ in the operation. And γ, controlling how many top similarities are kept in the cross fusion operator.

As we will see in the experiments carried out in the next chapter, it is the aggregation of several of these fusion functions, as hybrid fusion operations, 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. As said before, certain 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.

Before getting into the experimentation details, in order to make our hypergraph linguistic resource concrete, we present the process to obtain such a representation from a raw corpus, namely the English Wikipedia. In other words, we instantiate our model with a Wikipedia-based corpus in order to better understand the characteristics proposed. To get there, we first need a syntactically parsed Wikipedia corpus. In the following section, 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 accompanied by complementary metadata information regarding the meaning of each vertex and hyperedge).

[4] We adopted the *double* and *triple* notation to lighten the explicit name of this fusion operation: Early Early Early Late Cross Feature Late Cross Feature Fusion

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

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 a 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 resource.

The online encyclopedia Wikipedia[5] 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 alongside its corresponding metadata. Nonetheless, the resource is no longer available on the original URL although it may be obtained through Yahoo's Webscope[6]

[5] https://en.wikipedia.org

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

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 pretreating 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[7] and we share our source code[8] 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 maintained, it provides a common set of labels (Universal Dependencies[9]) used by

[7] https://eric.univ-lyon2.fr/~psoriano/SAWD.html
[8] https://github.com/psorianom/SAWD-maker
[9] http://universaldependencies.github.io/docs/

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 3.4.2). 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 sample of the Wikipedia corpus consisting of articles related to Natural Language Processing and Text Mining[10].

In the following sections we describe the steps we undertook to transform a Wikipedia dump into SAEWD, we give a detailed account of the contents of SAEWD and the format in which we stored the parsed information, then we explain the characteristics of our proposed network structure.

3.4.1 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 step of the process.

We begin the construction of the parsed corpus with the Wikipedia dump XML file obtained from the Wikipedia database[11] from early November 2014. This dump contains around 4.7 million article pages[12]. As shown in Figure 3.9, we apply the

[10] Later on during our experiments, we extracted a random sample of 200 thousand articles. We employ the larger corpus in the experiments in the following sections.

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

[12] We kept all articles available in the Wikipedia dump.

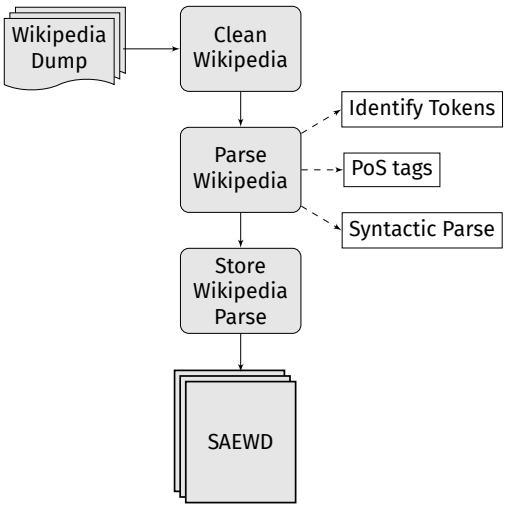


Figure 3.9: The tree steps we took to build SAEWD.

following processing steps in order to yield the final parsed version.

3.4.1.1 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[13]. 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 URL as well as its unique Wikipedia ID within an XML-like label that also doubles as article separator.

[13]We have folders named *AA*, *AB*, *AC* and so on.

3.4.1.2 Parsing Wikipedia

Next, once the Wikipedia dump had been cleaned, we use the Stanford CoreNLP[14] [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 3.4.2.

- **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.
- **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.
- **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.

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 3.4.2. After parsing, we found the statistics shown in Table 3.3.

3.4.2 SAEWD Description

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

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

[14] 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

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 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.

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
	brigand	1				1				1
NN	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.

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`[15] 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.

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 Tab Separated Values file (extension tsv), 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:

  1. FILENAME: indicates the original file used to extract the current parse,
  2. %%#PAGE: denotes a new Wikipedia article, as well as its title,

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

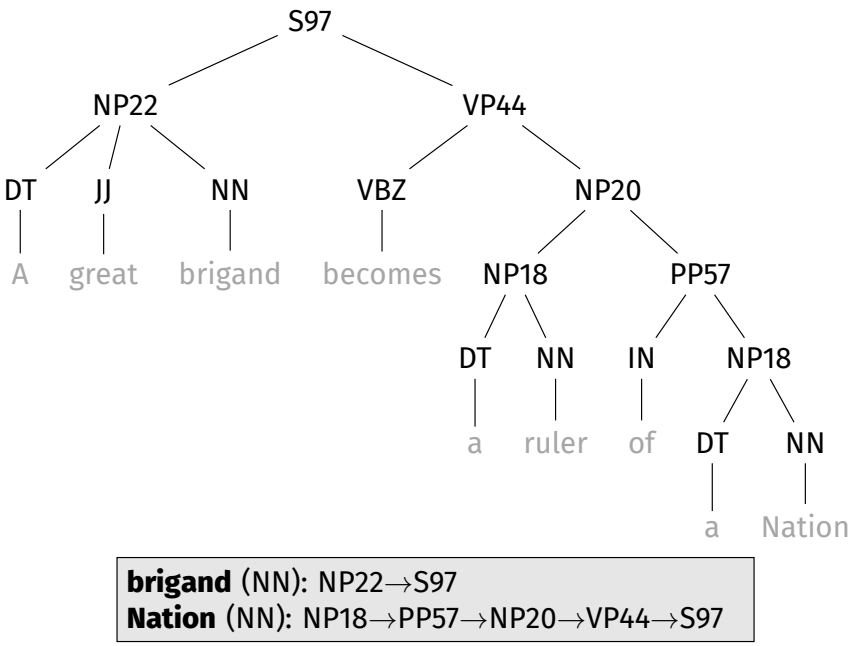


Figure 3.10: Constituency tree for the phrase *A great brigand becomes a ruler of a Nation*. On the bottom, we can see the bottom-up path stored for the words *brigand* and *Nation*.

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 Enriched Wikipedia-based Hypergraph

Once SAEWD is saved to disk, we leverage its information by building a linguistic network by connecting tokens according to their interaction within the Wikipedia corpus. Given the large size of the Wikipedia corpus, we chose a sample of it to illustrate our proposed representation. We randomly selected around 200 thousand articles.

We focus now on the combination of the linguistic features contained in the model to obtain a more diverse, enriched, and less sparse representation. In this subsection, we present a practical example of what are the differences between each the three essential fusion operators (early, late and cross fusion) and with respect to using single features independently. For sake of clarity, we focus on two types of linguistic information: lexical (with a context window of +2-2 around each word) and syntactic (using dependency relations).

The goals of this example are to show how the type of context affects the semantic relatedness of a given word, to get a glimpse of how heterogeneous features get combined into a single enriched representation ideally allowing us to get more knowledge about a given term, and finally, discover how the sparsity is alleviated by combining two different matrices together, especially using the late and cross fusion techniques.

The example consists in presenting the top 5 most similar words of the target word *priest* according to different representation spaces. These representation spaces are obtained using five representation matrices: the lexical features matrix (M^L), the syntactic features matrix (M^S), the early fusion matrix $E(M^S, M^L)$, the late fusion matrix ($L_{0.5}(S^S, S^L)$), and finally two cross fusion matrices ($X_F(S^S, M^L)$ and $X_F(S^L, M^S)$). We report the sparsity level of each matrix (percentage of zero values in the matrix) obtained.

The procedure to obtain said similar words consist in calculating a cosine-similarity matrix for each of the five fusion representations. In some cases, as in late fusion and cross fusion, this step is implicit as in this example, these operators already require the computation of similarity matrices (see Equations 3.6 and 3.7).

The most similar terms to the target word can be seen in Table 3.6. We note that in this example we are not interested in determining the quality of the semantic

related words discovered with each representation space. Even more, it seems hard to determine the semantic-relatedness quality of these similar (*similar* in the sense of cosine similarity) words. Still, we can say that, as expected, the words seem to be semantically related in a large sense.

As discussed by [Levy 2014a], lexical features seem to give words that are semantically related in a larger sense, in this example, religion related terms. On the other hand, dependency based relations similarities tend to discover functional words or words that are of the same *semantic type*. With respect to early and late fusion, while the similar words found are already known, we discover new terms that were until now unknown which seem to be semantically related, such as *relic* and *seer*. Concerning the cross fusion, in this case cross feature fusion, both transferring from the syntactic to the lexical similarity matrix and the other way around, we see that we also found previously seen words while discovering yet another couple of related words: *monk* and *chorus*. It is also clear how by selecting to transfer information from the syntactic matrix (fifth column) we get functionally related terms (occupations in the church or in a power structure) while on the sixth column (transfer from lexical to syntactic), we get mostly words that correspond to a broader similarity domain.

In Table 3.7 we present again the top 5 similar terms to the word *priest*. This time using similarity matrices as representation spaces. The overall behavior described above regarding the nature of the semantic relations is also kept in this representation spaces.

With regard to alleviating the data sparsity (indicated below the header of each column as the percentage of non-zero values in the matrix) it is quite obvious that by using a similarity matrix we densify the space by means of a matrix multiplication, which is the case of both cross fusion operators (columns four and five of Table 3.6): we pass from a sparsity of 5.49% in the lexical matrix and 4.97% in the syntactic matrix to 16.75% and 13.45% in the cross fusion matrices, respectively. Furthermore, for the cross feature fusions X_F, while we also employ a similarity matrix, we stay in the same space (same number of dimensions) of the feature matrix, while more than doubling the density of the space at the same time.

On the other hand, the same reduction of sparsity is achieved while using similarity matrices, (in Table 3.7). Originally having 75.25% and 60.64% sparsity using the lexical and syntactic similarity matrices respectively, we get to a maximum of 87.22% using cross similarity fusion going from syntactic to lexical information $X_S(S^S, S^L)$.

	Lexical Features (5.49%) M^L	**Syntactic Features (4.97%)** M^S	**Early Fusion (5.23%)** $E(M^L, M^S)$	X_F **Fusion (16.75%)** $X_F(S^S, M^L)$	X_F **Fusion (13.45%)** $X_F(S^L, M^S)$
priest	priests	monk	sailor	vassal	sailor
	nun	regent	regent	regent	fluent
	canton	aedile	nuclei	nun	dean
	sailor	seer	nun	sailor	nuclei
	burial	meek	relic	monk	chorus

Table 3.6: Target word *priest* and its top 5 most similar words using different representation matrices. The sparsity level (percentage of non-zero values) of each representation is shown below the header of each column.

	Lexical Similarity (75.25%) S^L	**Syntactic Similarity (60.64%)** S^S	**Early Fusion (67.94%)** $E(S^L, S^S)$	**Late Fusion (83.17%)** $L(S^L, S^S)$	X_S **Fusion (87.22%)** $X_S(S^S, S^L)$	X_S **Fusion (79.69%)** $X_S(S^L, S^S)$
priest	wholly	regent	regent	regent	regent	sailor
	burial	coach	slang	slang	vassal	nuclei
	monk	broker	broker	seer	vizier	nun
	lingua	dream	rebel	tutor	leader	canton
	nuclei	tailor	tiger	cradle	result	burial

Table 3.7: Target word *priest* and its top 5 most similar words using different representation similarity matrices. The sparsity level (percentage of non-zero values) of each representation is shown below the header of each column.

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 fusion enriched hypergraph linguistic model that is able to hold heterogeneous language information while allowing its combination and alleviating the data sparsity. This structure allows the integration multiple kinds of information and has potential in terms of which algorithms it can be used with. The three levels of contexts 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.

Chapter 4

Applications to named entity recognition and word sense disambiguation

Abstract. *This chapter presents the experiments we performed as applications of the proposed model. On the first subsection, we use well-known methods to solve named entity recognition while using fusion enriched representation spaces. We show that these kinds of representations, leveraging heterogeneous information and alleviating its data sparsity, are useful to improve the performance of the task. Indeed, our results on three different datasets using enriched representations are better than those of the baselines we propose, and more importantly, our results show that the combination of textual features indeed improves the performance compared to single feature and the trivial feature concatenation. We also give a detailed analysis into how the fusion operations get to improve the performance of the task at hand.*

In the second subsection, we change the NLP task to word sense induction and disambiguation. First, we apply the same fusion operations as before to solve the task using an existing literature approach. Our experiments on two different corpora show that the improvements shown in named entity recognition are consistent in word sense induction and disambiguation. Secondly, we propose a method to exploit the structure of the network within our linguistic model. Although this approach has been studied before, we improve over the previous literature results while having a reduced number of parameters and employing heterogeneous features to solve the task. We also analyze the results obtained according to the type of word studied, whether nouns or verbs, and according to the effect of the use of either lexical or syntactical information.

Contents

4.1 Introduction

In this applications chapter we set to solve two natural language processing tasks using as data source corpora in the form of the model described in Chapter 3. We address the tasks of Named Entity Recognition (NER) and Word Sense Induction and Disambiguation (WSI/WSD). Both tasks are located on the semantics sub-domain of NLP.

These experiments represent the third and final contribution of this thesis, after introducing the theoretical fusion enriched model in the previous chapter. Indeed, this contribution is the continuation of our set of propositions, as shown in Figure 4.1. We employ both a fusion enriched and a raw hypergraph network based on benchmark corpora to validate the utility of our proposals.

The general objectives of the experiments described below are two: (1) to test the effectiveness of using fusion enriched representations to solve NLP tasks, while combining heterogeneous information and densifying the feature space; and (2), to leverage the structure of a network built using the hypergraph structure described before.

There are two main parts in this chapter. First we address NER, we study how the different types of fusion operations affect the performance of the task. We train well-known classification algorithms with representations obtained from fusion operations. According to our results, we find that it is indeed interesting to combine different types of features into a single representation space. We also delve into a result's analysis to try to understand the reason behind the improvement using fusion techniques.

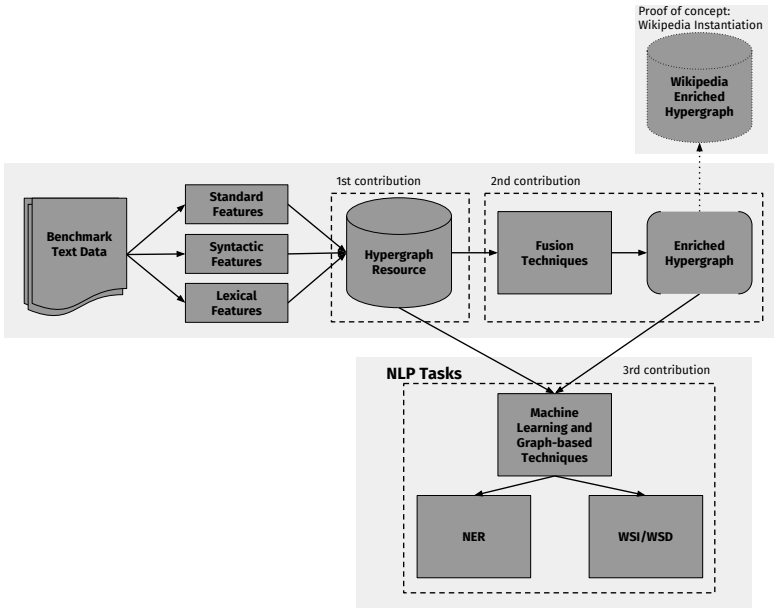


Figure 4.1: Complete description of the three contributions of this work. In this chapter we focus on WSD/WSI and NER as applications of the model proposed.

The second part deals with WSI/WSD. This subsection is divided in two segments. First, once we determined that combining features is interesting for NER, we want to verify that this combined representations are indeed useful for other NLP tasks, such as WSI/WSD. As before, we use a literature learning technique with the representation obtained by the combination of available features. Our results show that the fusion enrichment is useful also to solve WSI/WSD.

Secondly, we leverage the structure within the hypergraph resource to solve WSI/WSD. Briefly, the goal is to detect important words, according to their connections with other words, and extract their neighborhood to determine possible senses within the network. While this method has been used before, we improve the performance and reduce the number of parameters employed by those literature methods. We also study the effect of using different types of linguistic information, namely lexical and syntactic contexts.

At the end of the chapter we conclude by discussing the experiments performed, their limitations and future improvements and further research.

4.2 First Application: Named Entity Recognition

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, within a text, entities of type Location (LOC), Organization (ORG), Person (PER), Miscellaneous (MISC), or if the term is not an even an entity, assigning them a (O) label. The task is of great importance for more complex NLP systems, e.g, relation extraction, opinion mining [Nadeau 2007].

Generally, two common solutions to NER involve the use of matching patterns, created manually or extracted in a semi-automatically fashion[Gupta 2015]; or more popularly, by training a supervised machine learning algorithm with large quantities of annotated text [Aggarwal 2012] . The latter being the currently more popular solution to this task. As is usual with other NLP tasks, NER requires textual features to represent words in order to determine their role within a phrase. We propose to build representations based on our fusion enriched hypergraph model.

Usually, representations employed for NER are obtained from the surrounding context of the words in the input corpus. Mainly, two types of representations are used: lexical and syntactic. As we know, the first type requires no extra information than that contained already in the analyzed text itself. The second type, syntactic features are based on part of speech tags, phrase constituents information, and syntactical functionality between words, the later portrayed by syntactical dependencies. Likewise, there are specific features that are particular to one task are also be employed.

The main intuition of these experiments is that word similarities may be found at different levels according to the type of features employed. In order to exploit these similarities, we leverage our fusion enriched framework. Specifically, in our experiments, we try to mutually complement independent representations by utilizing said fusion techniques to generate a single feature space that improves the performance of NER, specially compared to the using features independently and the trivial feature concatenation (early fusion). Consequently, the main goal is to assess the effectiveness of simple, yet untested fusion techniques and their combination.

We consider the first three types of fusion techniques described in subsection 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

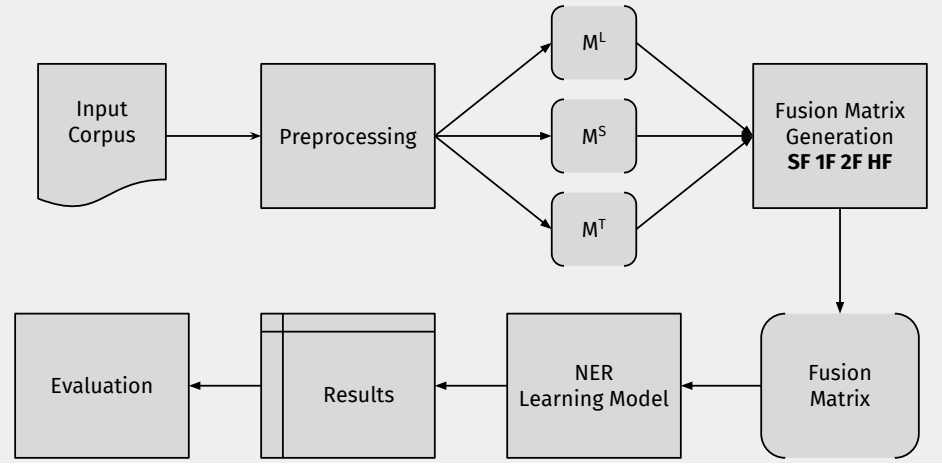


Figure 4.2: Steps followed on our fusion-enriched representations for NER experiments. First the corpus is preprocessed, then features are extracted from the text. A fusion matrix is generated, which in turn is used as input to a learning algorithm. Finally, the system yields its results and to be analyzed.

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. We also find new combinations empirically. Nonetheless, as we will show, their effectiveness replicates to different datasets and NLP tasks.

The specific methods used to solve both NER and WSI/WSD (in the following subsection) are already known to be robust. Our work focus specially on the generation of new representation spaces.

4.2.1 Fusion Enriched Representations

Our first goal in this subsection 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 through experimentation. Nonetheless, as we will show, their effectiveness replicates to different datasets and NLP tasks.

The experiments we carry on consist in generating fusion matrices that will serve as input to a learning algorithm in order to solve NER. These input feature matrices are based upon lexical, syntactical, or other NER standard types of representation. The procedure can be sen in Figure 4.2.

Representation Spaces In Chapter 3 we presented the fusion operators to be used in our experiments. Below we detail the three types of features matrices used to generate the fusion-enriched combinations that describe the words of the corpus tested.

Lexical Matrix (L) 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 $(w_{-2}, w_{-1}, w, w_{+1}, w_{+2})$. This type of context features is typical for general systems studying the surroundings of a word and in particular for the named entity recognition task [Daumé III 2006, Nothman 2009, Ratinov 2009]. We retake the example phrase from [Levy 2014a], the lexical-based features of the phrase *Australian scientist discovers start with telescope*, are shown in Table 4.1.

Word	Features
Australian	word:Australian, word+1:scientist, word+2:discovers
scientist	word-1:Australian, word:scientist, word+1:discovers, word+2:star
discovers	word-2:Australian, word-1:scientist, ..., word+2:telescope
star	word-2:scientist, word-1:discovers, word:star, ..., word+2:telescope
with	word-2:discovers, word-1:star, word:with, word+1:telescope
telescope	word-2:star, word-1:with, word:telescope

Table 4.1: Lexical features corresponding to the phrase *Australian scientist discovers start with telescope*.

Syntactical Matrix (S) 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 $m_1, \ldots, m_k$ and their corresponding PoS tags $p_{m_1}, \ldots, p_{m_k}$; a head h and its corresponding PoS tag p_h, we consider the context features $(m_1, p_{m_1}, lbl_1), \ldots, (m_k, p_{m_k}, lbl_k), (h, p_h, lbl\_inv_h)$. 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.

For the phrase *Australian scientist discovers start with telescope* the dependency-based context is shown in Table 4.2.

Word	Contexts
Australian	scientist/NN/amod_inv
scientist	Australian/JJ/amod, discovers/VBZ/nsubj_inv
discovers	scientist/NN/nsubj, star/NN/dobj, telescope/NN/nmod:with
star	discovers/VBZ/dobj_inv
telescope	discovers/VBZ/nmod:with_inv

Table 4.2: Syntactic contexts corresponding to the phrase *Australian scientist discovers start with telescope*.

NER Standard Features Matrix (T) The features used for NER are based roughly on the same as those used in [Daumé III 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 (also 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.

Learning Methods NER being a supervised learning task, we use an averaged structured perceptron [Collins 2002, Daumé III 2006] (see Section 3.2.1) to determine the tags of the named entities. We considered logistic regression and linear SVM. For the main experiments, we chose the perceptron because of its performance and the lower training time. On the other hand, for the analysis of the results, we use a logistic regression as it is considerably easier to interpret its results, keeping in mind that our goal is to give some insights regarding the usefulness of our fusion methods.

4.2.2 Experiments and Evaluation

We experiment with the four levels of fusion discussed before: Single Features (SF), First-degree Fusion (1F), Second-degree Fusion (2F) and Higher-degree Fusion (HF). The representation matrices for NER come from lexical context features M^L, syntactical context features M^S or standard features M^T. On the other hand, experiments on WSI/WSD exclusively employ matrices M^L and M^S.

We recall that our first goal is to compare the efficiency of the primary fusion techniques applied to named entity recognition. Then, we empirically determine a

fusion combination operator able to leverage the complementarity of the features used.

To this end, we evaluate the aforementioned 4 fusion levels. We note that the fusion combinations in the third and fourth level (2F and HF) 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 fusion combinations that can be tested.

Preprocessing 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 is again extracted with the Stanford's CoreNLP parser. We recall that the features used for these experiments on NER are those described before: lexical, syntactic and standard features, i.e., M^L, M^S, and M^T, 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 [Nothman 2009] of selected English Wikipedia articles, all of them annotated automatically with the author's semi-supervised method. In total, it contains 3.5 million words from an unspecified number of articles.

(3) Wikigold (WGLD): Also a corpus of Wikipedia articles [Balasuriya 2009]. Nonetheless, this one was annotated manually. This dataset is the smaller, using 149 articles and 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. Otherwise, while it is faster to train models with this corpus, it may be the case that they are not able to properly fit the data given its size, and thus performance is lower than the other datasets.

The three of these datasets employ the BIO text segment tagging schemes. This tag set suggests that a word is in the **B**eginning, **I**nside, or **O**utside of a named entity. Indeed, given that there are four categories, person (PER), location (LOC), organization (ORG) and miscellaneous (MISC), there are indeed 9 different classes (B and I for each category plus O).

Evaluation Measures We evaluate our NER models following the standard CoNLL-2003 evaluation script. Given the large amount of experiments we carried out and to reduce the number of reported results, 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.

4.2.3 Results and Discussion

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 $L_\beta(A, B) = \beta \cdot A + (1 - \beta) \cdot B$, to be $\beta = 0.5$. 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 4.3: NER F-measure results using the Single Features over the three datasets. These values serve as a first set of baselines. Results are obtained with the structured perceptron algorithm.

A	Single Features		
	CONLL	**WNER**	**WGLD**
M^T	77.41	77.50	59.66
M^L	69.40	69.17	52.34
M^S	32.95	28.47	25.49

Single Features Looking at Table 4.3, we see that the best single features (SF), in terms of F-measure come from the standard representation matrix M^T. 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^L performs relatively well, considering it only includes information contained in the dataset itself. Nevertheless, this kind of representation is the foundation of most word embedding techniques used nowadays. While we expected better results from the syntactical features M^S, 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 4.4 we present the first degree fusion level (1F). 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 feature and similarity 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^T 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 4.4: NER F-measure results using first degree fusion (1F). Operators in column B are either indicated on the table or specified as follows. In $X_F F$, depending on the dataset tested, $b^*_{X_F F}$ takes the matrix from the set $\{M^L, M^T\}$ which yields the best performing result. In $X_S F$, $\hat{b}^*_{X_S F}$ corresponds to the best performing matrix in $\{S^L, S^S\}$. These configurations serve as the main set of baseline results. Results are obtained with the structured perceptron algorithm.

A	B	**Early Fusion (EF)**		
		CONLL	**WNER**	**WGLD**
M^L	M^S	72.01	70.59	59.38
M^L	M^T	78.13	79.78	61.96
M^S	M^T	77.70	78.10	60.93
M^L	$E(M^S, M^T)$	**78.90**	**80.04**	**63.20**
		Late Fusion (LF)		
		CONLL	**WNER**	**WGLD**
S^L	S^S	**61.65**	58.79	44.29
S^L	S^T	55.64	**67.70**	48.00
S^S	S^T	50.21	58.41	**49.81**
		Cross Feature Fusion ($X_F F$)		
		CONLL	**WNER**	**WGLD**
S^L	M^T	49.90	**70.27**	**62.69**
S^S	M^T	47.27	51.38	48.53
S^T	$b^*_{X_F F}$	**52.89**	62.21	50.15
		Cross Similarity Fusion ($X_S F$)		
		CONLL	**WNER**	**WGLD**
S^L	S^T	27.75	**59.12**	38.35
S^S	$b^*_{X_S F}$	36.87	40.92	39.62
S^T	$b^*_{X_S F}$	**41.89**	52.03	**39.92**

Second Degree Fusion The second degree fusion techniques (2F) presented in Table 4.5 show that the recombination of cross fusion techniques gets us closer to the early

Table 4.5: NER F-measure results using second degree fusion (2F) operations. In $X_F X_S F$, $\hat{a}$ corresponds to the best performing matrix in the set $\{X_S(S^T, S^L), X_S(S^L, S^T), X_S(S^T, S^S)\}$. In $EX_F F$, depending on the dataset, b^*_{EXEF} takes the best performing matrix from $\{X_F(S^S, M^L), X_F(S^L, M^L), X_F(S^L, M^T), X_F(S^S, M^L), X_F(S^S, M^T)\}$. Finally, in $LX_F F$, $\hat{b}_{LX_F F}$ takes the best possible matrix from $\{X_F(S^L, M^T), X_F(S^S, M^T), X_F(S^S, M^L)\}$. Results are obtained with the structured perceptron algorithm.

A	B	**Cross Feature Cross Similarity Fusion ($X_F X_S F$)**		
		CONLL	**WNER**	**WGLD**
$\hat{a}$	M^T	37.69	59.44	**41.71**
$\hat{a}$	M^L	**38.31**	**58.73**	41.56
$\hat{a}$	M^S	29.31	52.06	34.91
				Cross Feature Early Fusion ($X_F EF$)
		CONLL	**WNER**	**WGLD**
S^T	$E(M^L, M^T)$	**54.34**	**64.20**	39.59
S^L	$E(M^L, M^T)$	49.71	71.84	**45.14**
S^S	$E(M^L, M^T)$	47.54	53.77	43.32
				Early Cross Feature Fusion ($EX_F F$)
		CONLL	**WNER**	**WGLD**
M^T	$b^*_{EX_F F}$	49.58	**77.32**	**61.69**
M^L	$b^*_{EX_F F}$	49.79	66.22	53.54
M^S	$b^*_{EX_F F}$	**51.53**	70.94	53.70
				Late Cross Feature Fusion ($LX_F F$)
		CONLL	**WNER**	**WGLD**
M^T	$\hat{b}_{LX_F F}$	54.82	**75.70**	**54.73**
M^L	$\hat{b}_{LX_F F}$	**56.53**	62.27	52.39

fusion baseline. Except for cross feature cross similarity fusion ($X_F X_S F$), the rest of the recombination schemes yield interesting results. First, in cross feature fusion, the best results, for the most part, are obtained while using the S^L matrix combined with the output of $E(M^L, M^T)$, which is still far from the baseline values. Concerning, EXEF, we

Table 4.6: F-measure results using high degree fusion (HF) operators. In $\text{EEELX}_\text{F}\text{LX}_\text{F}$, $\hat{b}_{\text{EEELX}_\text{F}\text{LX}_\text{F}} = E(E(M^T, L(M^L, X_F(S^s, M^L))), L(M^L, X_F(S^T, M^L)))$ for CONLL and $\hat{b}_{\text{EEELX}_\text{F}\text{LX}_\text{F}} = E(E(M^T, L(M^T, X_F(S^s, M^T))), L(M^L, X_F(S^s, M^L)))$ for WNER and WGLD. The best result is obtained in $\text{EEELX}_\text{F}\text{LX}_\text{F}$ when $\alpha = 0.95$. If α is not indicated there is no weighting on EF. Results are obtained with the structured perceptron algorithm.

A	B	**Early Late Cross Feature Fusion (ELX_FF)**		
		CONLL	**WNER**	**WGLD**
M^T	$L(M^L, X_F(S^s, M^L))$	67.16	79.45	62.37
		Triple Early Double Late Cross Feature Fusion ($\text{EEELX}_\text{F}\text{LX}_\text{F}$)		
		CONLL	**WNER**	**WGLD**
M^L	$\hat{b}_{\text{EEELX}_\text{F}\text{LX}_\text{F}}$	65.01	78.02	62.34
$M^L_{\alpha=0.95}$	$\hat{b}_{\text{EEELX}_\text{F}\text{LX}_\text{F}}$	**79.67**	**81.79**	**67.05**
EF Baseline		78.90	80.04	63.20

get already close to surpass the baselines with the M^T matrix, except for the CONLL dataset. In LXEF, even though the cross fusion $X_F(S^s, M^L)$ is not the best performing, we found experimentally that by combining it with M^L through a late fusion, it gets a strong complementary representation. Our intuition in this case was to complement M^L with itself but enriched with the S^s information. In the following high degree fusion results we discover that indeed this propagation of information helps us beat the baselines we set before.

High Degree Fusion Finally, the last set of experiments are shown in Table 4.6. Using a recombination of high degree fusion operations (HF), 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^L 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 4.4) baseline being 78.13, we reached 78.69, the

lowest improvement of the three datasets. Regarding the Wikipedia corpus, in WNER, we passed from 79.78 to 81.75; and in WGLD, from 61.96 to 67.29, the largest improvement of all. It is important that we tried the weighted Early Fusion operator with different α and the best result does not beat these fusion results.

4.2.4 Fusion Analysis

In this subsection we present an analysis on the results obtained with the combination fusion operators shown above. Namely, we want to understand how each addition of fusion operators helps to improve the result of the NER task. For simplicity, we focus on the most successful fusion combination found for the three tested corpora. While the procedure to build the models analyzed herein is the same as before, we do have certain dissimilarities due to the need to explain said models in an effective way. Namely, there are two important changes in the methodology presented before: (1) we focus exclusively on the Wikigold corpus, and more importantly (2), we change the learning method from a structured perceptron to a multinomial logistic regression with L1 regularization. The main reason is that the regression is somewhat easier to interpret as it fits a sparse vector of weights for each feature and for each possible class. While the structured perceptron, also fits a matrix of feature weights, its interpretation is complex as these weights are then used to decode the best combination of tags given a complete phrase, considering the preceding and following words for each term in the corpus. In other words, in the logistic regression we can explain each word prediction independently based on a sparse vector of fitted weights for each feature and the vector that represent the word itself. On the other hand, while using the structured perceptron, we need to look at whole phrases while considering precedent and subsequent words at each time, making the interpretation quite complex. We note that the performance is considerably lower using the logistic regression. Still, using the logistic regression also yields a sequential performance improvement by using enriched feature spaces, similarly to those experiments shown in the previous results tables (results with the structured perceptron).

The most performing fusion combination found during the previous experiments is reported in the second to last line in Table 4.6. We will use this fusion operator to investigate the characteristics of the feature space, which yields improved results. We note that, experimentally, this operator gave the best results for both the structured perceptron and the logistic regression learning methods (as can be seen in Table 4.7).

This operator is fully expressed as:

$$E_{\alpha=0.95}(M^L, E(E(M^T, L(M^T, X_F(S^S, M^T))), L(M^L, X_F(S^S, M^L)))) \tag{4.1}$$

This fusion is principally based on the early fusion operator. It is important to notice that only the left most fusion operator is weighted, that is, its first input is the only one affected by the weight $\alpha = 0.95$. The rest of the early fusions in the operator are non-weighted, i.e., no scaling is applied to their operands. Still, as they are second operator of the first weighted early fusion, they are implicitly affected by a weight of $(1 - \alpha) = 0.05$.

For the sake of clarity in the presentation of the operator in Equation 4.1, and while we defined early fusion as a binary function (in Chapter 3), we will express it below as a n-ary function which concatenates all the input values into a single representation. Again, we note that the parameter α applies exclusively too the first operand of the first and left most early fusion operation. Nevertheless, we include the implicit weights that affect each of the arguments of each function in the description below. Thus, we identify four main operations in equation 4.1:

$$\overbrace{\overbrace{E_{\alpha=0.95}(\underbrace{M^L}_{①}, M^T}^{②}, L(M^T, X_F(S^S, M^T)), L(M^L, X_F(S^S, M^L)))}^{③}}^{④} \tag{4.2}$$

Explicitly, these numbered operations are below. We associate to each operations a model, which is trained using the representation obtained with the corresponding fusion operation.

① M^L used to train model M_1.

② $E_{\alpha_1=0.95,\alpha_2=0.05}(\alpha_1 M^L, \alpha_2 M^T)$ used to train model M_2.

③ $E_{\alpha_1=0.95,\alpha_2=\alpha_3=0.05}(\alpha_1 M^L, \alpha_2 M^T, \alpha_3 L(M^T, X_F(S^S, M^T)))$ used to train model M_3.

④ $E_{\alpha_1=0.95,\alpha_2=\alpha_3=\alpha_4=0.05}(\alpha_1 M^L, \alpha_2 M^T, \alpha_3 L(M^T, X_F(S^S, M^T)), \alpha_4 L(M^L, X_F(S^S, M^L)))$ used to train model M_4.

As can be seen, the operation in Equation 4.2 is a concatenation of four elements, the feature matrices M^L and M^T, and two late fusions, each one containing a cross

feature fusion ($X_F F$). The analysis we make tries to elucidate the role of each numbered fusion combination. To this end, we analyze the models M_1 to M_4 and their corresponding predictions given to certain word instances.

Per-Entity Performance Gain First, we are interested into discovering what is the contribution of each model to the F-measure metric overall and for each specific type of named entity. In that sense, Table 4.7 identifies the gains in performance due to the incremental addition of fusion operations. In the first line we see the results using the M_1. As said before, the results are lower than those obtained with the structured perceptron. On the second line, it is shown that the increment in F-measure (shown in parentheses) for all classes obtained by using M_2 is considerable and in fact the largest (17.50) of them all. Also, while all the classes improve, the most important gain is obtained for the class PER (person), shown in bold letters. In the same sense, on the third line, for model M_3, the best improvement is found for the class ORG (organization). Finally, the last mode M_4, improves LOC (location) class among the rest of the classes.

Table 4.7: Results and improvements between four multinomial linear regression (L1 normalization) models. The performance (in F-measure) is lower than before but the improvement trend with more fusion enrichment is kept. Results are obtained with the logistic regression algorithm.

	NER Tags				
Model	**All Tags**	**LOC**	**MISC**	**ORG**	**PER**
M_1	38.03	49.02	30.24	27.49	41.52
M_2	55.53 (17.50)	65.04 (16.02)	40.03 (9.79)	39.46 (11.97)	**69.19 (27.67)**
M_3	56.11 (0.58)	65.75 (0.71)	40.26 (0.23)	**41.13 (1.67)**	68.99 (-0.20)
M_4	56.28 (0.17)	**66.08 (0.33)**	40.49 (0.23)	41.07 (-0.06)	**69.31 (0.32)**

In summary, the second element of the fusion operator we analyze improves on the PER class, the third on class ORG and the fourth and last on class LOC. This knowledge allows us to frame more easily our next analysis. In the following, we are interested in determining which are the features that most likely make each model take a decision towards one class or another. To that end, we look at three different words that were wrongly classified in a first model and correctly categorized in the

next fusion enriched model. We study words whose correct tags match the tag of the enriched model with the best improvement (see Table 4.7). For example, we are interested in the word *Kory*, which is wrongly classified by model M_1 (it is assigned a tag O) but it is correctly classified as PER in model M_2, since PER is the class with the largest improvement in regard to M_1.

To determine which features are the most relevant, we look into the words non-zero-valued feature columns and match them to the logistic regression coefficients' vectors (corresponding to the model's fitted decision function). In this way we can infer which features contribute or deter the model from selecting a given class according to whether these values are negative or positive. The words we study are:

- *Kory*: wrongly classified as O (out of an named entity) by M_1 and correctly classified as PER by M_2.
- *A-League*: wrongly classified as O by M_2 and correctly classified as ORG by M_3.
- *Green*: wrongly classified as ORG by M_3 and correctly classified as LOC by M_4.

In what follows we are interested in determining which features help to determine the correct classification of the words discussed.

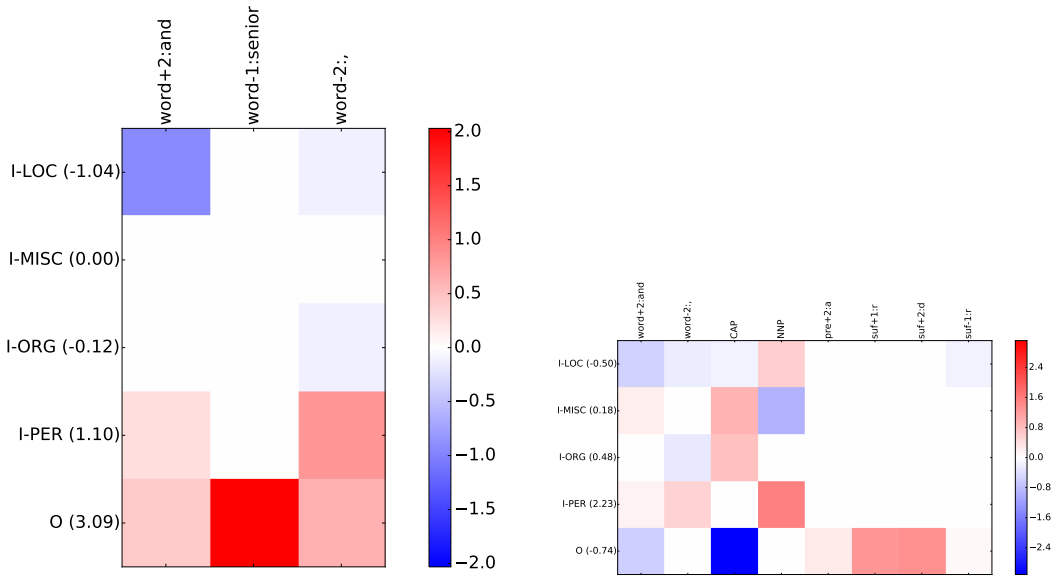
Per-model Feature Importance In Figures 4.3, 4.4, and 4.5 we present six heatmaps showing the features that contribute and prevent words from being classified as one of the five tags available according (broadly) to the weights fitted for each feature during training. Specifically, there is a line for each possible class and a column for each feature that has a non-zero fitted coefficient and a non-zero value on the representation space of its corresponding word. In parentheses, next to the classes, we see the product of the feature vector of the studied word times the coefficients' matrix of the corresponding model. These values serve as an indicator[1] of the class predicted by the model. Color wise, white indicates zero values, red indicates positive values and blue represent negative values. The color intensity is directly associated with the absolute value of the coefficient.

From M_1 to M_2 In the analysis from model M_1 to model M_2, we consider the word Kory.

[1]Indeed, these values are used to obtain the probability of each class by applying to them a logistic function, namely a softmax function.

In model M_1 (Figure 4.3a) we see it is wrongly classified as not a named entity, or tag O, as the feature *word-1:senior*, i.e., the previous word was *senior*, drives this decision. The tag PER is also supported by the features *word+2:and* and *word-2:,* but it is not enough to drive the decision towards it considering the features' coefficients of the O class.

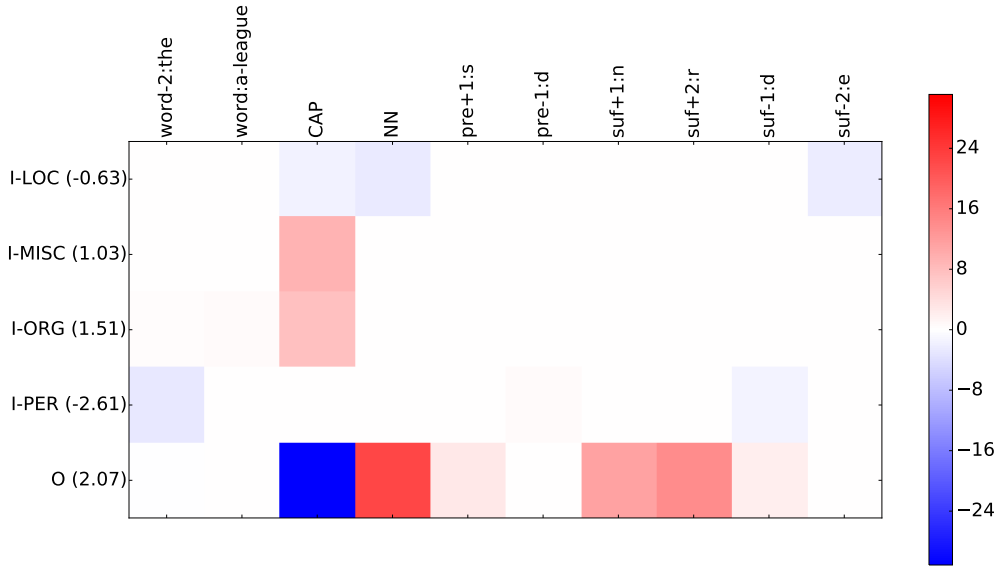
On the other hand, regarding the model M_2 (Figure 4.3b), *Kory* is correctly classified as person, since we added features from the M^T representation space. Specifically, its PoS tag (NNP), contributes towards the selection of PER by the regression decision function. At the same time, the feature CAP which determines whether the word is in upper case or not, contributes against the word being tagged as O.



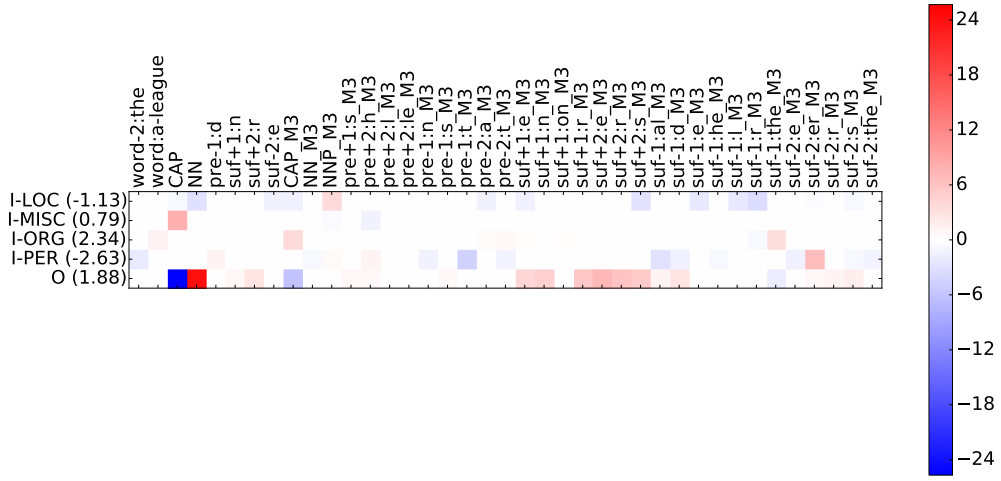
(a) The word *Kory* is predicted to be class O by model M_1 given the coefficients of the features indicated.

(b) The same word is correctly predicted to be class PER by model M_2 given the new features in the model.

Figure 4.3: Non-zero coefficients heatmaps for models M_1 and M_2 corresponding to the word *Kory*. On the left, M_1 fitted using the feature matrix M^L. On the right, model M_2 trained using the term $E_{\alpha=0.95}(M^L, M^T)$. Red colors are positive, blue are negative. The color intensity varies according to the magnitude of the value.



(a) The word *A-League* is predicted to be class O by model M_2 given the coefficients of the features indicated.



(b) The same word is correctly predicted to be class ORG by model M_3 given the new features in the model.

Figure 4.4: Non-zero coefficients heatmaps for models M_2 and M_3 corresponding to the word *A-League*. On top, M_2 fitted using the term $E_{\alpha=0.95}(M^L, M^T)$. On the bottom, model M_2 trained using the $E_{\alpha=0.95}(M^L, M^T, L(M^T, X_F(S^S, M^T)))$ combination. Red colors are positive, blue are negative. The color intensity varies according to the magnitude of the value.

From M_2 to M_3 Going from model M_2 to M_3, we focus on the word *A-League*. In the first model, M_3 (Figure 4.4a), *A-League* is classified as O, since it being a noun[2] (and not a proper noun) seems to be a good indication of a noun not being part of named entity, among other features, such as *suf+2:r*, i.e., the last letter of the second word to the right of *A-League*, in this case *r*.

With respect to model M_3 (see Figure 4.4b), *A-League* is correctly tagged as ORG. While the largest coefficients are assigned to the features of the model M2 (namely CAP and NN), we can see that the enriched features *CAP_M3*[3] and *suf-1:the_M3* play a decisive role into the assignation of the class ORG, as most of the values corresponding to the newly added features are positive for this class.

From M_3 to M_4 Finally, going from model M_3 to model M_4, in Figure 4.5 we have an incorrect ORG classification to a correct LOC classification after the application of the last fusion operation. The chosen word is *Green*. Both coefficients' values are quite similar to each other (see Figures 4.5a and 4.5b). In fact, the score in parentheses for both LOC and ORG are quite close in both models. This is expected as their difference in performance is small (see Table 4.7). Not surprisingly, there are only two features coming from the last fusion (the last two columns, indicated with a *_M4* suffix. Nonetheless, it seems that one of these enriched features, *word-1:in_M4* determines the model decision towards the LOC class, thus making the correct classification.

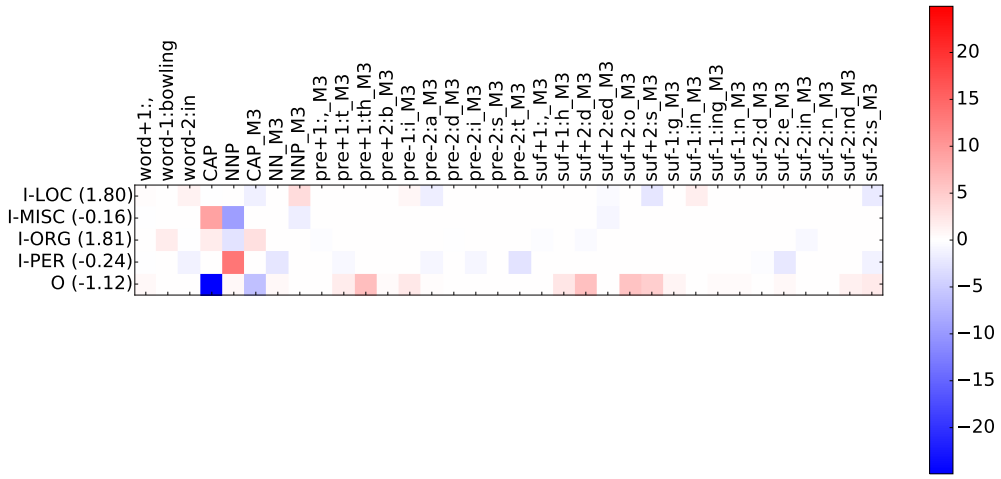
In general, in these experiments, we see that the added enriched features are not the highest valued in the fitted coefficients vectors, nonetheless, they provide the extra information needed to push the model towards the correct prediction, by enriching the features through cross and late fusion and by providing more descriptors for each word and consequently reducing the sparsity of the representation matrices.

Once we found a set of fusion operations that work reasonably well with NER, we experiment with another task, word sense induction and disambiguation, to confirm the usefulness of using fusion enriched representations to train better models.

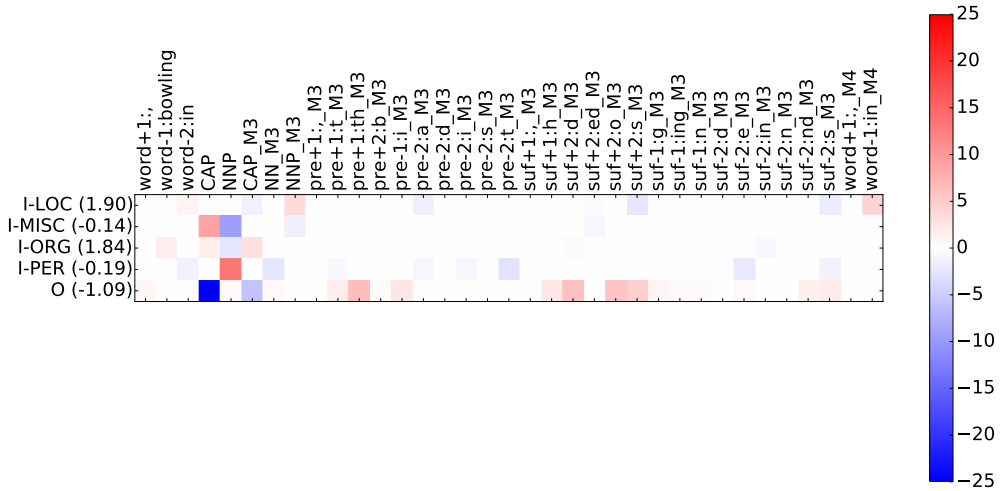
In the next subsection we present a series of analogous experiments, this time solving WSI/WSD.

[2]The PoS tagger identified it as a simple noun.

[3]Features added by the third fusion operation are labeled with a *_M3* suffix. The same is done for the fourth fusion with the suffix *_M4*



(a) The word *Green* is predicted to be class ORG by model M_3 given the coefficients of the features indicated.



(b) The same word is correctly predicted to be class LOC by model M_4 given the new features in the model.

Figure 4.5: Non-zero coefficients heatmaps for models M_3 and M_4 corresponding to the word *Green*. On top, M_3 fitted using the term $E_{\alpha=0.95}(M^L, M^T, L(M^T, X_F(S^S, M^T)))$. On the bottom, model M_4 trained using $E_{\alpha=0.95}(M^L, M^T, L(M^T, X_F(S^S, M^T)), L(M^L, X_F(S^S, M^L)))$ as fusion operator. Red colors are positive, blue are negative. The color intensity varies according to the magnitude of the value.

4.3 Second Application: Word Sense Induction and Disambiguation

Word Sense Induction and Disambiguation entails two closely related tasks[4]. WSI aims to automatically discover the set of possible senses for a target word given a text corpus containing several occurrences of said target word. Meanwhile, WSD takes a set of possible senses and determines the most appropriate sense for each instance of the target word according to the instance's context. WSI is usually approached as an unsupervised learning task, i.e., a cluster method is applied to the words occurring in the instances of a target word. The groups found are interpreted as the senses of the target word. The WSD task is usually solved with knowledge-based approaches, based on WordNet; or more recently with supervised models which require annotated data. It can be also solved reasonably well by comparing the words surrounding each target word and the words belonging to the induced senses (or clusters) found during the WSI step.

We believe that in order to solve WSD in a truly end-to-end unsupervised way, one would need to first automatically find a list of senses for a word without the help of pre-built semantic networks. In other words, solve WSI. Word sense induction is usually solved 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 (this part corresponds to WSD). The common four steps used are the following:

1. Build a lexical co-occurrence network (LCN), or similar, 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 either according to a frequency metric or using binary weights.

3. Apply a graph clustering algorithm. Each cluster found will represent a sense

[4]Even though these tasks are closely related, they are independent from one another. Still, we consider them to be a single one: WSI/WSD.

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.

Word sense induction, while being an unsupervised and thus more flexible task (language and word-domain independent, does not require human-made knowledge bases), require a good quality clustering algorithm, as its results are tightly linked to its performance.

4.3.1 Fusion Enriched Representations

In this subsection we also employ the hypergraph model introduced before to propose a solution to both WSD and WSI tasks, specifically the enrichment of features via fusion techniques.

The WSI method, i.e., the clustering algorithm, we employ is not original, as we will see. Nonetheless, our interest lies on using a combined representation, which is able to address certain concerns that are not deeply studied, namely the use of heterogeneous context features to solve semantic tasks while reducing the number of parameters compared to similar approaches. Our method is evaluated with a corpus corresponding to the WSI task of the international workshop of semantic evaluations, edition 2007, or Semeval 2007.

As shown in Figure 4.6, the procedure we follow is very similar to that of the previous NER experiments. The difference being the task addressed and the features employed (we find clusters using only lexical and syntactic contexts).

We discovered a set of successful fusion operations in the previous experiments. In these experiments we set to test if the improvements obtained before, using said fusion schemes, can be transferred into WSI/WSD and other corpora.

Representation Spaces We use the same set of features from the previous subsection (see 4.2.1), except for the standard NER features, that is, those represented by M^T, as they are specifically designed to tackle that task. Consequently, we will experiment with two representation matrices M^L and M^S.

Learning Methods Regarding the machine learning methods to induce senses, (the WSI part), we employ spectral clustering (as is described previously in Chapter 3,

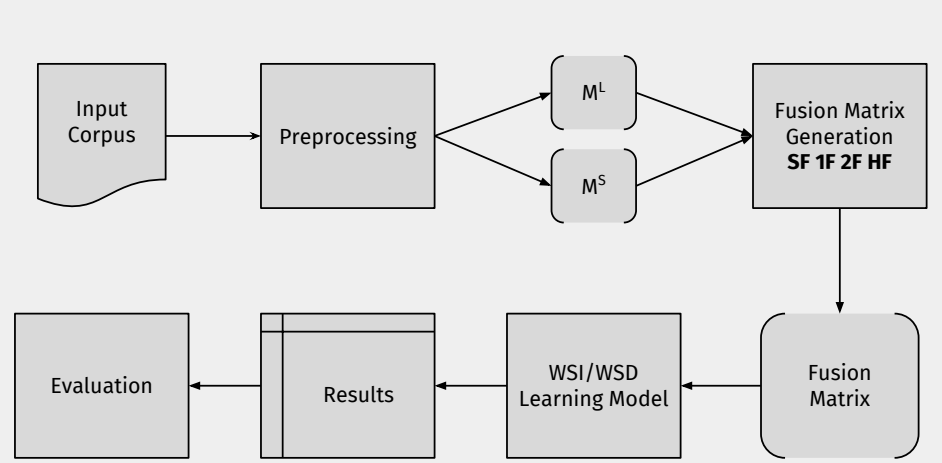


Figure 4.6: Steps followed on our fusion-enriched representation for WSI/WSD experiments. First the corpus is lightly preprocessed, then features are extracted from the text. A fusion matrix is generated, which in turn is used as input to the learning algorithm. Finally, the system yields its results and to be analyzed.

paragraph 3.2.1) on the input matrices in order to automatically discover senses (a cluster is considered a sense). As we noted before, this method require as input a symmetric positive similarity matrix. Given that our fusion operators already may yield similarity matrices (e.g., late fusion of similarity matrices). If that is the case, we use this type of matrix directly as input to the algorithm. Otherwise, we compute the cosine similarity to obtain the corresponding affinity matrix and then use it as input. Similarly, spectral clustering is said to be more efficient if the input matrix is sparse [Song 2008]. We do not sparsify the matrix, usually done by applying a top-k nearest neighbors selection, as we already apply said selection in certain fusion operators and it then becomes even harder to track the modifications applied to the feature space.

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.

Experiments and Evaluation 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. The features we extracted from the tested corpora with the same tools as in the previous task.

Test Dataset The WSI/WSD model is tested on the dataset of the Semeval 2007 WSID task [Agirre 2007a]. 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. The real number of average senses per word is 3.68. This number will be useful to determine the performance of the systems below.

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 2007a].

Ihe 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-measure 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.

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 baseline (or MFS) to each test instance of each target word. In other words, each test instance is assigned the sense that occurs the most in the training set. Consequently, this baseline produces an average of one sense (cluster) per word. A system that goes near this average may be indeed not be resolving the task efficiently but finding the one cluster per word trivial solution. Consequently, to show that we do not fall in the MFS solution, we display in our results the average number of clusters. Furthermore, given this problematic situation we introduce a simple measure, the H-measure, that takes into account three factors of the performance results of a system: the supervised recall of all-words (SR), the unsupervised F-measure of all-words (UF), and the number of true senses on the corpus. The H-measure is calculated as the harmonic mean

of SR and UF plus a weight that grows if the number of induced clusters (#cl) by the system is close to the true average number of senses, which we call δ. More formally:

$$\text{H-measure} = 2 * \frac{SR * UF}{SR + UF} + \log \frac{\delta}{|\#cl - \delta|} \tag{4.3}$$

This metric is not bounded between 0 and 1 as the F-measure and recall. The greater its value, the more confidently we say that the system produces good results. The way it is formulated, having the F-measure and the recall within the formulation serves as an assurance against having systems that are bad but coincidentally produces a correct number of senses. Given that we are calculating the harmonic mean of another harmonic mean (within the F-measure) makes the H-measure severe regarding F-measure and recall. Improvements on both metrics must be had to show a growth in the H-measure.

We consider this measure as a simple method to rank the results that follow, as the metrics provided by the WSI/WSD competitions are not always ideal and have their own issues, and more importantly, because they may contradict each other. Still, the H-measure is not intended to replace the classic metrics.

Results and Discussion Word sense induction and disambiguation results, using fusion enriched matrices with spectral clustering, are found in Tables 4.8 and 4.9 for the supervised recall and unsupervised F-measure respectively. We present the results for al words, nouns and verbs.

In the unsupervised evaluation results, we include an interesting baseline which had also the best performing results. The baseline consists in assigning one cluster per word (or 1c1word), i.e., it simply assigns a single sense to all the test instances of a word. This baseline was not beat during the competition. On the other had, for the supervised results, we include the Most Frequent Sense (or MFS) baseline which tags every test instance with the sense that occurred most often in the training corpus of the competition. Besides these baselines, we included also the best performing systems' results for both of the evaluations.

We experimentally set $\beta = 0.90$ and $\gamma = 50$. Remember that β controls the relevance of each matrix in the late fusion binary operator $L(\beta \cdot A, (1-\beta) \cdot B)$ and γ control number of nearest neighbors to take from the first operand of the cross fusion $\mathbf{K}(A, \gamma) \times B$. The parameter α of the early fusion operator is not employed (i.e., we concatenate matrices without weighting them) unless the value of α is explicitly specified.

Table 4.8: All-words, nouns, and verbs supervised recall for the Semeval 2007 corpus using fusion operations (SF, 1F, 2F, HF) and spectral clustering. We also display the average number of clusters found by each fusion configuration, the best performing system as well as the MFS baseline. In bold the best results per-column among our experiments.

Fusion Operation / System	Recall (%)			#cl	Fusion Level
	all	nouns	verbs		
Single Features					
M^L	79.20	82.10	75.80	4.13	**SF**
M^S	79.10	81.60	**76.20**	4.47	
Early Fusion (EF)					
$E(M^L, M^S)$	78.70	81.11	76.10	4.46	
Cross Feature Fusion (X_FF)					
$X_F(S^L, M^L)$	79.20	82.30	75.70	3.63	
$X_F(S^L, M^S)$	78.30	80.90	75.30	3.08	
$X_F(S^S, M^L)$	78.60	80.90	76.10	1.08	**1F**
$X_F(S^S, M^S)$	78.90	81.40	76.10	2.72	
Cross Similarity Fusion (X_SF)					
$X_S(S^S, S^L)$	78.70	80.90	**76.20**	1.01	
$X_S(S^L, S^S)$	78.80	80.90	76.06	1.33	
Cross Feature Cross Similarity Fusion (X_FX_SF)					
$X_F(X_S(S^L, S^S), M^L)$	78.40	80.40	76.10	3.11	
$X_F(X_S(S^L, S^S), M^S)$	78.90	81.80	75.60	3.16	
Early Cross Feature Fusion (EX_FF)					
$E(M^L, X_F(S^L, M^L))$	79.20	82.40	75.70	3.57	**2F**
$E(M^S, X_F(S^L, M^L))$	78.30	80.50	75.80	1.95	
Late Cross Feature Fusion (LX_FF)					
$L(M^S, X_F(S^L, M^S))$	78.60	81.10	75.80	4.22	
$L(M^L, X_F(S^L, M^L))$	**79.50**	**82.80**	75.70	3.96	
Early Late Cross Feature Fusion (ELX_FF)					
$E(M^L, L(M^S, X_F(S^L, M^S)))$	78.50	81.40	75.40	4.26	**HF**
$E(M^L, L(M^L, X_F(S^L, M^L)))$	**79.50**	82.70	75.90	3.99	
Baseline MFS	78.70	80.90	76.20	1.00	
Semeval 2007 Best System	81.60	86.80	75.70	1.15	

Table 4.9: All-words, nouns, and verbs unsupervised F-measure for the Semeval 2007 corpus using fusion operations (SF, 1F, 2F, HF) and spectral clustering. We also display the average number of clusters found by each fusion configuration, the best performing system as well as the MFS baseline. In bold the best results per-column among our experiments.

Fusion Operation / System	F-measure (%)			#cl	Fusion Level
	all	nouns	verbs		
Single Features					
M^L	72.70	76.90	67.90	4.13	**SF**
M^S	69.30	69.40	69.20	4.47	
Early Fusion (EF)					
$E(M^L, M^S)$	74.00	76.66	71.11	4.46	
Cross Feature Fusion (X_FF)					
$X_F(S^L, M^L)$	76.20	79.60	72.50	3.63	
$X_F(S^L, M^S)$	74.60	75.10	73.90	3.08	
$X_F(S^S, M^L)$	**78.90**	80.70	**76.90**	1.08	**1F**
$X_F(S^S, M^S)$	73.70	77.70	70.00	2.72	
Cross Similarity Fusion (X_SF)					
$X_S(S^S, S^L)$	**78.90**	**80.80**	76.80	1.01	
$X_S(S^L, S^S)$	78.70	80.50	76.80	1.33	
Cross Feature Cross Similarity Fusion ($X_F X_S$F)					
$X_F(X_S(S^L, S^S), M^L)$	70.00	68.70	71.40	3.11	
$X_F(X_S(S^L, S^S), M^S)$	75.20	77.40	72.80	3.16	
Early Cross Feature Fusion (EX_FF)					
$E(M^L, X_F(S^L, M^L))$	76.00	79.50	72.10	3.57	**2F**
$E(M^S, X_F(S^L, M^L))$	75.20	75.40	75.00	1.95	
Late Cross Feature Fusion (LX_FF)					
$L(M^S, X_F(S^L, M^S))$	67.80	71.40	63.80	4.22	
$L(M^L, X_F(S^L, M^L))$	76.09	79.10	72.70	3.96	
Early Late Cross Feature Fusion (ELX_FF)					
$E(M^L, L(M^S, X_F(S^L, M^S)))$	74.20	78.20	69.80	4.26	**HF**
$E(M^L, L(M^L, X_F(S^L, M^L)))$	75.80	78.50	72.70	3.99	
Baseline 1c1word	78.90	80.70	76.80	1.00	
Semeval 2007 Best Systems	78.70	80.80	76.30	3.06	

In the following paragraphs, we will discuss these results obtained. We note that we omit certain configurations that do not yield interesting results either by converging to the MFS solution (one sense found per target word) or because the performance shown by those configurations is simply not interesting. Also, we recall that our objective is to surpass the performance of using of single features, and/or their trivial early fusion combination. Nonetheless, in this WSI/WSD task, there are the baselines we mentioned before (MFS and 1c1word) which are very simple but hard to beat [Agirre 2007a]. Our goal is then to first beat our baselines while keeping an eye on these last two competition baselines.

Single Features Regarding Single Features (SF), M^L comes on top of M^S again, looking at both recall and F-measure regarding all the words (nouns and verbs). Nonetheless, M^S performs better for both metrics in terms of verbs. Thus, syntactic dependencies can provide useful information about verbs. This may be because different senses for different verbs can be better found using dependencies because the differences among head-dependent relations is clearer than between lexical windows of words.

First Degree Fusion On the 1F level, we see that the early fusion techniques in this task does not surpass the independent features representation in none of the metrics. This is unexpected as in NER, the early fusion operator was actually the best baseline during the experiments. This may be due to the fact that the clustering algorithm is sensitive to the noise produced when adding both matrices together, thus reducing the quality of the clusters found.

In cross feature fusion, with respect to the supervised evaluation, the $X_F(S^L, M^L)$ operator that performs as well as M^L, while producing almost the same number of average senses as the true value (3.63 versus 3.68). Indeed, this operator does improves on nouns, although it has lower performance on verbs. We will see later that this is indeed the best performing fusion operator according to our H-measure. Regarding the unsupervised F-measure, the best result is again obtained by $X_F(S^L, M^L)$. This configuration already beats the SF baselines by improving both noun and verb results on the unsupervised evaluation. Nonetheless, while it produces a bit more senses than the MSF average number of senses (1 sense per target word versus 1.08 by this fusion operator), it may be simply approaching the same MSF naive solution, that is, assigning one sense per word.

Looking at cross similarity fusion ($X_S F$), in both tables, we see that both $X_S(S^s, S^L)$ and $X_S(S^L, S^s)$ produces results that are too close to the baseline MFS and 1c1word, implying that we are converging to a naive solution.

Second Degree Fusion In level 2F, regarding the supervised evaluation, going directly to the early cross feature fusion ($EX_F F$), the operator $E(M^L, X_F(S^L, M^L))$ yields as good results as the 1F operator $X_F(S^L, M^L)$ before: it beats the MFS while producing clearly more than a single cluster per word. This result leads us test to test the same operands but combined with a late fusion combination, resulting in the operator $L(M^L, X_F(S^L, M^L))$. The performance obtained with this operator confirmed the intuition of enriching a single feature matrix with another weighted-down matrix to improve the performance. Indeed, we consider that $L(M^L, X_F(S^L, M^L))$ gets the best results in terms of all-words supervised recall (while not considering solutions that are too close to the MFS baseline).

Concerning the all-words unsupervised F-measure, in this level, all the operations surpass the baseline of the naive early fusion ($E(M^L, M^s)$ except for $X_F(X_S(S^L, S^s), M^L)$ and $L(M^L, X_F(S^L, M^s))$. The first one seems to be affected by the quality of the information contained in M^L, compared to M^s, used in the more performing operation $X_F(X_S(S^L, S^s), M^s)$, which is also a $X_F X_S F$. The latter performance-lacking operator in this level, $L(M^L, X_F(S^L, M^s))$, seems to be due the fact that the M^s matrix as the basis of the late fusion operation is not a good choice. If we look into the results of the matrix by itself, we see that it is easily outperformed by M^s.

High Degree Fusion As with the NER experiments on the HF level, the intuition in this stage is to recombine the best previous operators in new fusion modalities. In this case, we present the best performing operations. We note that we tried other fusions but they were found to have low results. As an example of these failed configurations, we tried $E(M^L, X_F(S^L, M^L))$ both recombined through an early and a late fusion operations. Furthermore, in order to have coherence with the best result obtained in NER, we tried solving WSI/WSD using the Triple Early Double Late Cross Feature Fusion (or $EEELX_F LX_F$). Unfortunately, the results in this level were not as interesting as before. Nevertheless, we present the two most successful high degree fusion operators found. The two operators we test in this level do improve on the early fusion baseline. However, they are not able to improve over the 1c1word baseline.

Indeed, contrary to what we reported in the previous NER experiments, the best

results are generally obtained in the 2F level, and not in the HF level, according to the all-word supervised recall and unsupervised F-measure. Still, it is clear that the best recombination of fusion operations (yield better results than our established baselines (single features and early fusion) and the MFS baseline.

Specifically, regarding supervised recall, the operations $L(M^L, X_F(S^L, M^L))$ and $E(M^L, L(M^L, X_F(S^L, M^L)))$ (with a performance of 79.5%) surpass the MFS baseline (78.7%), both single feature matrices M^L and M^S (79.2%), and the early fusion trivial operation $E(M^L, M^S)$ (with 78.7%). Concerning the unsupervised F-measure, we do surpass our two baselines but not the 1c1word competition baseline. While considering this performance metric is harder to determine the best performing model. There are several fusion operations that match the performance of the 1c1word baseline, although the number of clusters produced is very close to one. This is the case of the $X_F(S^S, M^L)$ and $X_S(S^S, S^L)$ operators, with 78.9% F-measure and generating 1.08 and 1.01 clusters respectively.

In order to determine the best performing operators, that stray away from the trivial baselines, in the following we consider the H-measure, introduced before, to help us identify those systems that perform the best. In Figure 4.7, we see the H-measure for each of the fusion operators reported. We calculate the H-measure with $\delta = 3.68$, as this is the number of true senses per target word in the Semeval 2007 corpus. According to this metric, we find the 1F and 2F degree levels the most interesting. Specifically, $X_F(S^L, M^L)$, in 1F, is the most performing fusion operator. Indeed, by transferring quality lexical similarities into its same feature matrix, we obtain more useful relations than by using any syntactic data. On the other had, in second place, the operator $E(M^L, X_F(S^L, M^L))$ of the 2F level consists also exclusively on lexical information. While the same operators that outperformed the rest in NER (in the HF level) are not as adequate in this WSI/WSD experiment, we see that most of the feature combination techniques improve over the baselines of the single features and early fusion operations.

In the following subsection, we put aside the fusion enrichment and we focus into another characteristic of our proposed hypergraph structure. We leverage the links (features) among nodes (words) to induce senses. Specifically, we propose a method that clusters together words which represent induced senses for a set of target words.

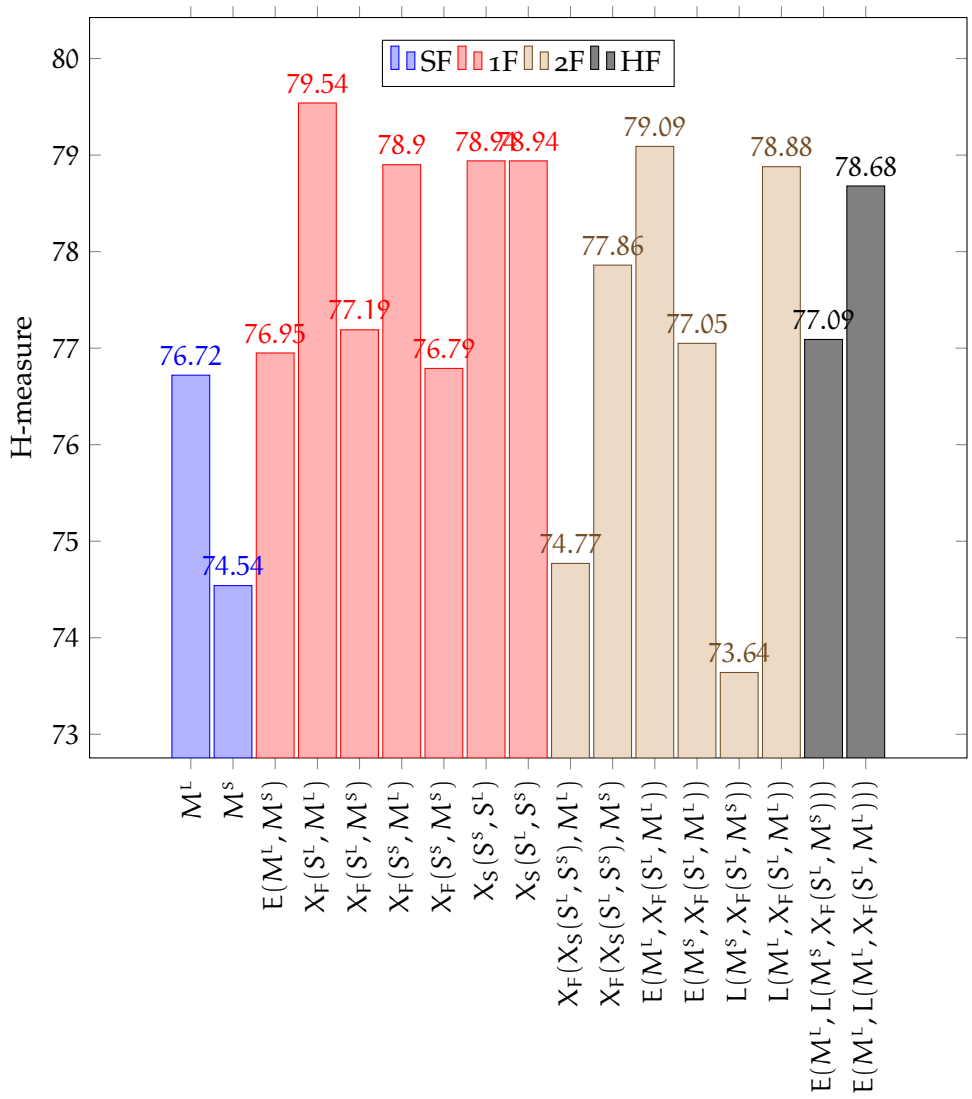


Figure 4.7: H-measure for the WSI/WSD task on the Semeval 2007 corpus. Results obtained with the spectral clustering algorithm. The best performing operator is the $X_F(S^L, M^L)$.

4.3.2 Leveraging the Linguistic Network Structure

Until now, we have employed the hypergraph representation in terms of leveraging the heterogeneous information to enrich and densify a feature space. Now, we will leverage the relations that exist within the network to identify words that, together with their neighborhood, represent a sense. Thus, we propose a network-based algorithm to solve word sense induction.

With respect to the information contained in the network, we find that few approaches include syntactic attributes into their model. We believe that finding semantic similarities can be improved by leveraging syntactic information by using dependency relations.

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 is inspired on previous approaches from both [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, called "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

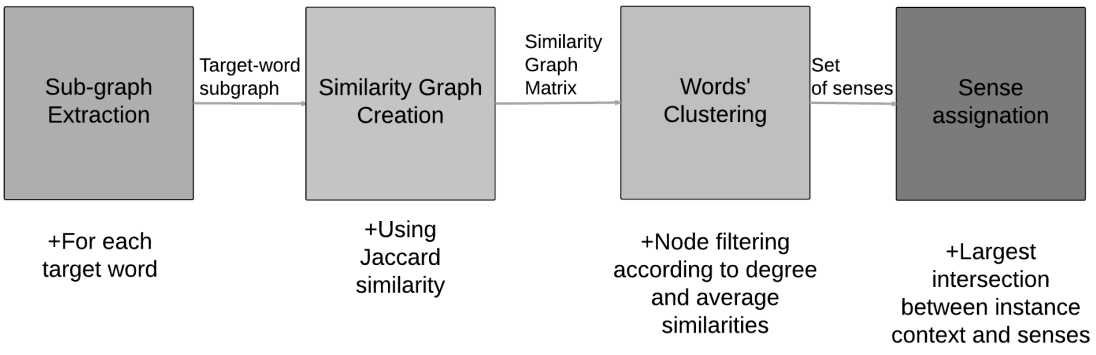


Figure 4.8: Block diagram of the WSI/WSD method proposed.

to have the closest set of nodes is chosen as the target word sense.

The second approach, UoY, described in [**?**], 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 4.8 shows an overview of the process. Also, in Algorithm 2 we show the general flow of our approach. We detail the steps taken alongside the corresponding line in the algorithm below.

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

Algorithm 2: 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

```
1  foreach target word tw in tw_set do
2      G_tw = extract_subgraph(G_H, tw);
3      B_tw = induce_bipartite_graph(G_tw);
4      S_tw = sim_matrix(B_tw);
5      F_tw = induce_hypergraph(S_tw, th1);
6      candidate_hubs = sort(degree(F_tw))[:100];
7      SoS_tw = [ ];
8      foreach candidate_hub in candidate_hubs do
9          candidate_hyperedges = get_hyperedges(candidate_hub, F_tw);
10         candidate_avg_jaccard = 0;
11         foreach hyperedge in candidate_hyperedges do
12             candidate_avg_jaccard += get_avg_jaccard(hyperedge);
13         end
14         if candidate_jaccard > th2 then
15             SoS_tw.add(get_words(candidate_hyperedges));
16             F_tw = F_tw \ candidate_hyperedges;
17     end
18     return SoS_tw
19 end
```

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 dependency head token. On the other hand, when dealing with a verb target word, we select the co-occurrent words having said verb as head of the dependency relation. The reason is that usually verbs are more often than not the head of dependency relations, so the intuition is that words which have the same verb governor are somehow semantically related.

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 $Jaccard(i,j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$ 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:

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

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

Experiments and Evaluation

Test Datasets We trained and evaluated our system on two datasets: Semeval 2007 Task 2 (as in the previous experiments) and Semeval 2010 Task 14 [Manandhar 2010]. We recall that 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. The average number of true senses in this dataset is 3.79. On this version of Semeval, 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 present within the test set.

We apply a light pretreatment, 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.

Implementation The objective of this experiment is to understand the performance of both lexical and syntactic co-occurrence information, used independently, while solving WSI and WSD tasks while using the method described in the previous subsection. To that end we build two independent systems, using : SEN, which uses exclusively lexical co-occurrence hyperedges, and DEP, which employs only syntactic dependency hyperedges. Both are obtained as described in Chapter 3, section 3.3.1.

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 SEN 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 (th_1) and for

the average Jaccard similarity (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.

Results and Discussion

Table 4.10: Unsupervised F-measure for the Semeval 2007 test set. The highest values for each column are in bold. These results are obtained with our proposed algorithm.

System	F-measure (%)			#cl
	all	nouns	verbs	
1c1word	**78.9**	80.7	**76.8**	1.0
UBC-AS	78.7	**80.8**	76.3	1.3
SC_{fusion}	76.2	79.6	72.5	3.6
GB_{fusion}	75.9	78.7	71.8	3.4
DEP	74.9	80.2	69.0	3.27
SEN	61.4	62.6	60.1	4.26
UoY(2007)	56.1	65.8	45.1	9.3
Random	37.9	38.1	37.7	19.7
1c1instance	9.5	6.6	12.7	48.51

Semeval 2007 In Table 4.10 we present the unsupervised F-measure evaluation results for our models as well as for some other systems: the previously presented UoY (version 2007) and UBC-AS [Agirre 2007b], the best performing system according to this measure. Additionally, three baselines are included, the already-known 1c1word plus two new ones, the Random baseline which assigns a sense randomly to each test instance; and the 1c1instance baseline, which attributes a unique sense to each one of the test instances. Finally, we also include the fusion operator that obtained the highest performance (according to the H-measure) in the experiments of the previous subsection, that is, the results of the $X_F(S^L, M^L)$ fusion. As the last experiments were performed with spectral clustering, we will call this system SC_{fusion}. Furthermore, given that we generate representation spaces, we can use the same fusion matrix but this time using the graph-based algorithm presented in this subsection, instead of the spectral clustering used before. We call this system GB_{fusion}

Table 4.11: Supervised Recall (SR) on the Semeval 2007 test set. The highest values for each column are in bold. Results are obtained with our proposed algorithm.

System	Recall (%)			#cl
	all	nouns	verbs	
I2R	**81.6**	**86.8**	75.7	3.1
GB_{fusion}	79.6	83.3	76.3	3.4
SEN	79.4	82.5	75.9	4.3
SC_{fusion}	79.2	82.3	75.7	3.6
DEP	79.1	81.5	**76.4**	3.3
MFS	78.7	80.9	76.2	1.0
UoY(2007)	77.7	81.6	73.3	9.3

In this table, as in the rest of the tables presented in this section, as before, the columns show the results for all the words, for the nouns, and for the verbs. Again, 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 aim to address this shortcoming with our H-measure.

Looking at the table, we can see 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-measure value. We also note that in this evaluation DEP, the system using only co-occurrent dependency relations outperformed the lexical co-occurrence only system SEN. It is very possible that this lack of lexical performance is due to the size of the surrounding words window, which in this case is selected to be the entire phrase were the word occurs.

With respect to our fusion space methods, SC_{fusion} and GB_{fusion} we find that both are competitive but cannot achieve a better performance that those systems prepared for the competition. Both systems are better than using independent features, according to the unsupervised F-measure. GB_{fusion} Nonetheless, for supervised recall, only GB_{fusion} beats both of the single feature matrices.

Moving onto the supervised results for Semeval 2007, in Table 4.11 we show the

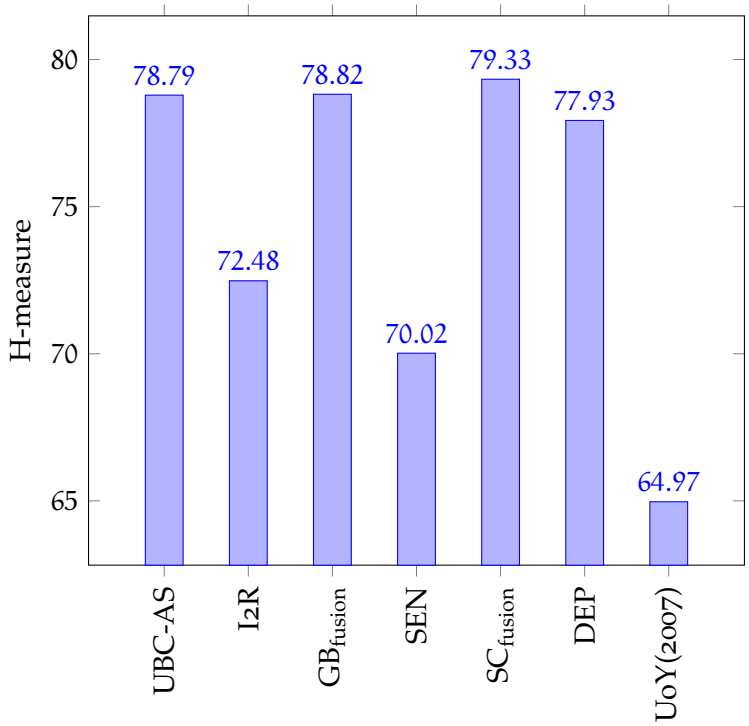


Figure 4.9: H-measure for the WSI/WSD task on the Semeval 2007 corpus using our network-based method as well as other systems, indicated on the x-axis. The best performing system was spectral clustering with a fusion representation space SC_{fusion}.

results obtained concerning supervised recall. In this table we include the competing system I2R [Niu 2007], 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 SEN 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).

We now look into the results from the H-measure perspective (see Figure 4.9). Again, as in the experiments with fusion opeartions, the best system SC_{fusion} uses spectral clustering with a single cross media fusion operator, i.e., $X_F(S^L, M^L)$. Close, in second place, we find the other fusion based system, this time using the network-based method proposed previously. Finally, the system UBC-AS is ranked third by the H-measure.

In general, it is interesting to notice that the fusion operators outperform again the single features and is on pair with other performing systems. What does seems

unexpected is that the roles played by the DEP and SEN system is inverted regarding to the fusion experiments presented in the previous subsection. Indeed, using our network-based approach the performance of DEP is considerably larger than that of SEN, whereas using spectral clustering the lexical information outranked (by a small margin) the syntactic information (based on dependencies). Again, we attribute this lack of performance of SEN to the size of the window employed, which seems to general to detect appropriate senses.

As a way of determining where does both systems complement each other, in Figure 4.10 and Figures 4.11 and 4.12 we show the unsupervised F-measure 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 (both nouns and verbs) that obtain better scores using SEN 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. Again, the SEN system does better while finding senses and assigning them to the verbs *avoid*, *fix*, and *work*.

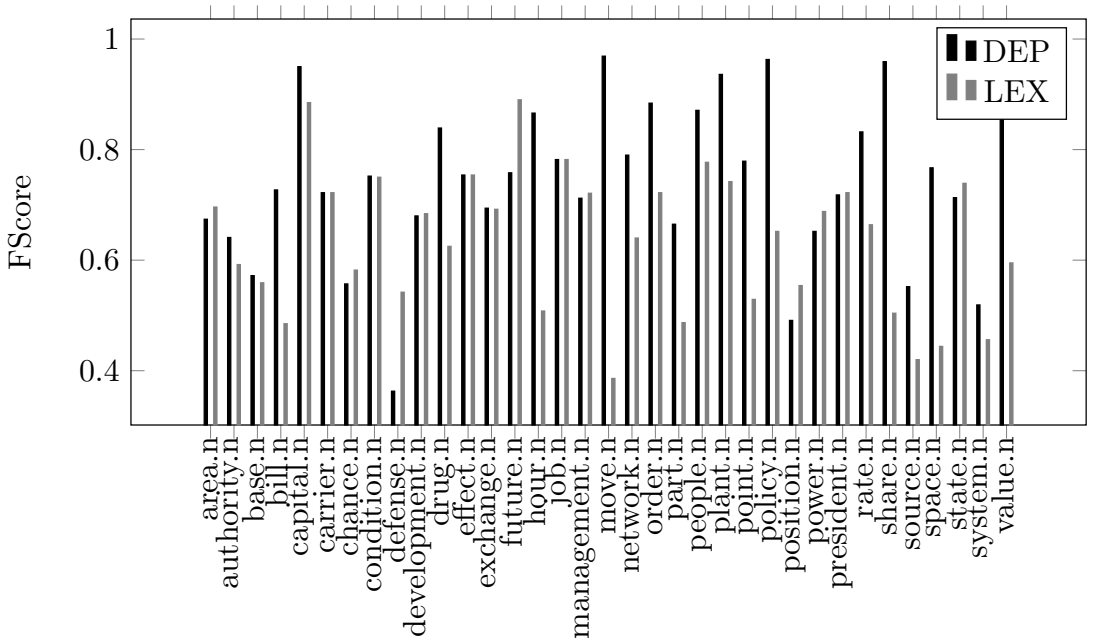


Figure 4.10: Unsupervised F-measure results for the nouns of the Semeval 2007 test set.

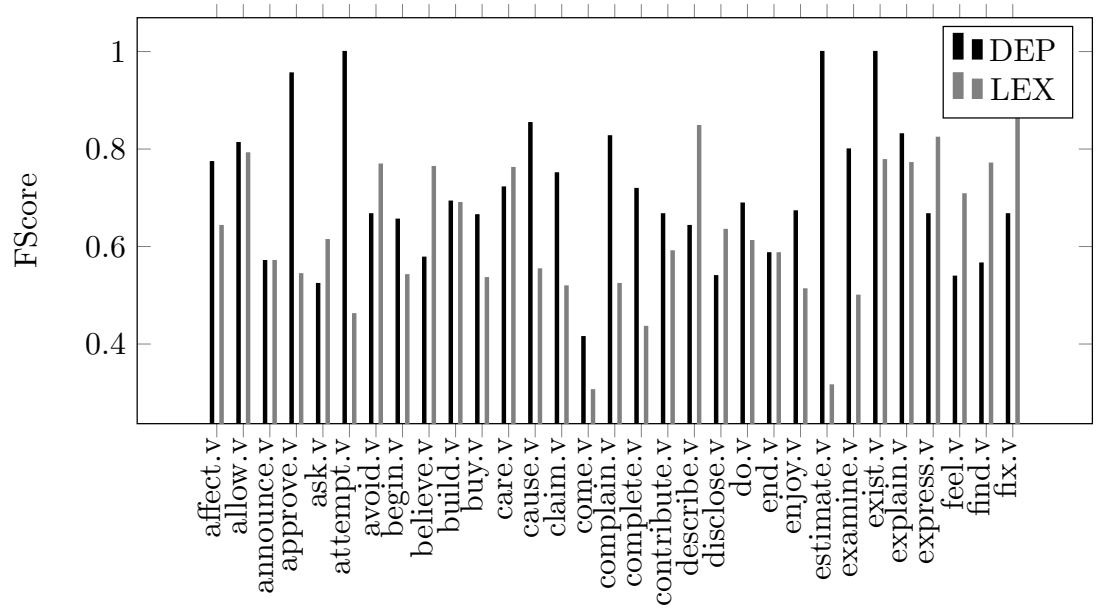


Figure 4.11: Unsupervised F-measure results for the first half of verbs of the Semeval 2007 test set.

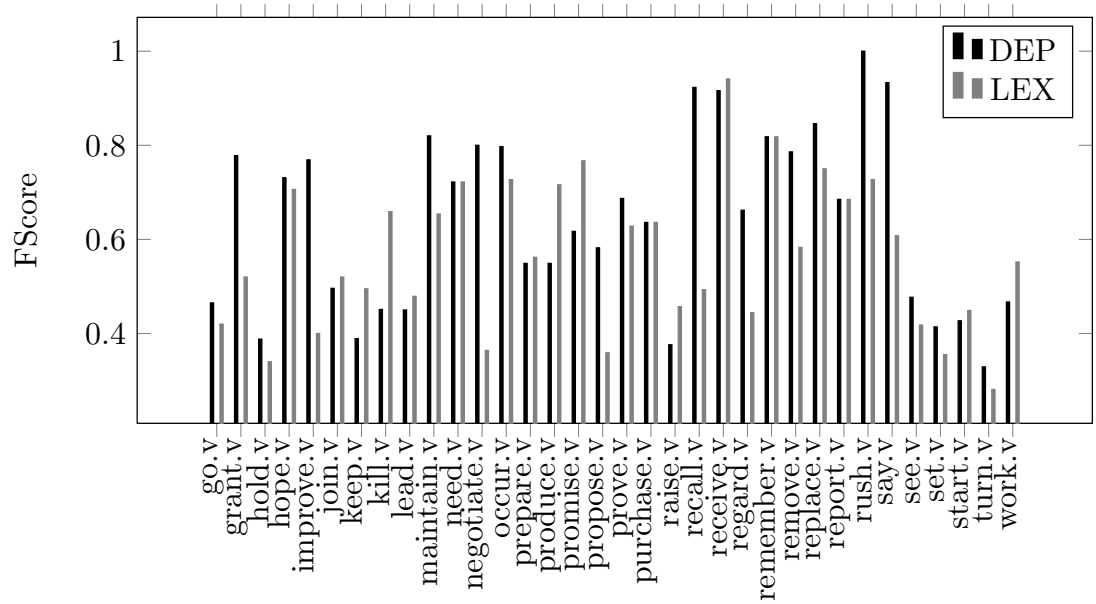


Figure 4.12: Unsupervised F-measure results for the second half of verbs of the Semeval 2007 test set.

Semeval 2010 In the Semeval 2010 dataset [Manandhar 2010], two unsupervised evaluation metrics are introduced: V-measure and paired F-measure. Briefly, the V-measure assesses the homogeneity or 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. It also takes into account completeness, or the level each gold standard class consists of instances primarily assigned to a single cluster. The V-measure is always greater than zero but it is not bounded to one as maximum value. We consider that the larger the measure, better. A well-known drawback is that the V-measure is biased towards systems that yield many senses [Manandhar 2010, Van de Cruys 2011, Pedersen 2010a]. We present this metric for completeness but we do not heavily base our conclusions on it.

On the other hand, the paired F-measure evaluation transforms the clustering problem into a classification. 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 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 the gold standard. The paired F-measure is the harmonic mean of both quantities.

Concerning the supervised evaluation, it follows the same of Semeval 2007, using recall as the main performance measure. The key difference is that the final results of the competition were obtained by reshuffling the data several times to avoid the issue of obtaining considerably different results with different splits of the data.

In Table 4.12 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 2010a]. 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 SEN 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 $\text{NMF}_{\texttt{lib}}$ [Van de Cruys 2011] and Hermit [Jurgens 2010]. The former did not participate on the competition but was developed later. We include it henceforth

Table 4.12: Unsupervised V-measure on the Semeval 2010 test set. The highest values for each column are in bold. Our results are obtained with our proposed algorithm.

System	V-measure (%)			#cl
	all	nouns	verbs	
Hermit	**16.2**	**16.7**	**15.6**	10.8
NMF_{lib}	11.8	13.5	9.4	4.8
SEN	11.6	8.8	11.9	10.5
Random	4.4	4.2	4.6	4.0
DEP	3.5	3.9	2.8	2.8
MFS	0.0	0.0	0.0	1.0

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 SEN. They were not included for the sake of clarity.

The second unsupervised measure, Paired F-measure, can be seen in Table 4.13. In this case both systems presented performed better than the random baseline. All the systems presented were able to beat the MFS baseline. We note that DEP does much better compared to SEN concerning verbs, namely 58% vs. 28% F-measure. Still, the results are low considering the best results of the competition, 63% from the Duluth system [Pedersen 2010b] , 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-measure. Our results are obtained with our proposed algorithm.

Finally, we in Table 4.14 we show the supervised Recall results of Semeval 2010. The best performing algorithm shown is NMF_{lib}. During the competition, UoY(2010) was the best method. It is a graph-based algorithm which shares the name with the UoY(2007) 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.

Table 4.13: Unsupervised Paired F-measure for the Semeval 2010 test set. The highest values for each column are in bold. Our results are obtained with our proposed algorithm.

System	Paired F-measure (%)			#cl
	all	nouns	verbs	
MFS	**63.5**	**57.0**	**72.4**	1.0
Duluth-WSI-SVD-Gap	63.3	**57.0**	**72.4**	1.0
DEP	53.6	50.1	58.7	2.8
NMF_{lib}	45.3	42.2	49.8	5.4
SEN	38.4	46.7	28.5	10.5
Random	31.9	30.4	34.1	4.0

Table 4.14: Supervised recall for Semeval 2010 test set (80% mapping, 20% evaluation). The highest values for each column are in bold. Results are obtained with our proposed algorithm.

System	Recall (%)			#cl
	all	nouns	verbs	
NMF_{lib}	**62.6**	57.3	**70.2**	5.4
UoY(2010)	62.4	**59.4**	66.8	11.5
SEN	59.8	55.8	67.4	10.5
DEP	59.3	53.9	67.2	2.8
MFS	58.7	53.2	66.6	1.0
Random	57.3	51.5	65.7	4.0

Again, in order to find the overall best systems, we present the H-measure as before. This time the H-measure will be the harmonic mean of the newly introduced unsupervised paired F-measure instead of the plain F-measure used before. The supervised recall stays also as before. We do not employ the V-measure for the reasons exposed before, namely it is biased towards systems that generate many senses. Ideally, the paired F-measure was conceived to punish systems that produced less or more senses than the true average number of senses for each target word. In reality, it does not seem to be the case. We present in Figure 4.13 the H-measure ($\delta = 3.79$) results for

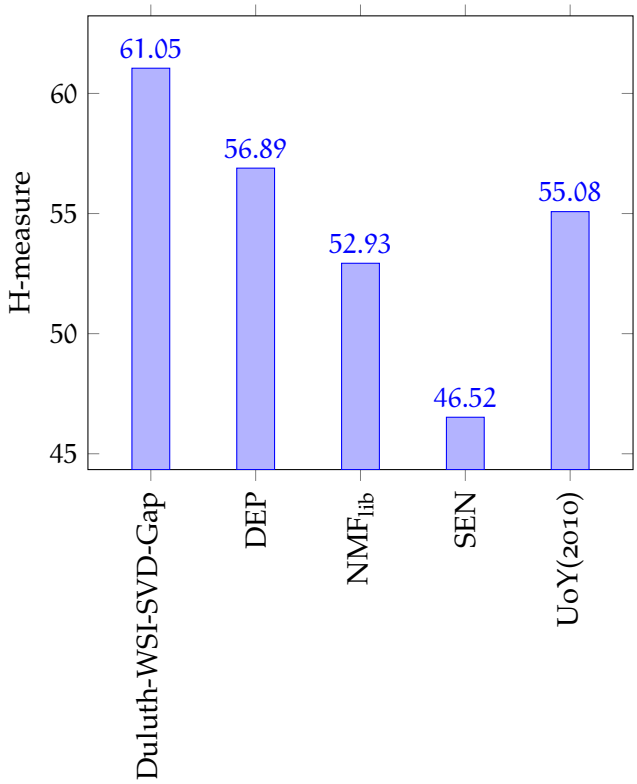


Figure 4.13: H-measure for the WSI/WSD task on the Semeval 2010 corpus using our network-based method as well as other systems, indicated on the x-axis. The best performing system is Duluth-WSI-SVD-Gap, while our best method is that based on syntactic information DEP.

the systems discussed previously in the Semeval 2010 context. Duluth-WSI-SVD-Gap is the best system overall, followed by our approach using DEP as input matrix.

Looking at the results, we argue that the Semeval 2010 task was more complex than Semeval 2007. We believe it is in part due to the fact that there are more verbs in the dataset, which are always harder to disambiguate than nouns. The dataset was also considerably larger, which may have introduced noise and required more sophisticated filtering techniques. We see that our system SEN performs poorly. Indeed, it generates too many senses, which is a symptom of poorly generated clusters, that is, the clusters found are not distinctive nor cohesive enough to enclose whole senses. On the other hand DEP does reasonably well, maybe in part due to the fact that again there are more verbs whose senses are better distinguished using syntactic information.

4.4 Conclusion

In this chapter we addressed two NLP tasks from two different points of view: on the one hand, we computed several representation spaces using fusion operations in order to enrich and densify otherwise sparse and independent features. The matrices generated were used to train both supervised and unsupervised models to solve named entity recognition and word sense induction and disambiguation.

On the other hand, we proposed a model that leverages the inner structure of the hypergraph network to group words that belong to a shared sense. This approach was used to solve word sense induction.

More specifically, concerning the first part, we presented a comparative study of multimedia fusion techniques applied to named entity recognition. We also tested hybrid 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 analyzed the results to understand how the enrichment of features improved the performance. We found that at each fusion step, a different type of NER tag is benefited. We studied what features where driving the decision towards the correct class and found that while the enriched features are not the most prominent in the decision function, they play an important role by tipping said decision towards the correct label and away from the wrong one.

Concerning fusion enrichment and WSI/WSD, we found that the fusion operations also improve the results of the task, although not as clearly as in NER. The metrics used to measure the performance on this ask does not allow a clear understanding on the behavior of the model employed. While we want to avoid converging to the trivial one sense per word solution, we know that words do not have numerous senses. In that sense, the results obtained stay reasonably away from the trivial solution while not producing many senses as other approaches.

We proposed a metric, the H-measure, to rank the systems by considering the classic performance metrics and the number of senses found relative to the true number of senses. This metric allowed us to identify with a single value the best system. We found that according to it, the fusion based systems, whether using well-known algorithms (spectral clustering) or using the method developed in this section, perform adequately and show general improvements over the single feature representations as

well as other systems.

While there is an improvement using fusion techniques, we do note that they enlarge the feature space, especially early fusion, which is used frequently. This may imply the need of larger quantities of memory and longer execution time. In that sense, as future work, 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 high 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.

Concerning our proposed network-based method, 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].

Chapter 5

Conclusions and Future Work

5.1 Conclusion

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 and second entail a fusion enriched linguistic network. The third is 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.

These heterogeneous features are important as they represent relations between words from different points of view. Nonetheless, they tend to generate sparse representations, as is common with text representations. In this sense, we proposed as second contribution the use of fusion techniques to combine these features while reducing the sparsity of the representation space.

Lastly, the third contribution entails a network-based method that leverages the structure of the 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.

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.

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.

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. Understanding what is the syntactic or lexical difference among contexts, which induce the good or bad performance of each type of feature could make the system more flexible to other text domains. Also, the hypergraph could be better leveraged by using hypergraph-specific methods, mainly through spectral analysis.

Lastly, we built a enriched hypergraph resource based on a very large corpus such as Wikipedia. The relations between words contained within could be leveraged to generate more powerful representations as the one created in this work. Indeed, we explored this avenue but was put aside given the size of a matrix extracted from a very large corpus. Furthermore, a comparison with other distributional representations may signal other advantages of our proposed model. Specially for smaller corpora.

Bibliography

[Aggarwal 2012] Charu C. Aggarwal and ChengXiang Zhai, editors. Mining text data. Springer, 2012. (Cited on pages 3, 29 and 82.)

[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 45 and 49.)

[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 pages 103 and 107.)

[Agirre 2007b] Eneko Agirre and Aitor Soroa. *UBC-AS: A Graph Based Unsupervised System for Induction and Classification*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 346–349, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on page 117.)

[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 42, 45 and 49.)

[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 42, 45 and 49.)

[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 34, 61, 66 and 87.)

[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 47.)

[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 58 and 64.)

[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 85 and 86.)

[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 20, 21 and 22.)

[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 32.)

[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 18.)

[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 32.)

[Berge 1985] C. Berge. Graphs and hypergraphs. Elsevier, 1985. (Cited on page 52.)

[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 69.)

[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 47.)

[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 20.)

[Booth 1955] Andrew Donald Booth. *Machine translation of languages, fourteen essays*. 1955. (Cited on page 14.)

[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 45.)

[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 42, 46 and 49.)

[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 16 and 34.)

[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 67.)

[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 26 and 27.)

[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, Model-

ing and Simulation in Science, Engineering and Technology, pages 145–166. Birkh?user Boston, 2009. (Cited on page 69.)

[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 23.)

[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, 20, 23 and 24.)

[Clark 2015] Stephen Clark. *Vector space models of lexical meaning*. Handbook of Contemporary Semantic Theory, The, pages 493–522, 2015. (Cited on page 24.)

[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 47.)

[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 60, 61 and 87.)

[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 pages 48 and 85.)

[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 16 and 32.)

[Daumé III 2006] Harold Charles Daumé III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, 2006. AAI3337548. (Cited on pages 17, 48, 84 and 85.)

[Daumé III 2012] Hal Daumé III. *A course in machine learning*. chapter, vol. 5, page 69, 2012. (Cited on page 48.)

[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 103.)

[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 20.)

[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 20.)

[De Saussure 1916] Ferdinand De Saussure. *Course in general linguistics (trans. Roy Harris)*. London: Duckworth, 1916. (Cited on page 14.)

[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 40, 44 and 49.)

[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 44 and 49.)

[Estrada 2005] Ernesto Estrada and Juan A Rodriguez-Velazquez. *Complex networks as hypergraphs*. 2005. (Cited on page 40.)

[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 16.)

[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 24.)

[Firth 1957] J. R. Firth. *A synopsis of linguistic theory 1930-55.* vol. 1952-59, pages 1–32, 1957. (Cited on page 14.)

[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 68.)

[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 ix and 33.)

[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 87.)

[Giuseppe Attardi 2015] Giuseppe Attardi. *WikiExtractor*. `https://github.com/attardi/wikiextractor`, 2015. (Cited on pages 68 and 70.)

[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 21.)

[Goyal 2014] Kartik Goyal and Eduard H. Hovy. *Unsupervised Word Sense Induction using Distributional Statistics*. In COLING, pages 1302–1310. ACL, 2014. (Cited on pages 47 and 48.)

[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.)

[Gupta 2015] Sonal Gupta. *Distantly Supervised Information Extraction Using Bootstrapped Patterns*. PhD thesis, Stanford University, 2015. (Cited on page 82.)

[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 29.)

[Harris 1954] Zellig Harris. *Distributional structure*. vol. 10, no. 23, pages 146–162, 1954. (Cited on page 13.)

[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 52.)

[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 47.)

[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 44, 47 and 49.)

[Jaccard 1908] P. Jaccard. Nouvelles recherches sur la distribution florale. Bulletin de la Société vaudoise des sciences naturelles. Impr. Réunies, 1908. (Cited on page 25.)

[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 67, 69 and 73.)

[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, 16, 17, 18, 21, 22 and 23.)

[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 23 and 30.)

[Jurgens 2010] David Jurgens and Keith Stevens. *HERMIT: Flexible Clustering for the SemEval-2 WSI Task*. In SemEval@ACL, pages 359–362. The Association for Computer Linguistics, 2010. (Cited on page 122.)

[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 40, 47 and 49.)

[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 67.)

[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 21, 22, 24 and 127.)

[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 43, 44, 46 and 49.)

[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 40, 46, 49, 52 and 111.)

[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 40, 42, 47 and 49.)

[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 42, 44, 47 and 49.)

[Lebret 2013] Rémi Lebret, Joël Legrand and Ronan Collobert. *Is deep learning really necessary for word embeddings?* Technical report, Idiap, 2013. (Cited on page 16.)

[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 31.)

[Lehecka 2015] Tomas Lehecka. *Handbook of Pragmatics 2015*. chapter 2. John Benjamins, 2015. (Cited on page 15.)

[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 17, 20, 21, 34, 76, 84 and 88.)

[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 16, 17 and 32.)

[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 22, 23, 24, 31 and 32.)

[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 20 and 21.)

[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 41, 46, 47, 49 and 52.)

[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.)

[Luxburg 2007] Ulrike Luxburg. *A Tutorial on Spectral Clustering*. Statistics and Computing, vol. 17, no. 4, pages 395–416, December 2007. (Cited on pages 47 and 48.)

[Manandhar 2010] 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. (Cited on pages 116 and 122.)

[Manning 1999] Christopher D Manning, Hinrich Schütze*et al.* Foundations of statistical natural language processing, volume 999. MIT Press, 1999. (Cited on page 22.)

[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 24.)

[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 page 71.)

[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 69.)

[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 45 and 49.)

[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 42, 44, 45 and 49.)

[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 25, 26, 27 and 69.)

[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 16, 17 and 32.)

[Molino 2017] Piero Molino. *Word Embeddings Past, Present and Future*, 06 2017. AI With The Best AIWTB 2017. (Cited on pages ix, 15 and 30.)

[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 42, 45, 46 and 49.)

[Murphy 2012] Kevin P. Murphy. Machine learning: A probabilistic perspective. The MIT Press, 2012. (Cited on page 45.)

[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 pages 4 and 82.)

[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 26 and 39.)

[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 40, 42, 44, 45 and 49.)

[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 40, 44 and 49.)

[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 14.)

[Niu 2007] Zheng-Yu Niu, Dong-Hong Ji and Chew-Lim Tan. *I2R: Three Systems for Word Sense Discrimination, Chinese Word Sense Disambiguation, and English Word Sense Disambiguation*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 177–182, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on page 119.)

[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 20.)

[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 23.)

[Nothman 2009] Joel Nothman, Tara Murphy and James R. Curran. *Analysing Wikipedia and Gold-standard Corpora for NER Training*. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, EACL '09, pages 612–620, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. (Cited on pages 84 and 86.)

[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 34.)

[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 21.)

[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 20, 21 and 84.)

[Pedersen 2010a] 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 page 122.)

[Pedersen 2010b] Ted Pedersen. *Duluth-WSI: SenseClusters Applied to the Sense Induction Task of SemEval-2*. In Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval '10, pages 363–366, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Cited on page 123.)

[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 32.)

[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 16, 21 and 30.)

[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 29.)

[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 30.)

[Qian 2014] 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. (Cited on pages 40, 47, 49 and 52.)

[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 30, 84 and 86.)

[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 69.)

[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 17, 21, 30 and 34.)

[Sahlgren 2008] Magnus Sahlgren. *The distributional hypothesis*. Italian Journal of Disability Studies, vol. 20, pages 33–53, 2008. (Cited on pages ix, 14, 15 and 17.)

[Saluja 2013] Avneesh Saluja and Jiri Navrátil. *Graph-Based Unsupervised Learning of Word Similarities Using Heterogeneous Feature Types*. page 29, 2013. (Cited on pages 43, 44, 46 and 49.)

[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 86.)

[Schaeffer 7 08] Satu Elisa Schaeffer. *Survey: Graph Clustering*. vol. 1, no. 1, pages 27–64, 2007-08. (Cited on page 46.)

[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 68.)

[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 20.)

[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 15.)

[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 68.)

[Shi 2000] Jianbo Shi and Jitendra Malik. *Normalized Cuts and Image Segmentation*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pages 888–905, August 2000. (Cited on page 47.)

[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 42, 45, 46 and 49.)

[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 42, 43, 44, 45 and 49.)

[Song 2008] Yangqiu Song, Wen-Yen Chen, Hongjie Bai, Chih-Jen Lin and Edward Y Chang. *Parallel spectral clustering*. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 374–389. Springer, 2008. (Cited on page 102.)

[Song 2016] Linfeng Song, Zhiguo Wang, Haitao Mi and Daniel Gildea. *Sense Embedding Learning for Word Sense Induction*. In *SEM@ACL. The *SEM 2016 Organizing Committee, 2016. (Cited on page 47.)

[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 11.)

[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 11.)

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

[Tsatsaronis 2007] George Tsatsaronis, Michalis Vazirgiannis and Ion Androutsopoulos. *Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri*. 2007. (Cited on pages 42, 43, 44, 46 and 49.)

[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 14, 16, 21 and 23.)

[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 32 and 122.)

[Véronis 2004] Jean Véronis. *HyperLex: lexical cartography for information retrieval*. vol. 18, no. 3, pages 223 – 252, 2004. (Cited on pages 40, 46, 49 and 111.)

[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 67.)

[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 67.)

[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 43, 46 and 49.)

[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 64.)