UNIVERSITY OF LYON - LUMIÈRE LYON 2

ECOLE DOCTORALE INFOMATHS

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University of Lyon - Lumière 2

Specialty : Computer Science

Defended by

Edmundo-Pavel Soriano-Morales

Design and Use of Aatomical Atlases for Radiotherapy

Thesis Advisor: Sabine Loudcher and Julien Ah-Pine

prepared at Université de Lyon 2, ERIC Laboratory defended on

Jury :

Contents

Chapter 1

Introduction

Contents

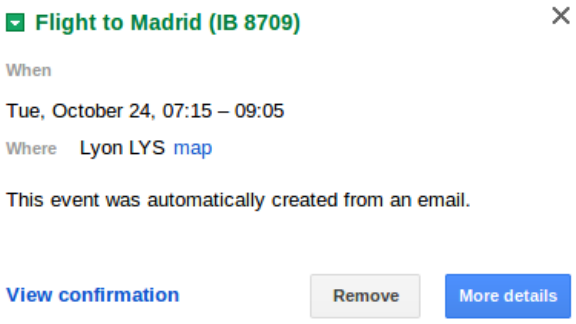
1.1 Context

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 multi-disciplinary area, Natural Language Processing[1] (NLP) is the domain that

[1] And its more applied branch, text mining or text analysis. While it is sometimes argued that text

(a)



(b)

Figure 1.1: (a) While searching *Who invented Python?*, Google recongizes 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.

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.

mining deals with the structured data information extracted from text, I believe both terms considerably overlap nowadays. For ease of readability, in this dissertation we use both terms interchangeably, while preferring Natural Language Processing.

- 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[2], Microsoft Azure[3], etc; or furnished by public organizations, such as France's Lyon 1 University[4] or the National Institute of Nuclear Physics computing centers[5].

- 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[6], Antwerp's University CLiPS Pattern[7]. Additionally, large Web companies, such as Facebook[8] and Google[9], 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`[10], a popular data science library implementing several well-known machine learning algorithms. Regarding NLP specifically, two up-to-date libraries stand out: `gensim`[11] and `spaCy`[12]. 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] . 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**, a large number of features is 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 knowledge that reveals an interesting piece of information contained in the input corpus. We can see in Figure 1.2 the typical steps of a NLP system.

Natural Language Processing is used today for several practical applications.

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

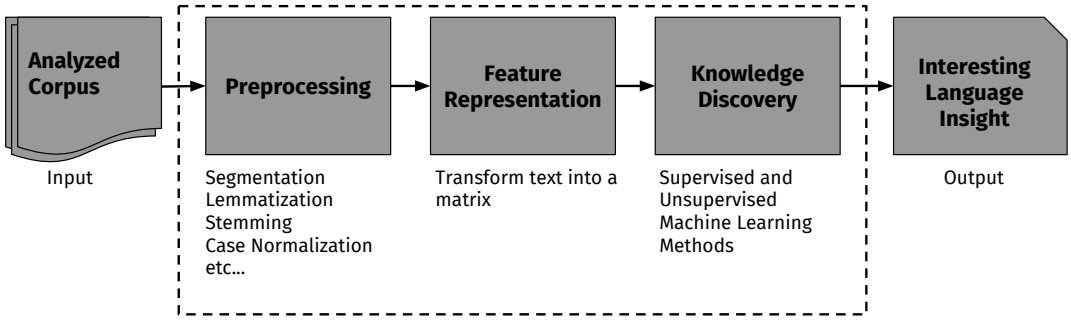


Figure 1.2: Typical steps of Natural Language Processing applications.

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 modelisation and two in terms of NLP applications. Specifically, the contributions that we propose in this work are the following:

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

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

1.2.1 Modeling linguistic features

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

allow us to properly design and build a linguistic resource containing heterogeneous descriptions of the textual units[13] adapted to solve NLP tasks.

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

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

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

1.2.2 Leveraging the network to find semantic relatedness

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

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

heterogeneous feature can help us to decide which is the most appropriate according to a NLP system objective.

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

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

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

1.2.3 Combining features and dealing with sparsity

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

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

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

Contribution Addressing the sparsity of textual data is not an easy task and often involves complex procedures and loss of information. To alleviate this issue, we propose the application of multimedia analysis fusion techniques to solve NLP semantic tasks. The fusion methods we employ comprise a set of methods to combine (or fuse) different types of features into a single unique representation. While combining attributes we also enrich them by leveraging the complementary information they carry individually. Furthermore, we address the challenge of data sparsity by transferring unseen relations from one feature space to another, that is, we obtain a denser similarity space by joining together both feature spaces. The experiments we carry out, in word sense induction and disambiguation and named entity recognition, show the pertinence of our approach. Specifically, we try different fusion techniques as well as several fusion configurations to improve the tasks' performance compared with using representations independently. Additionally, we study to what extent each type of fusion employed affects the performance of the tasks we evaluate.

1.3 Structure of the Dissertation

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

Chapter 2 This chapter contains the theoretical background on the concepts discussed in this thesis. At the same time, we present the state of the art on the techniques that are relevant to our work. Specifically, we discuss the basics on text representation and how they are all related together by the distributional hypothesis. We then introduce the two main types of mathematical entities to manipulate text in a computer: vector-space models and graph-based models. Given our choice to work with graphs, we continue this path and introduce their basic concepts and move onto how graphs are employed to hold textual data as well as some prominent NLP applications in the literature. Finally, we detail the machine learning methods we employ

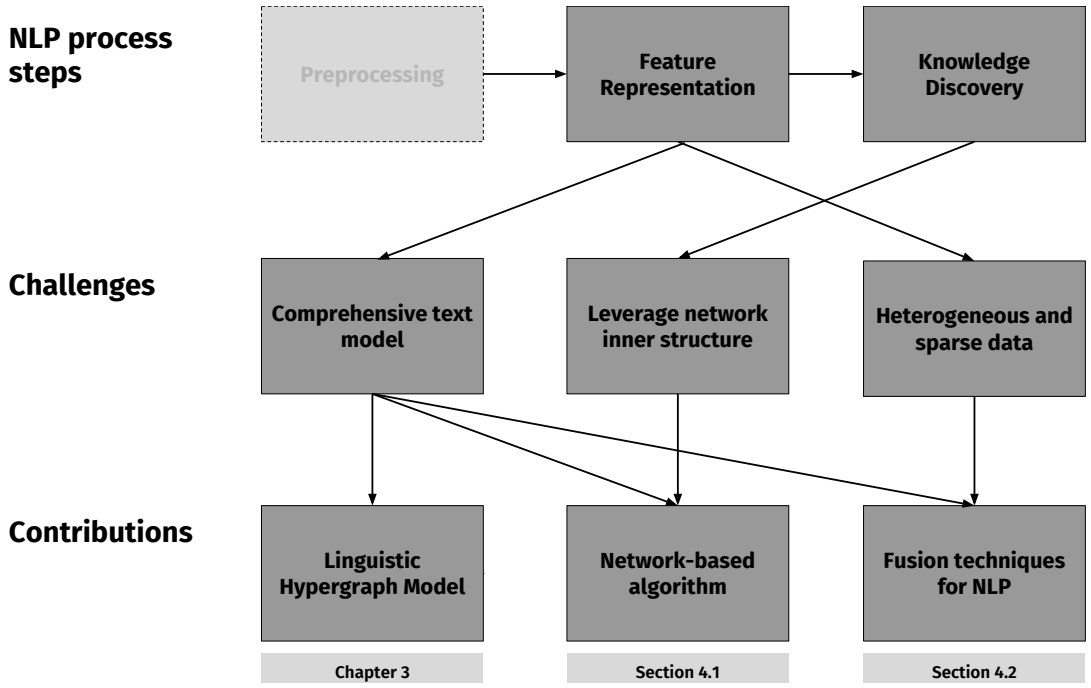


Figure 1.3: Block diagram of the NLP steps of interest, the challenges we address, and the contributions we propose.

in certain as part of our methodology.

Chapter 3 This As our first contribution, we present and define a novel structure to hold language information: the hypergraph linguistic model. We discuss its characteristics and the intuitions behind its conception: the hypergraph choice,the role of nodes and edges, the type of features stored, the advantages it represents in terms of accessing and manipulating the data.

This chapter describes the process of parsing and storing an English Wikipedia Syntactical Dump. We use open source tools on freely available data to generate a software that is able to extract lexical and syntactic data from a given corpus.We thus create and publicly release a syntactically annotated version of the English Wikipedia containing often-neglected information, stored in a format that facilitates its manipulation.

Chapter 4 In this chapter, we present an algorithm that exploits the structure of the network, i.e., the connections between nodes. Namely, we derive senses from a test corpus by finding related words connected to a single main node. We test the linguistic and lexical features and discuss about its qualities. Our results improve on the performance of similar propositions from the literature. We study the combination of

features using multimedia fusion techniques in the next chapter.

We explore the application of multimedia fusion techniques using linguistic features to solve NLP tasks. Briefly, fusion techniques may consist on trivially concatenating feature columns or a combination that aims to transfer relatedness information from one representation to a second one. 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 representation kind to each sense and class in each task respectively.

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

Chapter 2

Background

The main topic of this dissertation is language representation through a heterogeneous linguistic network and its applications on NLP. Our objective is twofold: first, to conceive a model able to handle multiple textual-data features extracted from. Second, to use this model to solve NLP tasks efficiently and making explicit use of the features diversity. Furthermore, we aim to leverage open-source corpus to complement in-domain datasets. To achieve our goals, we base our model, and proposed applications, on five main concepts: linguistic representations, the distributional hypothesis, network analysis (and related methods), multimedia fusion techniques, and lastly, machine learning methods, namely supervised and unsupervised approaches. these techniques are used as the tools to fit learning models over our data. Indeed, to accomplish the second goal, we propose solutions to two well-known natural language processing tasks: word sense induction and disambiguation and named entity recognition.

WSI is usually solved using vectorial representations for each word. However, since the goal of this application (and that of our third contribution) is to validate in

Preprocessing According to the final application of the system, we deal with "irregularities" present within the text, so that we can extract more informative features from it. Conventional techniques include conflating (or blend) words according to their stems (stemming), conflating words with respect to their lemmas (lemmatization), removing functional words (articles, determinants, etc.), case normalization, replacing numeric tokens with strings, among others.

Feature Representation In data analysis we often hear the saying "Garbage In, Garbage Out". Indeed, knowledge discovery (i.e., machine learning methods) can only do so much with the input data they are given. This step is then crucial for any NLP or data mining system. During this step we transform each textual unit from the input corpus into a numerical representation (i.e., a matrix) that is compatible with the machine learning methods used in the following step.

Knowledge Discovery Several NLP tasks can be seen, directly or indirectly, as tasks assigning labels to words [Collobert 2011]. In that sense, machine learning methods are used to solve this kind of problems and thus are naturally used to solve most NLP problems nowadays. Either supervised learning (leveraging annotated corpora) or unsupervised learning (finding related elements within the text, without any extra information), the goal is always to find interesting insights from the input corpus, towards a general language understanding.

Abstract. *This chapter goes into detail about the five theoretical axes of this dissertation. First, we present language representations and specifically the three different types that interest us: lexical, syntactic and semantic. Secondly, we cover how we can infer relatedness among words based on the neighbors of each word. This intuition is expressed by the distributional hypothesis. Thirdly, we explain how we can join words together through their common features by means of a graph-like structure. This structure is known as a language network. Fourth, we detail the techniques used in the multimedia literature to combine (or fuse) features coming from different sources and improve the performance of a knowledge-discovering system. In our work, instead of combining different multimedia sources, we combine diverse textual representations. Finally, we explain the (supervised and unsupervised) methods we employ to solve the NLP tasks at hand.*

Contents

2.1 Linguistic Representation Features

2.1.1 Lexical Representations

2.1.2 Syntactic Representations

2.1.3 Semantic Representations

2.2 Distributional Hypothesis

A simple and effective method of determining the nature of any textual unit (either words or larger quantities of text), without any other external information (human annotations), is to leverage the distributional hypothesis. Briefly, it states that a word meaning is determined by its own context. The most basic form of co-occurrence is determined by neighbor words that appear before or after a word of interest.

The propositions we present in this thesis are greatly based on the analysis of a word context as a source of valuable information to discover its nature. This context-analysis insight was introduced by [Harris 1954] and states that words that occur in similar contexts tend to have similar meanings. In a more light-hearted phrasing, [Firth 1957] formulated it as "You shall know a word by the company it keeps!".

The distributional hypothesis is popularly exploited in the NLP/Text mining domain by using the vector space model. In this model a word is represented by a vector in which the elements are derived from the occurrences of the word in various contexts, such as windows of words (n-words to the right and to the left), syntactical positions or grammatical dependencies, among other contexts. In our research we chose not to use vectors directly but instead use graphs (and hypergraphs) as an intuitive way to model the context of words in a corpus. Graphs have the capability to naturally encode the meaning and structure of a text by connecting language units (words in our case) through various relationships. Indeed, recent years have seen a rise in natural language methods based on graph-theoretical frameworks [Mihalcea 2011]. Specifically, in this work we represent words as vertices and the edges that link these nodes (words) correspond to various linguistic context properties shared among them.

2.3 Networks and Linguistic Networks

2.3.1 Network Representation

2.3.2 Types of Language Networks

2.4 Supervised and Unsupervised Methods

2.4.1 Logistic Regression

2.4.2 Perceptron and Structured Perceptron

2.5 Related Work

In this chapter, we present an overview of linguistic network's related work. We first categorize them by their objectives and then by the data contained within. Next, 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 introduce in the following chapter.

2.5.1 Types of Linguistic Networks

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

In the following paragraphs we introduce the general categories of LNs 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] defines broader 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: semantic, lexical co-occurrence, syntactic co-occurrence and heterogeneous networks. The following sections will elucidate what each kind of network represent, we will mention works that employ this kind of networks and also list the main methodology differences that variate from one approach to another.

Semantic Networks A Semantic Network (SN) relates words, or concepts, according to their meaning. 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 synonymmy-antnonimy, hypernym-hyponym, holonym-meronym. However, other semantic similarities can be defined. The edges are usually not weighted, although in some cases certain graph similarity measures may be used.

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

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

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

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

Other semantic tasks can also be solved using a SN. For example, Entity linking [Moro 2014]. In their work, they leverage the BabelNet LKB to jointly disambiguate and link polysemous words to Wikipedia articles.

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.

In [Matuschek 3 05] they propose a technique that aligns SNs, i.e., they link senses from two different networks. This task is called word sense alignment. Several SNs are used (Wordnet, Wikipedia, Wiktionary, etc.) thus nodes can represent synsets, articles, or concepts. The links may depict semantic relations or may be links joining two concepts or pages together. Their approach aims to find the shortest path between nodes of any two given SNs while leveraging already existing links between equal concepts found in both SNs.

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.

Lexical Co-occurrence Networks Most co-occurrence based intuitions in NLP have their origin in the distributional hypothesis. An effective way to represent word co-occurrences is by means of a graph structure. Indeed, this kind of graphs are the central column of a Lexical Co-occurrence Network (LCN). In these structures, nodes represent words and edges indicate co-occurrence between them, i.e., two words appear together in the same context. A context can vary from a couple of words (before or after a given word) 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.

To solve a task in a truly unsupervised way, researchers generally use this kind of networks instead of LKBs. It is then natural that word sense disambiguation approaches leverage lexical co-occurrence networks, and in return, the distributional hypothesis, to automatically discover senses for a given target word. That is why WSI methods [Véronis 2004, Klapaftis 2007, Navigli 2010, Klapaftis 2008, Di Marco 2011, Jurgens 2011] are tightly related to LCNs. The cited works use a LCN as described before while other works such as [Navigli 2007, Qian 4 08] represent the co-occurrence by means of a hypergraph scheme. In short, a hypergraph structure is a graph generalization where an edge (called hyperedge) can link more than two vertices per edge and thus it is able to provide a more complete description of the interaction between several nodes [Estrada 2005].

In their paper, given an input document with several contexts for each target word, they first group together the contexts via a topic-modeling technique. Thus, each context is assigned a particular group (in this case, a topic). Secondly, a hypergraph is built where the vertices represent contexts and the hyperedges link two nodes together if they share the same topic. Thirdly, the hypergraph is clustered and the words of each context (of each node) are used to build vectorial representations

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

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

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

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

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

As with semantic networks, not only WSD or WSI can be solved with LCNs. Finding semantic associated terms in a corpus is a critical step in several NLP systems. This task is solved in the system proposed by [Liu 2011]. They also use a LCN although instead of a co-occurrence graph, they also employ a co-occurrence hypergraph, where nodes represent words and edges describe co-occurrence at the paragraph level. In this work, they use such structure to find related terms in a given

corpus. In order to do it, they mine the hypergraph as in a frequent itemsets problem, where the items are the words from a text. The method consists in first finding similar itemsets by means of measuring similarity between nodes. Once the 2-itemsets are found, they induce a graph from the original hypergraph by drawing edges between nodes that have a similarity superior to an arbitrary threshold. Lastly, in order to find k-itemsets ($k > 2$), the find either complete or connected components in the induced graph.

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

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

Syntactic Co-occurrence Networks A Syntactic Co-occurrence Network (SCN) is very similar to a LCN in the sense that both exploit the distributional hypothesis. Nonetheless, SCNs go further by leveraging syntactic information extracted from the text. There are two main types of syntactic information both represented as tree structures: constituency-based parse trees and dependency-based parse trees. Briefly, the former structure splits a phrase into several sub-phrases. In this way we can get a glimpse of the role of each word inside a phrase. The latter tells us about the relationships existing between words in the phrase. SCNs employ, most of the time, dependency trees to create a graph that relates words according to their syntactic relations. In the case of [Hope 3 06], a graph is built using syntactic dependencies. It is used to perform WSI using a very similar approach as those systems using LCNs.

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 reserch avenue to explore.

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].

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.

2.5.2 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 algorithms used to solve practical NLP tasks. In this section we will cover the details of four different types of graph algorithms.

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

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

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

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

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

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

In this subsection we also consider subgraph extracting techniques which are exploited to find separated groups of words and thus induce senses. In this context we found the approaches of [Véronis 2004, Silberer 2010]. These systems make use of both the Minimum and Maximum Spanning Trees algorithms (MinST and MaxST, respectively) as a final step to disambiguate a target word given its context. Meanwhile,

both [Liu 2011, Qian 4 08] use the Hypergraph Normalized Cut (HCT) approach, a hypergraph clustering method based on minimum cuts, to induce senses.

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

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

2.5.3 State of the Art Discussion

We have covered the network attributes of several approaches on semantic related NLP tasks. A summary of these strategies is shown in Table 2.1. In this section we will 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

Table 2.1: Survey summary table.

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

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 consituency 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 cover some of the limitations stated above.

Chapter 3

Hypergraph Linguistic Model

Abstract. *We present in this chapter our model proposition to stock linguistic data. This structure, based on the concept of hypergraphs, holds heterogeneous textual features, improving on the limitations of common solutions of the state of the art. At the same time, our network allows for fast access to words and their properties from different points of view. Finally, we also discuss our motivations and the scope and limitations of the network.*

Contents

Building upon previous linguistic representations [Klapaftis 2007, Liu 2011, Qian 4 08], our model is also based on the use of a hypergraph. Their single most important difference with regular graphs, being able to relate more than two vertices at the same type, allows for a better characterization of interactions within a set of individual elements (in our case, words) [Heintz 2014]. Indeed, our hypergraph modelization initially integrates four types of relations between tokens: sentence co-occurrence, part-of-speech tags, words' constituents data and dependency relations in a single linguistic structure. These relationships were chosen because its is relatively easy to obtain them for high-resource languages. These features can be seen as building blocks for NLP models. Extracting deeper language features would implicate relying even more on general domain systems. In any case, our goal is to arrive to more complex annotations (e.g., named entities) from the selected features and relations.

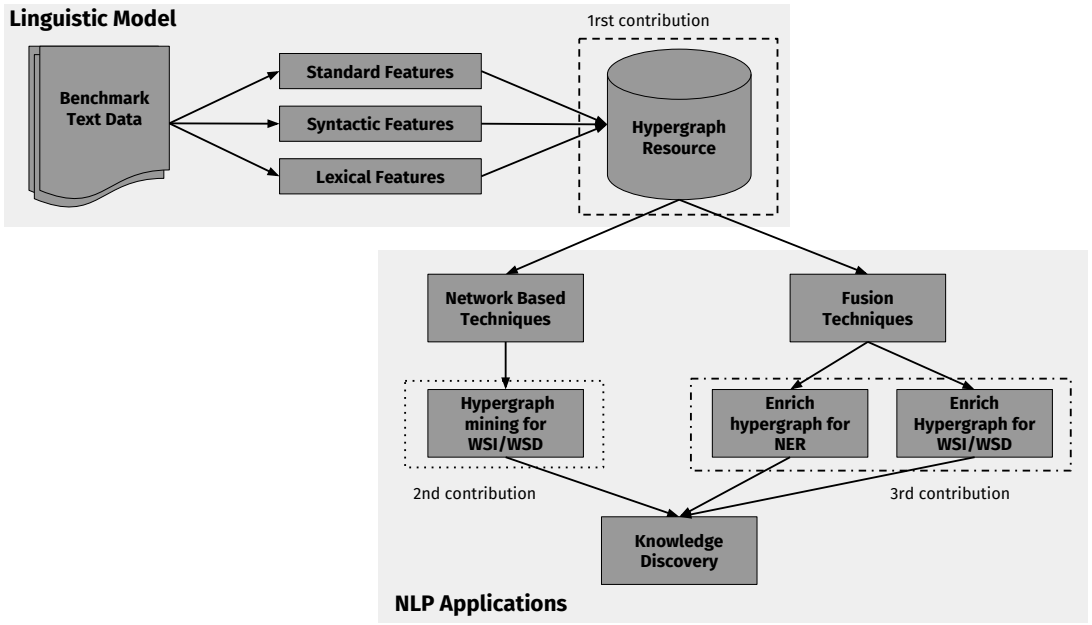


Figure 3.1: Modular description of this dissertation

3.1 Characteristics of our proposition

In our model we group words together according to these features, into a hypergraph schema. Formally, a hypergraph [Berge 1985] is a graph generalization that allows more than two vertices to be linked by a single edge. Let V denote a finite set of objects, and let E (the hyperarcs or hyperedges) be a group of subsets of V such that $V = \cup_{e_j \in E} e_j$. We call $\mathcal{H} = (V, E)$ a hypergraph with the vertex set V and the hyperedge set E. A hyperedge containing just two nodes is a simple graph edge. A hyperedge e is said to be *incident* with a vertex v when $v \in e$.

In our case, the set of tokens 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. Each hyperedge may be one of three types: noun phrase[1] constituents (*CONST*), dependency relations (*DEPENDENCY*), or sentence context (*SENTENCE*). We consider that a token v belongs to a hyperedge e_j of type NP or SENTENCE if the token appears in the same noun phrase or in the same sentence. A token v belongs to a hyperedge of type DEPENDENCY if it is the dependent of a certain dependency relation coupled with its corresponding head (or governor). 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.

[1] In this work we consider only noun phrases (NPs). Still, we can easily add other type of phrase chunks.

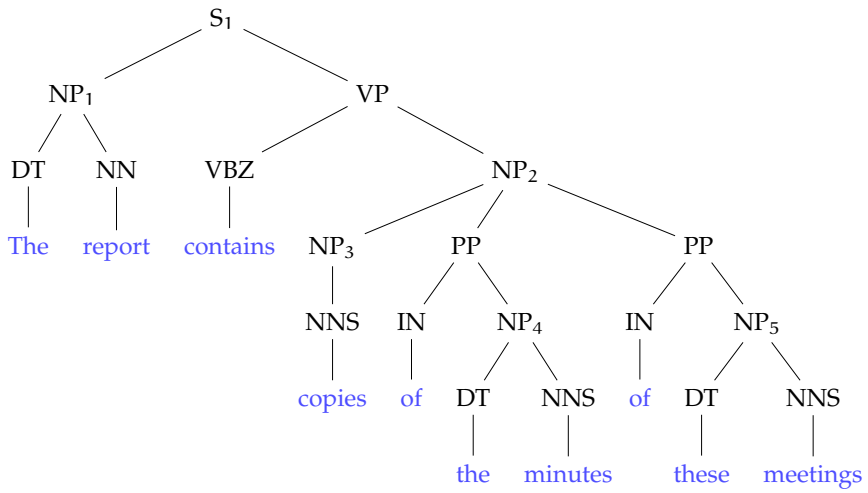


Figure 3.2: Constituency-based tree of the phrase *The report contains copies of the minutes of these meetings.*

Table 3.1: Dependency relations of the example phrase.

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

We illustrate our hypergraph incidence matrix 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.

The constituency tree of the example phrase is shown in Figure 3.4. The sentence, as well as each noun phrase (NP) node is identified by a number. We can observe that this phrase is composed by five noun phrases (NP) and one verb phrase. Meanwhile, some of the NPs are formed by other kind of phrases, depending on the grammar production rule used to build each one of them. 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.

The dependencies of the example phrase are shown in Table 3.1. They indicate the syntactic relation between the governor of a phrase and dependent. In these relations' examples, the head is the first token to appear followed by the dependent word.

From both of these types of information we can build a hypergraph representation

Table 3.2: Incidence matrix of the example phrase hypergraph modelization.

		CONSTITUENT			**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
	contains						1
	copies		1	1		1	1
	minutes		1				1
	meetings		1				1

as stated before. The incidence matrix is illustrated in Table 3.2. For brevity, we only show nouns as well as only the first three noun phrases and the nominal subject (*nsubj*) and direct object (*dobj*) dependency relations. Looking at the table, we can infer that the word *copies* appears in two hyperedges of type CONSTITUENT: first in NP_2, which is built from a NP 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*.

For the sentence hyperedges, we see that the token *copies* appeared in the same sentence S_1 as the other four noun words.

With this short example we show the intuitive way in which we store three different kinds of relations: lexical co-occurrence, dependency co-occurrence and sentence (at chunk level) co-occurrence.

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

3.2 Fusion Techniques

3.2.1 Early Fusion

3.2.2 Late Fusion

3.2.3 Cross Fusion

3.3 Wikipedia Syntactic Dump

In order to have a working dataset, we first built a software that process and parse any input corpus. We describe its properties, its inputs, the information extracted, as well as the output generated by the software.

3.3.1 Introduction and Related Work

Today, the broad reach of Wikipedia in Text Mining (TM) and Natural Language Processing (NLP) research is indisputable. Several recent approaches and tools have been conducted based on the explicit and implicit knowledge contained in it. Certainly, Wikipedia provides a common ground for researchers and developers to test and compare their results.

Wikipedia 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 (PoS) 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[2] datasets library. In [Flickinger 2010], they perform a deep parse analysis is performed to provide detailed syntactic and semantic information. The authors leverage a previously manually annotated portion of the English Wikipedia. They extract a grammar from this portion and also train a statistical model to automatically parse the rest of Wikipedia. Even though the parse offered is deep and rich in details, the annotation labels, as well as the corpus output format, may not be convenient and easy to use because of its complexity and particular nature. [Schenkel 2007] released a purely semantic XML parse that links WordNet concepts to Wikipedia pages. They focus greatly on cleaning and pre-treating Wikipedia. In this paper we do not focus as much into the cleaning of Wikipedia as already available tools can solve the task quite well for non-specific needs. Finally, there are certain Wikipedia dumps that offer the raw cleaned text without any extra subsequent parsing or analysis. Such is the case of the corpus made available by [Shaoul 2010]. This corpus makes use of the *WikiExtractor* script [Giuseppe Attardi 2015] to clean the Wikipedia dump.

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

SAEWD uses widely known English language parsing tools, namely those included in the Stanford CoreNLP suite. Aside from being accessible and regularly maintained, it provides a common set of labels (Universal Dependencies[3]) used by

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

[3] 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 3.3.3). Constituent's paths, and hence chunk's production rules, have been proved useful as a complement feature to classic text representations [Sagae 9 10, Bergsma 2012, Massung 2013].

As a second contribution, 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 [Rada Mihalcea and Dragomir Radev 2011]. That is why, in addition to SAEWD, we also propose, as a proof of concept, a hypergraph representation that stores certain information found in a SAEWD in a practical way that allows for fast and effortless data extraction. This hypergraph can be indeed considered as a Linguistic Network [Choudhury 2009b]. It aims to facilitate the implementation of graph-based approaches by allowing researchers to jump directly into the algorithm development stage. We use a sub-sample of the Wikipedia corpus consisting of articles related to Natural Language Processing and Text Mining.

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

3.3.2 Construction of SAEWD

The three main steps we followed to build SAEWD are presented in Figure 3.3. Briefly, we have one input, which is the Wikipedia dump and one output which is the parsed snapshot. In the following we provide a detailed description of each of the process.

We begin the construction of the parsed corpus with the Wikipedia dump XML file obtained from the Wikipedia database[4] from early November 2014. This dump contains around 4.7 million article pages[5]. As shown in Figure 3.3, we apply the following processing steps in order to yield the final parsed version.

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

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

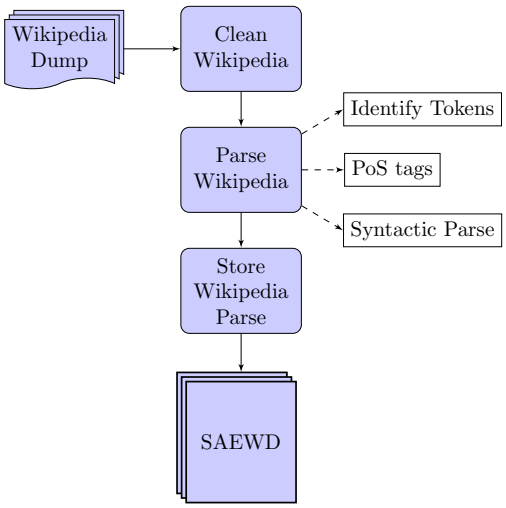


Figure 3.3: The tree steps we took to build SAEWD.

3.3.2.1 Cleaning Wikipedia

First, we discard Wikipedia's tables, references and lists, markup annotations and HTML tags with the *WikiExtractor* [Giuseppe Attardi 2015] script. We used this tool to clean and split the content of the original XML file into 429 folders each one containing 100 files of approximately 300 kB. These files contain a certain number of complete Wikipedia articles which is automatically determined by WikiExtractor according to the maximum possible size assigned for each file, 300 kB in our case, thus the number of articles in each file may vary. We decided to use numerous files as well as a small size to easily read their content into memory while parsing. Having multiple small files also makes it easier to handle the multi-threading aspect of our parsing tool. We kept WikiExtractor's original folder nomenclature which assigns to each one of them a sequence of letters sorted lexicographically[6]. 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.

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

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

3.3.2.2 Parsing Wikipedia

Next, once the Wikipedia dump had been cleaned, we use the Stanford CoreNLP[7] [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 3.3.3.

- PoS tagging: We obtain the grammatical category of each word, i.e., the part-of-speech tag, using the CoreNLP default tagger, the *left3words* PoS tagging model.
- Dependency parse: this attribute consists on an extracted tree that describes the types of grammatical relations between words, i.e., the dependency-based parse tree. The analysis was performed with the Stanford's *Shift-Reduce* parser. As information representation, we use the basic dependencies scheme, as we wanted to include each one of the possible dependency relations without any collapsing between them.
- Constituents parse: the output of this analysis is a rooted tree that represents the syntactic structure of a phrase. This tree is commonly known as the constituency-based parse tree. For each word, we store its complete path in the constituency tree. Specifically, we keep all the nodes of a word's own branch from the root to the word itself. We employ the Stanford Shift-Reduce parser. This path is transformed into a single line and included in SAEWD.

Finally, once the parsing process is complete, the parsed files are stored into individual files and thus there are as much parsed files as input Wikipedia cleaned files. The parsed files keep their original name plus the `parsed` extension, e.g., `wiki_00.parsed`. The structure within the files is described in Section 3.3.3.2. After parsing, we found the statistics shown in Table 3.3.

3.3.3 SAEWD Description

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

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

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

FILENAME wiki_00.parsed

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

3.3.3.1 Constituency parse storage in detail

We will use an example to better explain the storage of the constituency-based parse tree. In Figure 3.4 we can see the constituency parse of the phrase *A great brigand becomes a ruler of a Nation*. On the bottom of the figure, we observe the constituent's path (or branch), of the words *brigand* and *Nation*. As in any tree structure, each leaf node has a defined path from the root node to itself. In this example, the leaf containing the noun *brigand* follows the bottom-up path *NP22→S97*. *Brigand's* parent node is a Noun Phrase (NP) node which in turn comes from the root of the tree, the Sentence node *S*. We assign to each phrase chunk an identifier (22 and 97 in this case) in order to distinguish them according to their building elements as specified by the grammar rule used. In other words, a phrase chunk, e.g., a NP, a Verbal Phrase (VP), a Prepositional Phrase (PP), or other chunk defined by the grammar in CoreNLP, may be built from different types of PoS tags. Thus, again from Figure 3.4, 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

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.

figures produced by the `hashCode`[8] Java method.

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

3.3.3.2 Annotation scheme

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

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

- Metadata tags:
  1. FILENAME: indicates the original file used to extract the current parse,
  2. %%#PAGE: denotes a new Wikipedia article, as well as its title,
  3. %%#SEN: marks the beginning of a new sentence. It is followed by two integers: (1) the number of the current sentence, and (2), the number of tokens in the sentence.
- Parse columns for each token:
  1. Token: the token itself,
  2. Lemma: the token the canonical form,

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

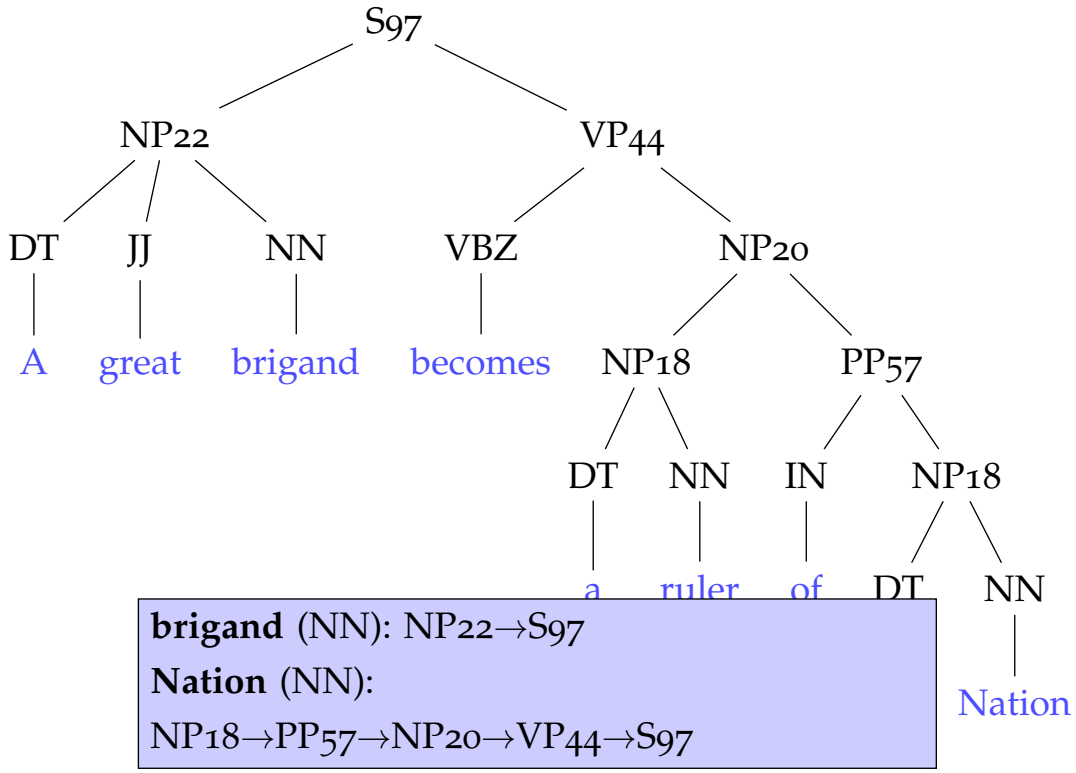


Figure 3.4: Constituency tree for the phrase *A great brigand becomes a ruler of a Nation.*

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.

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

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

Chapter 4

Linguistic Model Applications: NLP Semantic Tasks

Abstract. *Word Sense Induction and Disambiguation (WSD/WSI) is an elementary semantic NLP task useful to build larger systems. In this chapter, we leverage the network presented in the previous chapter and propose an algorithm that takes into account the structure of our model. Based on the real-world property of the network, we induce senses by grouping words that share a similar sense. We then assign these senses to target words. We show that our method improves on similar network-based methods.*

Contents

4.1 Word Sense Induction and Disambiguation

4.1.1 Introduction

In this chapter we employ the model introduced before to propose a solution to both Word Sense Disambiguation (WSD) and Induction (WSI) tasks. We begin by giving a description and a succinct state of the art for both tasks. Specifically, we identify what type of language network (from those defined in Section**??**) is used and how. We then present the characteristics of our proposed method. Finally we describe and discuss the obtained results.

4.1.2 Related Work

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

Word Sense Disambiguation In the literature [Sinha 2007, Tsatsaronis 7 01, Navigli 2007, Agirre 2009, Klapaftis 2010, Silberer 2010, Moro 2014], WSD is usually solved using semantic networks (SN). Specifically, Wordnet and to some extent BabelNet or Wikipedia (considering an article as a single concept).

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

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

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

The cited methods differences boil down to three aspects. Firstly, the type of semantic relation implied by the edges of the network. Secondly, the function used to assign a weight to each edge, as well as its meaning. On the other hand, a common

characteristic of these approaches is that, as said before, they all rely on semantic networks and thus are constrained by the limitations of such resources, mainly their static content and human annotated nature.

We find that the use of a lexical knowledge base, such as Wordnet, is pervasive in this task. Indeed, new resources, such as BabelNet, solves to some extent the fixed (no new senses are included automatically) nature of this type of resources by leveraging the ever-evolving knowledge of Wikipedia.

Word Sense Induction We believe that in order to solve WSD in a truly end-to-end unsupervised way, one would need to automatically find a list of senses for a word without the help of pre-built semantic networks. In that sense, researchers generally use lexical co-occurrence networks, or LCNs, and thus leverage the distributional hypothesis, to automatically discover senses for a given target word. Therefore, WSI graph-based methods [Véronis 2004, Klapaftis 2007, Navigli 2010, Klapaftis 2008, Di Marco 2011, Jurgens 2011] are tightly related to LCNs.

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

1. Build a LCN, assigning tokens as nodes and establishing edges between them if they co-occur in a given context (usually if they both appear in the same sentence, paragraph or fixed window of words).
2. Determine the weights for each edge according to a frequency metric.
3. Apply a graph clustering algorithm. Each cluster found will represent a sense of the polysemous target word.
4. Match target word instances with the clusters found (the senses) by using the word context. Specifically, assign a sense to each instance by looking at the tokens in the context. This step is actually the word sense disambiguation task.

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

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

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

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

4.1.3 Approach

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

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

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

The second approach, *UOY*, described in [Klapaftis 2007], relies itself on the

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

In our method, we generate a network for each tw and consider that the high-degree nodes inside this network may represent a tw sense. In Algorithm 1 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 they may exist inside any of the three different types of hyperedges defined (sentence, noun phrase or dependency contexts). If any hyperedge is repeated through the corpus, we increment a counter and keep the number of apparitions instead of adding redundant columns to the hypergraph incidence matrix.

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

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

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

given hyperedge. Thus, we have as many nodes in W as we had hyperedges in G_H.

We compute the Jaccard index between each node $n_{i,j} \in U$ as $Jaccard(i,j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$ in order to build a $|U| \times |U|$ similarity matrix S_{tw} (line 4). We induce from S_{tw} a new filtered hypergraph incidence matrix F_{tw} (line 5), which contains word nodes as rows and columns as hyperedges. Each of these hyperedges represent a set of words that are deemed similar between them according to their Jaccard index value, which must be equal or higher than an assigned threshold th_1 .

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

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

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

The Jaccard similarity measure allows us to easily determine the neighbors of each node in the current bipartite hypergraph representation. As each node is joined to a sentence or dependency node, calculating the Jaccard similarity amounts to determining the level of co-occurrence between each word according to a specific type of hyperedge (represented as a node from the other graph partition) while taking into account the total number of hyperedges the words participate in. We differentiate specifically from the previously described method, UOY, in that in the case of that system, the weighting of the hyperedges is done by computating 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 tak-

ing the value of the threshold directly, without having to change it according to the characteristics of the data.

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

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

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

4.1.4 Experiments

4.1.4.1 Datasets

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

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

4.1.4.2 Implementation

The objective of this experiment is to show the complementarity of both lexical and syntactic co-occurrence information while solving WSI and WSD tasks while using

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

Input: A set $tw\_set = \{tw_1, tw_2, ..., tw_n\}$ of target words
Input: A background linguistic network G_H
Input: Filtering thresholds th_1, th_2
Output: A set SoS_{tw} of senses for each target word

```
foreach target word tw in tw_set do
    G_tw = extract_subgraph(G_H, tw);
    B_tw = induce_bipartite_graph(G_tw);
    S_tw = sim_matrix(B_tw);
    F_tw = induce_hypergraph(S_tw, th1);
    candidate_hubs = sort(degree(F_tw))[:100];
    SoS_tw = [ ];
    foreach candidate_hub in candidate_hubs do
        candidate_hyperedges = get_hyperedges(candidate_hub, F_tw);
        candidate_avg_jaccard = 0;
        foreach hyperedge in candidate_hyperedges do
            candidate_avg_jaccard += get_avg_jaccard(hyperedge);
        end
        if candidate_jaccard > th2 then
            SoS_tw.add(get_words(candidate_hyperedges));
            F_tw = F_tw \ candidate_hyperedges;
    end
    return SoS_tw
end
```

the method described in the previous subsection. To that end we build two independent systems: **LEX**, which uses exclusively lexical co-occurrence hyperedges, and **DEP**, which employs only syntactic dependency hyperedges. Each type of hyperedge has its own network characteristics as mentioned before. Sentence hyperedges tend to have a much smaller number of words than those of the dependency category. This make sense as sentences usually contain less than 30 words, meanwhile a dependency hyperedge may contain up to hundreds of words (several words may share the same dependency relation). Taking this into consideration we set different threshold values for **LEX** and for **DEP**. First, we consider only the top 100 nodes as candidate sense hubs. Secondly, we do not set the thresholds' values directly but instead we set up a percentile value for the Jaccard similarity (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.

4.1.5 Results

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

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

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

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

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

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

Moving onto the supervised results for Semeval-2007, in Table 4.2 we show the results obtained concerning the Recall performance metric. In this table we include the competing system *I2R*, based on an Information Bottleneck based clustering algorithm, which obtains the best results according to all the words and nouns. Both

our systems **DEP** and **LEX** beat the baseline of assigning the most frequent sense to an instance (*MFS*). More interestingly, *DEP* was able to beat the MFS verb baseline, something that was not achieved during the competition. As was the case before, our systems beat UoY(2007).

As a way of determining where does both systems complement each other, in Figure 4.1 and 4.2 we show the unsupervised F-Score value for nouns and verbs respectively. We can see that, as the previous result tables indicated, **DEP** did better overall. Nonetheless, and what is most interesting in these figures, is that there are certain words, nouns and verbs, that obtain better scores using **LEX** instead of **DEP** and vice versa. For example, the nouns *area*, *future*, and *state* are better treated by **SEN**, according to this measure, even if by a small margin. On the other hand, with respect to the verbs, the differences between performance are more important. The system **SEN** does better while finding senses and assigning them to the verbs *avoid*, *fix*, and *work*. This information will be useful during the design of hybridization techniques between feature of our hypergraph structure.

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

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

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

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

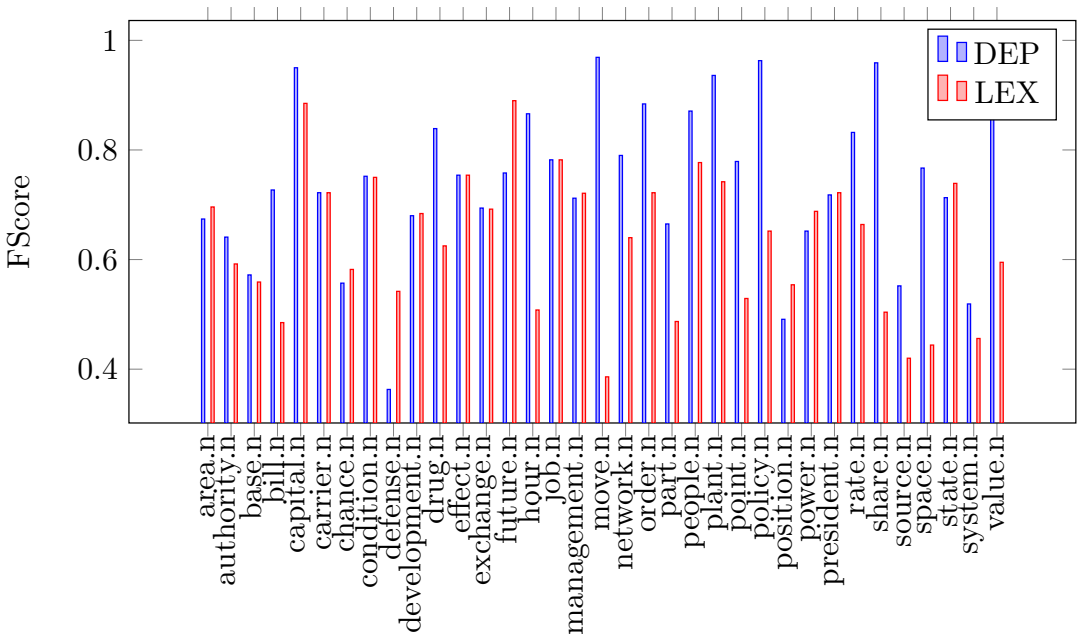


Figure 4.1: Unsupervised F-Score results for the nouns of the Semeval 2007 test set

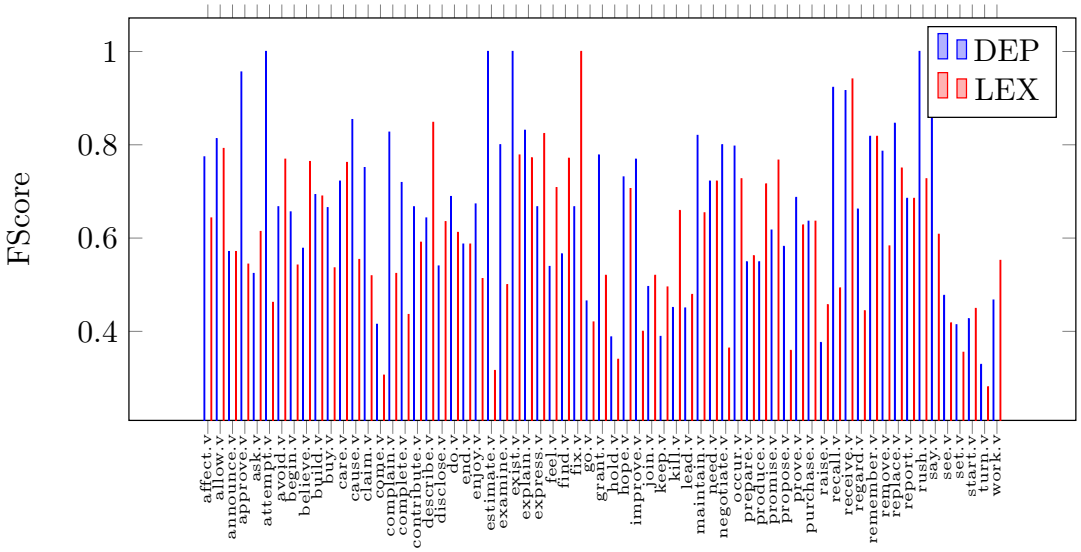


Figure 4.2: Unsupervised F-Score results for the verbs of the Semeval 2007 test set

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

primarily assigned to a single cluster.

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

Concerning the supervised evaluation, it follows explicitly that of Semeval-2007, using recall as the main performance measure. For more details about these measures, we refer the reader to the more comprehensive descriptions found in [Manandhar o 07, Van de Cruys 2011, Pedersen 2010].

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

The second unsupervised measure, Paired F-Score, can be seen in Table 4.4. In this case both systems presented performed better than the random baseline. Any system presented was able to beat the MFS baseline. We note that **DEP** does much better compared to **LEX** concerning verbs, namely 58% vs. 28% F-Score. Still, the results are low considering the best results of the competition, 63% from Duluth, although again, it generates a number of senses very similar to the MFS baseline. Both our systems induce a considerable amount of clusters while keeping a descent F-Score.

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

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

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

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

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

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

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

The F-score of both systems is shown in Figure **??**. Indeed, in our experiment, the dependency based model *dep* preformed better than *lex* using classic lexical co-occurrence. Even more, in Figure **??** we appreciate that even while using different threshold values, we achieve, in general, better recall and precision by using syntactic dependencies. It must be noted that this particular Semeval task was dominated by the most frequent sense with an F-score of 80.7, assigning an average of one sense

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

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

per target word. Our solutions assign an average of 1.257 and 1.200 for *lex* and *dep* respectively.

4.1.6 Discussion and Conclusion

4.2 Named Entity Recognition

Abstract. *We explore the use of well-known multi-modal fusion techniques to solve two prominent Natural Language Processing tasks. Specifically, we focus on solving Named Entity Recognition and Word Sense Induction and Disambiguation by applying feature-combination methods that have already shown their efficiency in the multi-media analysis domain. We present a series of experiments employing fusion techniques in order to combine textual linguistic features. Our results show that the combination of textual features indeed improves the performance compared to single feature and the trivial feature concatenation. Furthermore, we perform an extensive analysis on the importance of each feature importance with respect to the senses and classes discovered in WSI/WSD and NER, respectively.*

4.2.1 Introduction

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

or by training a supervised machine learning algorithm with large quantities of annotated text. The latter being the currently more popular solution to this task.

Word Sense Induction and Disambiguation entails two closely related tasks[1]. 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.

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

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

The main intuition of the present work is that word similarities may be found at different levels according to the type of features employed. In order to exploit these similarities, we look into multimedia fusion methods. In order to better perform an analysis task, these techniques combine multimodal representations, their corresponding similarities, or the decisions coming from models fitted with these features. In our experiments, we try to mutually complement independent representations by utilizing said fusion techniques to combine (or fuse) features in the hope of improving the performance of the tasks at hand, specially compared to the use of features

[1]Even though these tasks are closely related, they are independent from one another. Still, in this chapter we consider them to be a single one.

independently.

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

The rest of the chapter is organized as follows: in section 4.2.2, we go into further details about fusion techniques. We introduce the fusion operators that we use in our experiments in section 4.2.3. Then, in section 4.2.4 we show the effectiveness of the presented methods by testing them on NER and WSI/WSD and their respective datasets. Finally, in section 4.2.8 we present our conclusions and future directions to explore. A more general overview on fusion for multimedia analysis can be found in [Atrey 2010].

4.2.2 Background and Related Work

We first describe below the fusion techniques we use in our methodology as well as relevant use cases where they have been employed. Then, we focus exclusively on fusion techniques applied to NLP tasks, which is the general domain of the two tasks (NER and WSI/WSD) we focus on.

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

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

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

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 models trained each with an unique set of features [Clinchant 2011]. In this setting, decisions produced by each model are combined into a single final result set. The methods used to combine preliminary decisions usually involve one of two types: rule-based (where modalities are combined according to domain-specific knowledge) or linear fusion (e.g., weighting and then adding or multiplying both matrices together). This type of fusion is very close to the so-called ensemble methods in the machine learning literature. Late fusion combines both modalities in the same semantic space. In that sense, we may also combine modalities via an affinity representation instead of final decision sets. In other words, we can combine two modality matrices by means of their respective similarities. A final representation is then usually obtained by adding the weighted similarity matrices.

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 chose which algorithm is more adequate for each type of features.

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

1. Compute the corresponding similarity matrices for each type of feature.

2. Select only the k-nearest neighbor for each word within the lexical similarity matrix. These neighbors are to be used as lexical representatives to enrich the syntactical similarities.

3. Linearly combine both similarity matrices (lexical k-nearest lexical neighbors with the syntactical features) via a matrix product.

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 two 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 hybrid fusion by leveraging the combination of the fusion strategies described before.

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

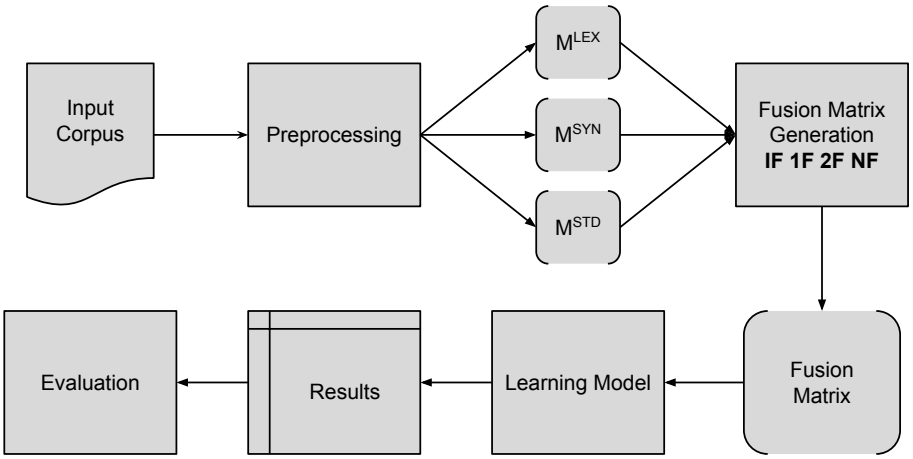
Our first goal is to assess the effectiveness of the classic fusion methods and then, as a second goal, to propose new combinations that yield better outcomes in terms of performance than the simpler approaches. The new combinations are found empirically. Nonetheless, as we will show, their effectiveness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

4.2.3 Applying Fusion Techniques

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

The experiments we carry on consist in generating fusion matrices that will serve as input to a learning algorithm in order to solve NER and WSI/WSD. These input feature matrices are based upon lexical, syntactical, or other types of representation. The procedure can be sen in Figure 4.3.

Figure 4.3: Steps followed on our experiments. First the corpus is preprocessed, then features are extracted from the text. A fusion matrix is generated, which in turn is used as input to a learning algorithm. Finally, the system yields its results and to be analyzed.



4.2.3.1 Fusion Strategies

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

Early Fusion

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

Weighted Early Fusion

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

Late Fusion

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

Cross fusion

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

Parameters A and B are arbitrary input matrices. They may initially represent, for example, the lexical (M^{LEX}) or syntactical based (M^{SYN}) features matrix, or their corresponding similarity matrices, S^{LEX} and S^{SYN}, respectively. In a broader sense, matrices A and B may represent any pair of valid[2] fusion matrices.

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

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

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

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

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

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

2. **First Degree Fusion**: we consider the three elementary fusion techniques by themselves (early fusion, late fusion, cross fusion) without any recombination. These experiments, as well as those from the previous level, serve as the baselines we set to surpass in order to show the efficacy of the rest of the fusion approaches. As an example, we may obtain a representation matrix by performing an early fusion between the lexical matrix and the syntactical features matrix: $E(M^{LEX}, M^{SYN})$. In this level we distinguish two types of cross fusion: Cross Early Fusion (XEF) and Cross Late Fusion (XLF). The first one combines a similarity matrix with a feature matrix: $X(S^{LEX}, M^{SYN})$. The second one joins a similarity matrix with a similarity matrix: $X(S^{SYN}, S^{LEX})$.

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

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

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

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

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

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

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

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

4.2.3.2 Feature Matrices

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

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

Syntactical Matrix (SYN) Based on the syntactic features used in [Levy 2014, 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 $p_1^m, \ldots, p_k^m$; a head h and its corresponding PoS tag p_h, we consider the context features $(m_1, p_{m_1}, lbl_1), \ldots, (m_k, p_{m_k}, lbl_k), (h, p_h, lbl\_inv_h)$. 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.

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

4.2.3.3 Learning Methods

We use supervised and unsupervised learning methods for NER and WSI/WSD respectively. On the one hand, for NER, as supervised algorithm, we use an averaged

structured perceptron [Collins 2002, Daume 2006] to determine the tags of the named entities. We considered Logistic Regression and linear SVM. We chose the perceptron because of its performance and the lower training time.

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

4.2.4 Experiments and Evaluation

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

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

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

4.2.5 Named Entity Recognition

4.2.5.1 Pre-processing

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

4.2.5.2 Features

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

4.2.5.3 Test Datasets

We work with three corpus coming from different domains:

(1) CoNLL-2003 (CONLL): This dataset was used in the language-independent named entity recognition CoNLL-2003 shared task [Sang 2003]. It contains selected news-wire articles from the Reuters Corpus. Each article is annotated manually. It is divided in three parts: training (*train*) and two testing sets (*testa* and *testb*). The training part contains 219,554 lines, while the test sets contain 55,044 and 50,350 lines, respectively. The task was evaluated on the *testb* file, as in the original task.

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

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

4.2.5.4 Evaluation Measures

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

4.2.5.5 Results

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

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

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

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

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

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

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

First Degree Fusion In Table 4.7 we present the First Degree fusion level. The best performance is obtained by trivially concatenating the representation matrices. This baseline proved to be the toughest result to beat. Late fusion does not perform well

in this setting, still, we see further on that by linearly combining weighted representation matrices, we can add information to an already strong representation. Finally, regarding the cross fusion techniques, cross early and late fusion, we see that they depend directly on the information contained in the similarity matrices. We note that, as is the case on single features, the combinations with matrix S^{STD} yield almost always the best results. While these fusion techniques by themselves may not offer the best results, we see below that by recombining them with other types of fusion we can improve the general performance of a representation.

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

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

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

4.2.6 Word Sense Induction and Disambiguation

Having learned the best fusion configuration from the previous task, in this experiments we set to test if the improvements achieved can be transfered into another NLP

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

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

task, namely Word Sensed Induction and Disambiguation (WSI/WSD).

4.2.6.1 Pre-processing

We simply remove stopwords and tokens with less than three letters.

4.2.6.2 Features

We use the same set of features from the previous task, with the exception of the standard NER features, that is, those represented by M^{STD}, as they are specifically designed to tackle NER.

4.2.6.3 Test Dataset

The WSI/WSD model is tested on the dataset of the SEMEVAL-2007 WSID task [Agirre 2007b]. The task was based on a set of 100 target words (65 nouns and 35 verbs), 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.

4.2.6.4 Evaluation Measures

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

4.2.6.5 Results

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

to the denser characteristic of the matrices, there is a larger quantity of true similar words that are useful to project information into another matrix, through cross fusion.

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

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

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

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

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

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

4.2.7 Feature Importance Analysis

4.2.7.1 WSI/WSD

4.2.7.2 NER

4.2.8 Conclusion and Future Work

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

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

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

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

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

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

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

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

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

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

Chapter 5

Conclusions

Bibliography

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

[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 20 and 23.)

[Agirre 2007a] Eneko Agirre and Aitor Soroa. *Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 7–12, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on page 45.)

[Agirre 2007b] Eneko Agirre and Aitor Soroa. *Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems*. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07, pages 7–12, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. (Cited on page 65.)

[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 15, 20 and 23.)

[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 15, 20, 23 and 40.)

[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 55 and 62.)

[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 22.)

[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 54 and 56.)

[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 59 and 61.)

[Berge 1985] C. Berge. Graphs and hypergraphs. Elsevier, 1985. (Cited on page 26.)

[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 31.)

[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 22.)

[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 20.)

[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 18, 21 and 23.)

[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 29.)

[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 page 14.)

[Choudhury 2009b] Monojit Choudhury and Animesh Mukherjee. *The Structure and Dynamics of Linguistic Networks*. In Niloy Ganguly, Andreas Deutsch and Animesh Mukherjee, editors, Dynamics On and Of Complex Networks, Modeling and Simulation in Science, Engineering and Technology, pages 145–166. Birkh?user Boston, 2009. (Cited on page 31.)

[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 and 4.)

[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 22.)

[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 55 and 62.)

[Collins 2002] Michael Collins. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, pages 1–8. Association for Computational Linguistics, 2002. (Cited on page 60.)

[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 page 12.)

[Daume 2006] Harold Charles Daume III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, 2006. AAI3337548. (Cited on pages 59 and 60.)

[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 65.)

[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 17, 19, 23 and 41.)

[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 19 and 23.)

[Estrada 2005] Ernesto Estrada and Juan A Rodriguez-Velazquez. *Complex networks as hypergraphs*. 2005. (Cited on page 17.)

[Firth 1957] J. R. Firth. *A synopsis of linguistic theory 1930-55*. vol. 1952-59, pages 1–32, 1957. (Cited on page 13.)

[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 30.)

[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 62.)

[Giuseppe Attardi 2015] Giuseppe Attardi. *WikiExtractor*. `https://github.com/attardi/wikiextractor`, 2015. (Cited on pages 30 and 32.)

[Goyal 2014] Kartik Goyal and Eduard H. Hovy. *Unsupervised Word Sense Induction using Distributional Statistics*. In COLING, pages 1302–1310. ACL, 2014. (Cited on page 60.)

[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 19.)

[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 25.)

[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 22.)

[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 18, 19, 22 and 23.)

[Jordi Atserias 2008] Massimiliano Ciaramita Jordi Atserias Hugo Zaragoza and Giuseppe Attardi. *Semantically Annotated Snapshot of the English Wikipedia*. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). European Language Resources Association (ELRA), May 2008. (Cited on pages 29, 31 and 35.)

[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, 2009. (Cited on pages 2, 3 and 4.)

[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 17, 22, 23 and 41.)

[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 29.)

[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 16, 20, 21 and 23.)

[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 17, 21, 23, 25, 41 and 42.)

[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 15, 17, 22, 23 and 41.)

[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 15, 19, 22, 23 and 40.)

[Levy 2014] Omer Levy and Yoav Goldberg. *Dependency-Based Word Embeddings*. In ACL (2), pages 302–308. The Association for Computer Linguistics, 2014. (Cited on pages 59 and 62.)

[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 17, 21, 22, 23 and 25.)

[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.)

[Manandhar o 07] Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach and Sameer Pradhan. *SemEval-2010 Task 14: Word Sense Induction & Disambiguation*. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 63–68. Association for Computational Linguistics, 2010-07. (Cited on pages 45 and 50.)

[Manning 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky. *The Stanford CoreNLP Natural Language Processing Toolkit*. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60, 2014. (Cited on pages 33 and 61.)

[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 31.)

[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 16, 20 and 23.)

[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 15, 20 and 23.)

[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 13 and 14.)

[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 15, 21, 23 and 40.)

[Murphy 2012] Kevin P. Murphy. Machine learning: A probabilistic perspective. The MIT Press, 2012. (Cited on page 20.)

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

[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 15, 17, 20, 23 and 40.)

[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 17, 19, 23 and 41.)

[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 page 59.)

[Pedersen 2010] Ted Pedersen. *Duluth-WSI: SenseClusters applied to the sense induction task of SemEval-2*. In Proceedings of the 5th international workshop on semantic evaluation, pages 363–366. Association for Computational Linguistics, 2010. (Cited on pages 50 and 51.)

[Qian 4 08] Tao Qian, Donghong JI, Mingyao Zhang, Chong Teng and Congling Xia. *Word Sense Induction Using Lexical Chain based Hypergraph Model*. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1601–1611. Dublin City University and Association for Computational Linguistics, 2014-08. (Cited on pages 17, 22, 23 and 25.)

[Rada Mihalcea and Dragomir Radev 2011] Rada Mihalcea and Dragomir Radev. Graph-based natural language processing and information retrieval. Cambridge University Press, 2011. Cambridge Books Online. (Cited on page 31.)

[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 page 60.)

[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 31.)

[Saluja 2013] Avneesh Saluja and Jirı Navrátil. *Graph-Based Unsupervised Learning of Word Similarities Using Heterogeneous Feature Types*. page 29, 2013. (Cited on pages 19, 21 and 23.)

[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 61.)

[Schaeffer 7 08] Satu Elisa Schaeffer. *Survey: Graph Clustering*. vol. 1, no. 1, pages 27–64, 2007-08. (Cited on page 21.)

[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 30.)

[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 30.)

[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 15, 20, 21, 23 and 40.)

[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 15, 20, 23 and 40.)

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

[Tsatsaronis 7 01] George Tsatsaronis, Michalis Vazirgiannis and Ion Androutsopoulos. *Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri*. 2007-01. (Cited on pages 15, 19, 21, 23 and 40.)

[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 50 and 51.)

[Véronis 2004] Jean Véronis. *HyperLex: lexical cartography for information retrieval*. vol. 18, no. 3, pages 223 – 252, 2004. (Cited on pages 17, 21, 23, 41 and 42.)

[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 29.)

[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 29.)

[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 16, 21 and 23.)

[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 56.)