Semantic Feature Extraction Using Multi-Sense Embeddings and Lexical Chains

by

Terry L. Ruas

**A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer and Information Science)
in the University of Michigan-Dearborn
2019**

Doctoral Committee:

**Professor William Grosky, Chair
Assistant Professor Mohamed Abouelenien
Rajeev Agrawal, US Army Engineer Research and Development Center
Associate Professor Marouane Kessentini
Associate Professor Luis Ortiz
Professor Armen Zakarian**

Terry L. Ruas

truas@umich.edu

ORCID iD 0000-0002-9440-780X

DEDICATION

To my father, Júlio César Perez Ruas, to my mother Vládia Sibrão de Lima Ruas, and my dearest friend and mentor William Grosky.

This work is also a result of those who made me laugh, cry, live, and die, because if was not for them, I would not be who I am today. . .

ACKNOWLEDGEMENTS

First, I would like to express my gratefulness to my parents, who always encouraged me, no matter how injurious the situation seemed. With the same importance, I am thankful to my dear friend and advisor Prof. William Grosky, without whom the continuous support of my Ph.D. study and related research, nothing would be possible. His patience, motivation, and immense knowledge are more than I could ever have wished for. His guidance helped me during all the time spent on the research and writing of this dissertation. I could not have asked for better company throughout this challenge. If I were to write all the great moments we had together, one book would not be enough.

Besides my advisor, I would like to thank the rest of my dissertation committee: Dr. Mohamed Abouelenien, Dr. Rajeev Agrawal, Dr. Marouane Kessentini, Dr. Luis Ortiz, and Prof. Armen Zakarian, for their insightful comments and encouragement, but also for the hard questions which inspired me to widen my research from various perspectives.

My sincere thanks also goes to Prof. Akiko Aizawa, who was my supervisor during my Ph.D. internship at the National Institute of Informatics (NII) in Tokyo, Japan. The connections and experiences I had there were essential for my growth as a researcher. Working in her lab helped me polish and mature my research work, assisting me in becoming a better scholar.

I also would like to thank my dear friend Edward Williams, the person who first introduced me to the University of Michigan during a conference in Argentina in 2013. I had a pleasure to share with him countless lunches and talks during my stay in the United States. If was not for him, I would probably still be working in my previous job at IBM in Brazil.

I would like to leave a special appreciation to Charles Henrique Porto Ferreira, whom I had the pleasure to work with during my last year as a doctoral student. His visit provided stimulating discussions, great quality time, and an amazing perspective on how to conduct research with integrity and organization. Anyone who has the opportunity to work with him should feel honored.

Last, but not the least, I would like to thank my dear friends in Brazil, which I chose to be my family. It would not be fair to list them risking leaving someone out. For this reason, I express my gratitude to all of them, who directly and indirectly made my journey easier during tough times.

Absque sudore et labore nullum opus perfectum est...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI Artificial Intelligence

ACL Association for Computational Linguistics

ACM Association for Computing Machinery

BOW Bag-of-Words

BSD Best Synset Disambiguation

BERT Bidirectional Encoder Representations from Transformers

biLM bidirectional Language Model

BoMW Bag-of-Meta-Words

BOWR Bag-of-Words-Raw

BOWN Bag-of-Words-WordNet

BOWS Bag-of-Words-Synsets

BOWB Bag-of-Words-Best

BSID Best SynsetID

CBOW Continuous Bag-of-Words

CNN Convolutional Neural Network

CRF Conditional Random Fields

CNN-MSSG Convolutional Neural Network Multi-Sense Skip-Gram

CWS Context Word Similarity

ELMo Embeddings from Language Models

F2F Flexible to Fixed Lexical Chains

FCM	Features Combination Meta-Word Model
FLC	Flexible Lexical Chains
FSID	Former SynsetID
FXLC	Fixed Lexical Chains
FLLC II	Flexible Lexical Chains II
FXLC II	Fixed Lexical Chains II
GloVe	Global Vectors
IC	Information Content
IR	Information Retrieval
LC	Lexical Chains
LR	Logistic Regression
IDC	International Data Corporation
IEEE	Institute of Electrical and Electronics Engineers
K-NN	K-Nearest Neighbors
LDA	Latent Dirichlet Allocation
LSID	Latter SynsetID
ML	Machine Learning
MWE	Multiword Expressions
MeSH	Medical Subject Headings
MSSA	Most Suitable Sense Annotation
MSSA-D	Most Suitable Sense Annotation - Dijkistra
MSSA-NR	Most Suitable Sense Annotation - N Refined
MSSG	Multi-Sense Skip-Gram
MT-DNN	Multi-Task Deep Neural Network
NB	Naïve Bayes
NIM	Naïve Interval Meta-Word Model
NLP	Natural Language Processing

NLU Natural Language Understanding

NLTK Natural Language Toolkit

NCWS No Context Word Similarity

NP-MSSG Non-Parametric Multi-Sense Skip-Gram

NASARI Novel Approach to a Semantically-Aware Representation of Items

OOV Out-of-Vocabulary

PI Principal Investigators

PV Paragraph Vectors

POC Proof-of-Concept

POS Part-of-Speech

Ph.D. Doctor of Philosophy

PV-DM Distributed Memory Model of Paragraph Vectors

PV-DBOW Distributed Bag-of-Words of Paragraph Vectors

RF Random Forests

RSS Recommender Systems

RIS Research Information Systems

RCV1 Reuters Corpus Volume I

skip-gram Continuous Skip-Gram

SANs Spreading Activation Networks

SPR Scientific Paper Mining and Recommendation

STM Semantic Topic Model

SVM Support Vector Machine

SCWS Stanford Context Word Similarity

tf-idf term frequency-inverse document frequency

USE Universal Sentence Encoder

WN WordNet

WSD Word Sense Disambiguation

ABSTRACT

The relationship between words in a sentence often tell us more about the underlying semantic content of a document than its actual words individually. Natural language understanding has seen an increasing effort in the formation of techniques that try to produce non-trivial features, in the last few years, especially after robust word embeddings models became prominent, when they proved themselves able to capture and represent semantic relationships from massive amounts of data. These recent word embeddings models represent words, sentences, and entire documents in a dense n-dimensional space that can be used in any problem involving natural language text. Although, the new dense vector representations indeed leverage the baseline in natural language processing arena, they still fall short in dealing with intrinsic issues in linguistics, such as polysemy and homonymy. Systems that make use of natural language at its core, can be affected by a weak semantic representation of human language, resulting in inaccurate outcomes based on poor decisions (e.g. tutoring systems, recommender systems, health companions).

In this subject, word sense disambiguation and lexical chains have been exploring alternatives to alleviate several problems in linguistics, such as semantic representation, definitions, differentiation, polysemy, and homonymy. However, little effort is seen in combining recent advances in token embeddings (e.g. words, sentences, documents) with word sense disambiguation and lexical chains. To collaborate in building a bridge between these areas, this work proposes two collections of algorithms to extract semantic features from large corpora as its main contributions. The first group of contributions is composed of three techniques, namely Most Suitable Sense Annotation (MSSA), Most Suitable Sense Annotation - Dijkistra (MSSA-D), and Most Suitable Sense Annotation - N Refined (MSSA-NR). These approaches are focused on disambiguating and annotating each word by its specific sense, considering the semantic effects of its context. The algorithms in this category combine the benefits of word embeddings, lexical structures, and word sense disambiguation in a mutual self-improving system. While MSSA works in a local context, trying to choose the best representation for a word, MSSA-D adopts a global approach, in which it considers the most similar word-senses from the first to the last word in a document. In MSSA-NR, we propose a novel recurrent approach that can be applied to refine the semantic representation results of either MSSA or MSSA-D. The second group offers another two new techniques, called Flexible Lexical Chains II (FLLC II) and Fixed Lexical Chains II (FXLC II), which derive the semantic relations between consecutive words in a document. These original techniques' target is to uncover the implicit semantic links between words using their lexical structure, incorporating multi-sense embeddings, word sense disambiguation, lexical chains, and the prior knowledge from lexical databases. In the FLLC II version, the semantic lexical chains are assembled dynamically, as long as there is a semantic relation that connects two adjacent words. In FXLC II, these semantic lexical chains are built using pre-defined chunks, which will hopefully describe their semantic content.

A few natural language problems are selected to validate the contributions of this work, in which our techniques outperform state-of-the-art systems. All the proposed algorithms can be used separately as independent components or combined in one single system to improve the semantic representation of words, sentences, and documents. Additionally, they can also work in a recurrent form, refining even more their results.

CHAPTER I

Introduction

Semantic analysis is arguably one of the oldest challenges in Natural Language Processing (NLP), still present in almost all its downstream applications, and many real-world problems. Even among humans, the precise definition of semantics is not a consensus, which leads to multiple interpretations of text, making computational semantics even more challenging [136].

Despite being a classical problem, the popularity of semantic analysis continues to draw the attention of many research projects in different areas of study, under the rubric of *semantic computing*. For example, Grosky and Ruas [60] analyzed 2,872 multimedia publications (e.g. papers, journals, reports) between 2005 and 2015, revealing an increasing trend in publications involving *semantics* and *contextual aspects* in different areas of multimedia. In these publications, methods applying different techniques try to capture semantic characteristics of text documents using state-of-the-art approaches, such as latent semantic analysis, word embeddings, machine learning, and artificial neural networks.

As we are aware, data production is reaching numbers never seen before in the world. Documents being uploaded in online repositories, instant text messages, ubiquitous computing, the internet itself, and many other channels are used every second, encouraging even more this scenario. In 2013, we had reached the mark of 4.4 zettabytes (10^{21} bytes), by 2020 this number is expected to be close to 44 zettabytes, and in 2025 we have a forecast of 175 zettabytes, according to International Data Corporation (IDC) [138]. In fact, some reports state that about 90% of all data available today was produced in the last two years [21].

Following the accelerated growth in data production, the need for developing methods that are able to extract useful information from it has also become essential. More specifically, Natural Language Understanding (NLU) is a real-world necessity for any intelligent system willing to achieve true synergy between humans and machines. The race for such an objective has become evident in the last few years, especially after robust word embeddings models became prominent, when they proved themselves able to capture and represent semantic relationships from massive amounts of data. Nevertheless, traditional models often fall short in intrinsic issues of linguistics, such as polysemy and homonymy.

As a consequence of this accelerated data production, textual documents provide the ideal scenario for semantic feature extraction methods. However, their large quantity also brings certain difficulties in uncovering useful information underneath it. From a different perspective, even though our ability to produce data has grown rapidly, our efficiency in semantically analyzing it did not, which leads to what is called the *semantic deficiency gap* [67, 151]. In this work, we contribute in trying to solve this problem by proposing new techniques that will hopefully produce more robust semantic features to be used for solving natural language problems.

1.1 Problem Context

The relationships between words in a sentence often tell us more about the latent semantic content of a document than its individual words. Recent advances in the NLP arena, more specifically those using word embeddings, try to incorporate semantic aspects into their word vector representations by considering the context of words and how they are distributed in a document collection. Unfortunately, the amount of data available does not equal the semantic information we are currently able to extract. Even though word embeddings techniques alleviate some problems (e.g scalability) of traditional count-based methods, such as Bag-of-Words (BOW), they still fall short in properly dealing with common challenges in linguistics (e.g. polysemy, homonymy).

Extending the problem of identifying the relationships between words in a sentence to entire documents, text classification provides a scenario even more challenging. Text mining [15] is a sub-area of machine learning (ML) that intends to automatically extract knowledge from text documents. Its usefulness can be extended to several areas in the NLP domain, such as detecting spam emails [42, 68], authorship identification [72, 189], and text summarization [7, 95]. Among these areas, text classification has received considerable attention and has been the subject of several recent research topics [188, 82, 49, 50, 165]. In the document classification task, one has to create a model using features extracted from a set of text data, often referred to as a training set, that is capable of inferring the correct labels of unseen text documents. The success of this model depends on the balance between accuracy and generalization of the basis approximation function and the quality of the extracted features. Feature extraction from text data is a research topic all its own in document classification, due to its unstructured nature, which provides an interesting scenario for the proposed techniques in this work.

After recent contributions [109, 110, 130], word embeddings techniques have received much attention in the NLP community. These approaches represent words or phrases as real vectors which can be used to extract relationships between them. The overall performance of these algorithms has demonstrated superior results in many different NLP tasks, such as: chunking [39], meaning representation [20], machine translation [110], relation similarity [74, 111], sentiment analysis [166], word-sense disambiguation (WSD) [26, 32, 120], word similarity [31, 74, 122], and topic categorization [132].

Notwithstanding their robustness, however, most traditional word embedding approaches fail to deal with polysemy and homonymy problems [91]. Recently, researchers have been trying to improve their representations by producing multiple vectors (multi-sense embeddings) based on the word's sense, context, and distribution in the corpus [73, 140]. Another concern with traditional techniques is that they often neglect exploring lexical structures with valuable semantic relations, such as: WordNet (WN) [48], Concept-

Net [94], and BabelNet [121]. Some publications take advantage of these structures and combine them into multi-sense vector representations, improving their overall performance [74, 75, 91, 99, 133, 145].

In addition to ignoring valuable prior knowledge from lexical databases, traditional word embeddings often neglect the semantic relationship between consecutive words (i.e. word order) in their representation. As a response to the absence of word order, the construction of lexical chains presents itself as a viable option [179, 58, 148, 149]. *Lexical chains* are defined as sequences of related words delineating portions of text for a specific topic in a cohesive manner [118]. A text in which its sentences are semantically connected often produces a certain continuity in its ideas, providing good cohesion among its sentences. Cohesion, which illustrates how well lexically or grammatically structured a text is, often occurs in words close to each other in a text, especially those adjacent to one another [62]. Even though the applicability of lexical chains is quite diverse, we see little work in combining and comparing them with recent advances in NLP, more specifically with word embeddings [164, 58, 100].

The novelty of this work relies on the proposition of a complete modular recurrent process that incorporates the benefits of WSD, multi-sense word embeddings, prior knowledge from lexical databases, and lexical chains. Others have worked on these topics separately, in rare occasions combining two areas (e.g. WSD and word embeddings, lexical chains and word embeddings), but not with all of them nor with the same cohesion here proposed. Details of the overall system architecture are discussed in Chapter III, in which we explain the general idea of the main contributions proposed and how their constituent components can also be used in a stand-alone fashion.

1.2 Main Objectives

The main objective in this work is to decrease the semantic deficiency gap between natural language text and their real semantic content. For this, we propose a collection of novel approaches that incorporate WSD, multi-sense word embeddings, semantic relations from lexical databases, and lexical chains. This has been accomplished through five specific objectives, as follows:

1. The development of multiple unsupervised word sense disambiguation and annotation algorithms. These help us to identify the most suitable word-sense of a given word considering the influence of its immediate neighbors;
2. The development of multiple unsupervised algorithms to extract the semantic relations between words in text documents. These relations are extracted by building lexical chains in a dynamic or fixed manner;
3. A modular system combining the proposed algorithms in a single recurrent process. The word sense disambiguation and lexical chains algorithms are designed to work as stand-alone components or integrated as part of one end-to-end system;
4. The validation of the proposed techniques via particular NLP tasks, such as word similarity, document classification, document retrieval, and keyword extraction tasks. State-of-the-art benchmark systems are used as a form of comparison in each task; and
5. A collection of lightweight multi-sense word embeddings models for the proposed techniques. The generated models, which are at least 75% smaller than the compared pre-trained models, are publicly available[1] so they can be used in various natural language problems.

1.3 Dissertation Structure

This dissertation is structured as follows.

[1]https://github.com/truas

Chapter II presents the necessary background for the comprehension of this dissertation. This includes the topics of: WSD, lexical chains, and word embeddings. Besides the introductory theory, this chapter also presents some of the related work in each field and the NLP downstream tasks publications associated with them.

Chapter III brings the main contributions of this work, starting from a more general perspective to a more detailed one. The overall architecture is presented and each component is followed by a detailed explanation of its internal mechanisms and peculiarities. The early versions of the proposed techniques are also included in Section 6.2.

Chapter IV contains the experiments performed to validate the proposed techniques in the word similarity and document classification tasks. In addition, this chapter also includes some deeper discussions about the performed experiments and exposes the main limitations of each approach, while providing insights on how to mitigate them. The experiments for document retrieval and keyword extraction, considering our early techniques, are described in Section 6.3.

Chapter V discusses the final considerations about the proposed techniques, experiments, and their most relevant constraints. Furthermore, solid future directions about this work and how the main contributions presented can be applied in other problems are also explored.

Chapter VI presents the early versions and experiments of the proposed techniques in Chapter III, and their respective related work.

CHAPTER II

Background and Related Work

This chapter presents the necessary background related to the main contributions of this work. This includes the topics of: WSD, lexical chains, and word embeddings. Besides the introductory concepts, this chapter also presents the related work in each field as well.

2.1 Word Sense Disambiguation

The meaning of a sentence in a document is more easily determined if its constituent words exhibit cohesion with respect to their individual semantics. The multiple interpretations of words make it hard to provide a common definition for them, especially if all human vocabulary is considered. Weaver [178] first introduces the problem of WSD as a language translation issue, in which he states "[. . .] *the obvious fact that a multiplicity of languages impedes cultural interchange between the peoples of the earth, and is a serious deterrent to international understanding*". Even in simple sentences, the multiple senses of a word can bring high ambiguity into play. Let us consider the following sentence:

The good person likes the club.

it is not clear if the term *club* is related to the sense of *baseball club*, *clubhouse*, *golf club*, or any other appropriate sense. Moreover, *good* and *person* suffer similar problems with respect to their multiple meanings. Systems that are able to differentiate between the many word senses, can leverage the semantic representation of words in any natural language problem (e.g. document classification, sentiment analysis).

Mallery [98] describes WSD as an AI-complete problem. In other words, WSD has the same difficulty of fundamental problems of artificial intelligence (AI) [175]. The hardship in solving WSD problems comes from several aspects in linguistics, in which two play an important role in computer science [120]: (i) word representation and (ii) knowledge dependency. Word representation is related to how we represent all possible word senses for each word, how granular a word sense should be, how to deal with the recursive representation of a word sense definition, the domain-oriented versus unrestricted nature of texts, etc. Knowledge dependency pertains to the prior knowledge of words and how to properly represent their true meaning. In addition, the creation of lexical databases is an expensive and time-consuming task [123], especially because each time the context changes, a new disambiguation process must be considered. To this circumstance, Gale et al. [54] give the name of *knowledge acquisition bottleneck*.

The association between senses and words has an important role in any task involving WSD. Saying that, the bridge between knowledge representation and words can come from different resources, such as lexical databases, machine-readable dictionaries, thesauri, raw corpora, annotated corpus, ontologies, and collocation corpora [120]. Once the relation between words and their senses is considered, one has to decide which sense better represents a word in a given circumstance. In this context, we suggest two broad groups in which word senses are evaluated to better describe a word: (i) no word embeddings-based sense disambiguation, and (ii) word embeddings-based sense disambiguation. In the first group (i), two words are said to be similar (or dissimilar) given a specific metric [108]:

Path-based. Path-based approaches are represented by functions that describe how two concepts are related (path and position) in the lexical database [181];

Information content-based. Information Content (IC) approaches assume that every word sense has incorporated some amount of information. Similarity scores in this category are calculated taking into account the IC for each word sense [77];

Feature-based. Feature-based measures do not specifically rely on a lexical database. Instead, they use properties from a lexical structure to calculate their similarity scores, such as glosses[1] in WordNet [177]; and

Hybrid. Hybrid measures combine more than one approach. Similarity scores in this category can use IC with feature-based measures, IC with path-based measures, and other combinations [190].

In the second group (ii), are all techniques that use word embeddings during the WSD process. The proposed techniques in Chapter III follow research segment (ii) and are based in a multi-source knowledge representation, composed of raw corpora, annotated corpora, and a lexical database. For more details on the metrics and references in group (i), Section 6.1 provides an extensive explanation for each example above.

2.1.1 Related Work in Word-Sense Disambiguation

Word Sense Disambiguation (WSD) is the task of selecting the most suitable word-sense (i.e. meaning) of a given word in a specific context. Navigli [120] presents an extensive study in the WSD arena, in which he proposes an unsupervised WSD algorithm based on generating Spreading Activation Networks (SANs) from word senses of a thesaurus and the relations between them. Meng et al. [108] explore the theory behind state-of-the-art techniques for semantic similarity measures in four main categories: path length-based, information content-based, feature-based, and hybrid measures. AlAgha and Nafee [5] propose an approach to improve document clustering, exploring the semantic knowledge offered in Wikipedia. The authors discuss this hypothesis, comparing the results using WordNet and Wikipedia, claiming that the latter is more robust for finding the semantic clusters.

In [134], several measures of similarities (e.g. normalized Google distance, normalized compression distance, cosine distance, latent semantic similarity) are applied to catego-

[1]https://wordnet.princeton.edu/documentation/wngloss7wn

rize words, sentences, paragraphs, and documents according to their lexical and semantic similarities. Bär et al. [6] present a detailed study on how several text similarity measures work. They explore these measures in the context of the semantic text similarity task. In their report, they propose an architecture that combines text similarity measures in a unified classification framework. As part of their semantic evaluation and text reusability detection, they argue that text similarity should not be considered a static entity. Instead, one has to carefully define in which levels and perspectives two documents are similar or not. In [170], they perform a comprehensive review of the methods related to XML-based semi-structured semantic analysis and disambiguation. Although their work is more focused on XML, they provide an overview about the semantic disambiguation field as well, covering traditional WSD methods, potential application scenarios that could benefit from it (e.g. data clustering, semantic-aware indexing), and discuss current ongoing challenges in the area.

Even though there are more references in the WSD arena, they incorporate word embeddings, in their core activity, to differentiate among multiple word senses. For this reason, it is better to include these contributions after the word embeddings field is properly introduced in Section 2.3.

2.1.2 Keyword Extraction and Related Work

The task of keyword extraction from documents can be done using several different approaches (e.g. most frequent word, human tagging), including using semantic analysis. Saratlija et al. [158] propose a fully unsupervised keyphrase extraction method to build topic-related word clusters from documents' keywords. These selected keywords are expanded into valid lexical keyphrases with a reported accuracy of 44.5%. Hasan and Ng [65] conduct a survey on state-of-the-art approaches for keyphrase extraction. They perform a systematic evaluation and analysis of keyphrase extraction algorithms on standard datasets. Their results show that, despite the robustness of traditional weighting schemes (e.g. tf-idf) for keyphrase extraction, alternative approaches should not be ignored.

In [102], the authors develop an algorithm to extract keywords based on the co-occurrence distribution of words from a single document. This is done without the need of an actual corpus, in comparison to traditional term frequency-inverse document frequency (tf-idf) approaches, in which they claim to achieve comparable results. Silva and Lopes [163], use statistical and language independent methods to describe the core content of documents, based on the automatic extraction of single words and Multiword Expressions (MWE). They conclude that, by introducing the median of a word's length for each MWE and the preference for (2 or 4)-grams, the quality of documents, with respect to precision and recall, increases approximately 11% and 9% respectively. Zhang et al. [187] present a small survey relating lexical chains, semantic fields, and complex networks. The latter approach allows for a deeper investigation via discourse coherence analysis, showing encouraging results. In [12], a survey and categorization of keyword extraction methods are presented, focusing on supervised, unsupervised, and graph-based approaches.

In the field of Semantic Topic Model (STM), Ercan and Cicekli [45] select the most representative and significant sentences from a text to summarize them. Their approach makes use of co-located lexical chains to better represent lexical cohesion clues and uses them as the core representation of a text. The proposed summarization algorithm achieves better results in comparison with other ones in the same category. In a more ambitious approach, Chen and Lin [30] use Google similarity distance to measure keywords in webpages and find potential future terms to suggest to the user. Their methodology does not require the user to type any keywords. Instead, they track what the user is searching for based on the webpages recently browsed. This is accomplished with an average precision and recall above 50%. In the medical arena, Jonnalagadda et al. [81] design a hybrid approach (i.e. supervised and unsupervised) for extracting clinical concepts from the i2b2/VA corpus[2]. Their approach uses Conditional Random Fields (CRF) to extract information

[2]https://www.i2b2.org/NLP/DataSets/Main.php

from medical statements and estimates the relatedness between words in the corpus and the training/testing corpora.

2.2 Lexical Chains

Lexical cohesion is more likely to occur between words close to each other in a text, especially those contiguously ordered. The semantic similarity between these groups of words is classified as a *lexical chain* [118]. Morris and Hirst [118] first introduced the term *lexical chains* as an extension of *lexical cohesion* [62]. A text in which many of its sentences are semantically connected often produces a certain degree of continuity in its ideas, providing good cohesion among its sentences.

The definition used for lexical cohesion states that coherence is a result of cohesion, not the other way around [28, 62]. In other words, cohesion is related with a set of words that belong together due to some abstract or concrete relation. Coherence, in the other hand, is more concerned with the actual meaning in the whole text [118]. In this work, we seek to obtain lexical cohesion through the proposal of new algorithms for building lexical chains, which will hopefully bring some coherence to a document representation. However, it is important to mention that the very presence of lexical cohesion will not, by any means, assure this document superior coherence in relation to another [105].

Halliday and Hasan [62] divide cohesion into two major categories: grammatical and lexical cohesion. In grammatical cohesion, they include five general categories of cohesive devices: reference, substitution, ellipsis and conjunction. Lexical cohesion, on the other hand, is related to semantic construction of a text portion and groups: reiteration (i.e. repetition, synonyms, superordinate and general words) and collocation. Where *repetition* is a linguistic artifact to obtain lexical cohesion through the (near) duplication of words. *Synonyms* are constructions (e.g. words, phrases) that can be replaced by others with the same or similar meaning. *Superordinates* are words that represent a higher level of abstraction, providing less specific information about something and are quite different from synonyms since they cannot be easily switched in a document. *General words* refer to a set of words where each word can be used to refer the same topic/event. Finally, *collocation* is the juxtaposition of two or more words that are often used together in a sentence (e.g. orthogonal projection, ice cream) [105].

As Halliday and Hasan [62] state, "*Cohesion does not concern what a text means; it concerns how the text is constructed as a semantic edifice*". In other words, cohesion is related to the structure of a document through a semantic perspective. Morris and Hirst [118] define that lexical chains make use of these structures to compute, in a non-domain-specific environment, the semantic context for interpreting words, concepts and sentences. While lexical cohesion is more focused on the relationships of word pairs, lexical chains extend this notion to a successive number of adjacent words. There are two main reasons why lexical chains are important for computational text understanding systems [118]:

- They provide a feasible context to assist in the ambiguity and narrowing problems to a specific meaning of a word; and
- They provide clues to determine coherence and discourse, thus a deeper semantic-structural meaning of the text.

The method presented in [118] is the first to bring the concept of lexical cohesion to the computational world via lexical chains. Using their intuition, they identify lexical chains in text documents and build their structure considering [62] observations. For this task, they consider five text documents, totaling 183 sentences from different and non-specific sources. Repetitive words (e.g. high frequency words, pronouns, propositions, verbal auxiliaries) are not considered as prospective chains elements, since they do not bring much semantic value to the structure themselves.

Lexical chains are built according to a series of relationships between words in a text document. In the seminal work of Morris and Hirst [118], they consider an external thesaurus (Roget's Thesaurus [143]) as their lexical database to extract these relations. A lexical chain is formed by a sequence of words $\{w_1, w_2, \ldots, w_n\}$ appearing in this order, such that any two consecutive words w_i, w_{i+1} possess the following properties[3] [118]:

- Two words share one common category in their index;
- The category of one of these words points to the other word;
- One of the words belongs to the other word's entry or category;
- Two words are semantically related; and
- Their categories agree to a common category.

As for the quality of a built chain, three factors can be considered to measure how strong a lexical chain is [118]: *reiteration*, *length*, and *density*. Reiteration illustrates how often a lexical chain occurs, the length is related to the number of tokens on it, and density is the ratio of words in the lexical chain to the words in the text document. Since our main objective is to evaluate how the proposed lexical chains perform in downstream tasks, we leave the evaluation of their strength to a future opportunity.

We consider the five characteristics defined in [118] to build our lexical chains and validate their quality in the document classification problem. These characteristics can be observed in various text documents, especially in scientific/academic ones, where theories and experiments have to be explained in great details through several different perspectives. The very notion of cohesion is accepted as a good tool for text analysis to capture information beyond pure syntax [157]. For example, in the sentence "*the code in my programs is more cohesive thanks to the new version of Java*", the set $\{code, programs, cohesion, Java\}$ expresses a successive chain of semantically related terms representing an association with computers or programming languages. In a larger scope, other characteristics can be explored, such as how the structural distribution of related chains, present in documents, can be used to summarize and recommend related scientific papers.

[3]Where category, indexes, and pointers are attributes in the lexical database considered.

The proposed algorithms (Chapter III) explore how these lexical cohesive characteristics, presented in [62, 118], can be extended, considering sets of cognitive synonyms, instead of just words. The next section presents some of the related work that apply and extend lexical chains in various scenarios.

2.2.1 Related Work in Lexical Chains

The use of lexical chains in NLP tasks (e.g. text similarity, word sense disambiguation, document clustering) has been widely studied in the literature. In [7], lexical chains are used to produce summaries from texts. They propose a technique based on four steps: segmentation of original text, construction of lexical chains, identification of strong chains, and extraction of significant sentences. Their results show improvements for both precision and recall, when compared to Microsoft Summarizer. Silber and McCoy [162] also investigate text summarization, but their approach for constructing the lexical chains runs in linear time, in comparison with [7], which does not.

Some authors use WordNet to improve the search and evaluation of lexical chains. Budanitsky and Hirst [24, 25] compare several measurements of semantic distance and relatedness using lexical chains in conjunction with WordNet. Their study concludes that the similarity measure of Jiang and Conrath [77] presents the best overall result. Moldovan and Novischi [115] study the use of lexical chains for finding topically related words for question answering systems. This is done considering the glosses for each synset[4] in WordNet. According to their findings, topical relations via lexical chains improve the performance of question answering systems when combined with WordNet. In [71], they explore the benefits of using WordNet to improve document clustering based on an explicit matching between terms found in the text and the lexical database. They improve their clustering results incorporating background knowledge into the representation of documents. McCarthy et al. [103] present a methodology to categorize and find the most predominant synsets in

[4]Set of synonyms [1]

unlabeled texts using WordNet. Different from traditional approaches (e.g. BOW), they consider relationships between terms not occurring explicitly. They obtained a WSD precision of 64% on all noun-tasks related against the baseline (SemCor[5] and SENSEVAL2[6]).

In [159], WordNet is used for document clustering, exploring the benefits of incorporating hypernyms and synonyms into their approach. However, they reported poor results when using only Part-of-Speech (POS) tags for document clustering. Pedersen et al. [127] propose *WordNet::Similarity*, an application developed in Perl to calculate the relatedness of concepts via WordNet through different measures of similarity. Ercan and Cicekli [44] explore the effects of lexical chains in the keyword extraction task through a supervised machine learning perspective. Their methodology obtains superior results when considering a decision tree induction algorithm, based on lexical chains and text-only features. Guo and Diab [61] hypothesize that if the semantics of words are known in advance (i.e. prior knowledge), it is possible to get a better statistical inference concerning a document's overall idea. Their STM approach combines explicit semantic information and word distribution information, presenting better results than traditional Latent Dirichlet Allocation (LDA) models.

In more recent works, Pradhan et al. [134] apply several measures of similarities (e.g. normalized Google distance, normalized compression distance, cosine distance, latent semantic similarity) to categorize sentences, words, paragraphs, and documents according to their lexical and semantic similarities. Their integrated approach, combining kernel-based and cosine-based similarity, provides better results if compared to the traditional cosine similarity. Wei et al. [179] combine lexical chains and WordNet to extract a set of semantically related words from texts, and use them for clustering. Their approach uses an ontological hierarchical structure to provide a more accurate assessment of similarity between terms during the WSD task.

While covering many aspects in the lexical chains arena, the current literature still

[5]http://web.eecs.umich.edu/m̃ihalcea/downloads.html#semcor

[6]http://www.hipposmond.com/senseval2/Results/guidelines.htm#rawdata

presents opportunities for expansion. Most authors rely on machine learning techniques to improve their semantic extraction, which are highly dependent on the available corpus. Others ignore the context-effect present within the text. The meaning of a word is directly affected by its surroundings, so the textual neighborhood of a word should be used to decide what is its best semantic representation. Moreover, prior knowledge from external lexical databases have much to offer with respect to semantic relationships besides trivial artifacts, such as hypernyms and hyponyms.

Those inspecting lexical chains often build them using words individually, or using some common/direct synonym. Although these are interesting approaches, they are only focused on the word itself, leading to an alternative BOW representation. Semantic and contextual aspects are difficult to extract, but important elements of effective human communication to track. In the last eleven years, the interest in these topics, and their contributions to traditional approaches, have been increasing among distinct scientific communities. In [60], 2,872 records (e.g. papers, journals) in the multimedia field are examined revealing an increasing number of articles exploring semantic and contextual aspects in different areas as a common trend. Finally, little effort is seen in combining the benefits of word embeddings and lexical chains, which could leverage the semantic representation of textual documents.

2.3 Word Embeddings

The distributed representation of words from documents has received substantial attention from the NLP community in the last few years, especially after the extremely popular *word2vec* approach was proposed [109, 110]. However, the idea that words with similar contexts should have similar meanings goes back to the 20th century in the *Distributional Hypothesis* [64]. Later, the presence of these words would be described using count-based methods (e.g. BOW [155]). Due to its simplistic and naive methodology, the BOW approach has some drawbacks, such as data sparsity, loss of word order, and high dimensionality, to name a few. Bengio et al. [13] try to solve the latter problem (dimensionality) proposing a neural probabilistic language model that learns a representation while keeping a compact probability distribution of word sequences through back-propagation [153]. Collobert and Weston [33] later define a faster general single Convolutional Neural Network (CNN) architecture, showing that multitask learning and semi-supervised learning can improve the generalization in shared tasks, such as: POS tagging, morphological segmentation, named entity recognition, and word similarity. Besides these, other language prediction models are also popular in the NLP community [20, 174, 176, 191].

The fundamental objective in word embeddings techniques is to find word representations in a vector space that will improve the performance in solving various natural language problems. Two techniques deserve special attention in this scenario, called Continuous Skip-Gram (skip-gram) and Continuous Bag-of-Words (CBOW) [109, 110]. In the skip-gram training model, one uses an input word to predict its context, while in the CBOW variation the converse happens, in which given a context the model tries to find the most probable word to appear. Figure 2.1 shows an example of both architectures in a high-level perspective. The input and output layers are one-hot encoded, and have the same number of dimensions. However, the projection layer, usually significantly smaller than the vocabulary size, can be of any dimension desired.

The values used to encode each dimension of the vector are the weights used in the activation function in the artificial neural network. The training objective of the skip-gram is to maximize the log probability in Equation 2.1 [110].

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,\ j\neq 0}\log p(w_{t+j}|w_t) \tag{2.1}$$

where c is the size of the context window (considering w_t as the center) and $p(w_{t+j}|w_t)$ is defined using the softmax function in Equation 2.2.

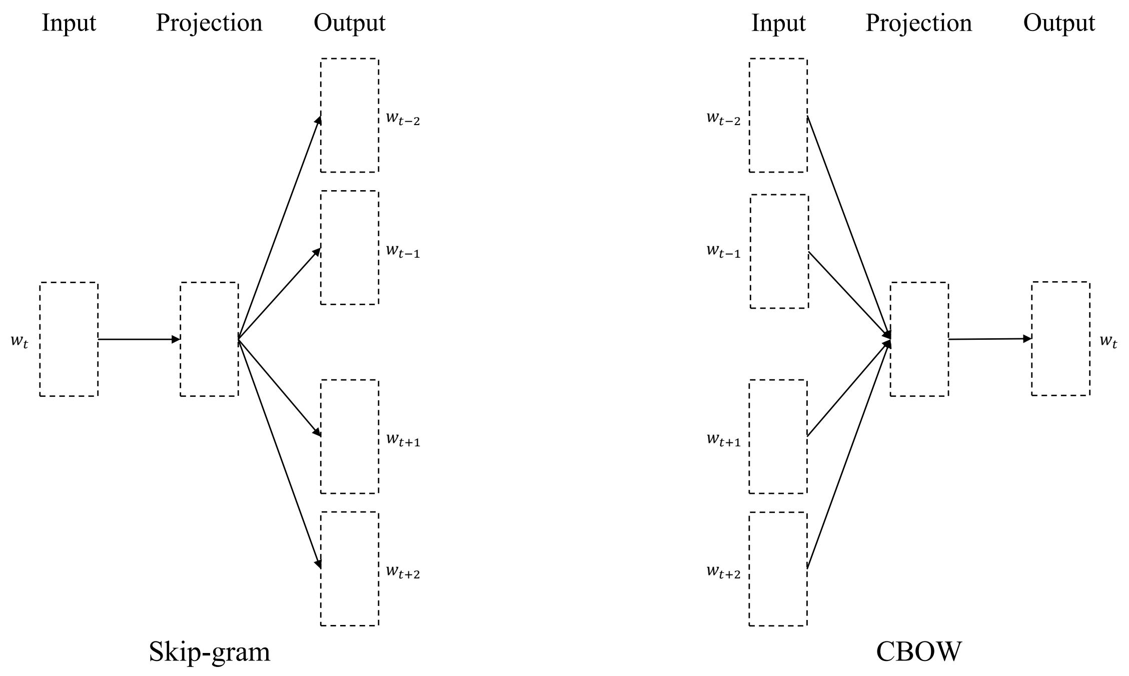


Figure 2.1: Skip-gram and CBOW architectures. The skip-gram predicts context words given the current word, and CBOW predicts the word considering its context [109]
.

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^\top v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^\top v_{w_I})} \tag{2.2}$$

where v_w and v'_w are the input and output vector representation of w respectively and W is the size of the vocabulary considered. However, as W increases, it is not feasible to calculate Equation 2.2. Morin and Bengio [116] propose a binary tree representation for the output layer with the W words as its leaves, resulting in an effective approximation of the softmax function, called *hierarchical softmax*. Thus $p(w_O|w_I)$ can be redefined as Equation 2.3 shows [110].

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I}) \tag{2.3}$$

where $n(w, j)$ is the j-th node on the path from the root to w and $L(w)$ the length of this path, for $n(w, 1) = root$ and $n(w, L(w)) = w$; $ch(n)$ is a fixed child of node n and $[\![x]\!]$ is 1 if x is true and -1 on the contrary; $\sigma(x) = 1/(1 + \exp(-x))$. Additionally, v_w has one representation for w and one representation v'_n for every inner node n of the binary tree. This approximation implies that $\Sigma_{w=1}^{W} p(w|w_I) = 1$, so the computation of $p(w_O|w_I)$ is proportional to $L(w_O)$, which is no greater than $\log W$. Mikolov et al. [110] use a binary Huffman tree to calculate the hierarchical softmax because of its assignments for short codes to frequent words, resulting in a more efficient training step. An alternative to hierarchical softmax is the negative sampling, described in [110]. The methodology used in CBOW is somehow similar to the one used for skip-gram, but instead of starting with a word w to predict its surroundings, we would consider the context to derive a target word.

Even though the traditional word2vec technique is robust enough in representing words in a vector space, the proposed techniques in Chapter III make use of a multi-sense notation. Thus, if for a given word there are multiple synonyms (polysemy), we want to represent all of those that are somehow related with the overall meaning of the document.

2.3.1 Multi-Sense Embeddings

In a standard single n-dimensional vector representation as word2vec, there will be just one embedding for each token to calculate any similarity measure between them. In a multi-vector representation, each word is associated with a set of senses, each with a vector representation. Hence, the similarity of these word-senses need to be calculated in a different way. Both representations make use of several benchmarks to evaluate their performance in NLP tasks (e.g. word similarity, document similarity). These benchmarks can be grouped into two categories: with or without context information. In the first category, a similarity score is given for two words in isolation, without any extra information about them. In the second category, each word is presented with a sentence to better help contextualize its semantic content.

Considering the multi-vector representation, Reisinger and Mooney [140] propose two metrics: *AvgSim* and *MaxSim*. In AvgSim, the word similarity is calculated considering the average similarity of all word-senses for the pair, as shown in Equation 2.4.

$$AvgSim(u, w) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} d(e(u, i), e(w, j)) \tag{2.4}$$

where u and w are the words to be compared; N and M are the total number of available senses for u and v, respectively; $d(e(u, i), e(w, j))$ is the similarity measure between the word-sense embeddings sets, denoted as $e(u, i)$ and $e(w, j)$, between the i^{th} sense of word u and j^{th} sense of word w. In MaxSim, the similarity is the maximum value among all pairs of word-sense embeddings, as illustrated in Equation 2.5. In this work, we consider the *cosine similarity* as the default similarity measure between any two vectors.

$$MaxSim(u, w) = \max_{1 \leq i \leq N, 1 \leq j \leq M} d(e(u, i), e(w, j)) \tag{2.5}$$

Reisinger and Mooney [140] also propose *AvgSimC* and *MaxSimC*. These take into account the similarity of two words when their context is available. In this scenario, the context is represented as sentences in which the target words are used. For tasks with this setup, two words are evaluated with respect to their similarity and each of them has a sentence illustrating their use. Both AvgSimC and MaxSimC are described in Equations 2.6 and 2.7 respectively.

$$AvgSimC(u, w) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} P(u, c_u, i)\ P(w, c_w, j)\ d(e(u, i), e(w, j)) \tag{2.6}$$

$$MaxSimC(u, w) = d(e_k(u, i), e_k(w, j)) \tag{2.7}$$

where $P(u, c_u, i) = d(e(u, i), c_u)$ is defined as the similarity of the i^{th} sense of word u with its context c_u. The context c_u is obtained by averaging all vectors of the words in the sentence where u is used. Different from single word vector representations, our model produces vectors for each word-sense, so when we calculate the average vector of c_u, we need only to consider all available word-sense vectors. $e_k(u,i) = arg_{max} d(e(u,i), c_u)$ is the maximum similarity obtained among all word-senses $e(u,i)$, with respect to its context c_u. All these terms are defined analogously for w and j as well. It is important to mention that the context defined for AvgSimC and MaxSimC are not related with the sliding context window presented in our approach (Section 3.2, or the sliding window in word2vec).

Huang et al. [73] also defend that word representations should be discriminated considering their surrounding words (*local context*) and their role in the entire document (*global context*). Their training model produces two vector types, one representing each word-sense and another for the word in the entire document, evaluated through *LocalSim* and *GlobalSim* respectively [122]. Unlike [73, 122], our approach does not produce global vectors during the training step, only local ones. Therefore, to obtain a global representation of a word, we average all word-sense vectors of u and w available to calculate their similarity, as Equation 2.8 shows.

$$GlobalSim(u,w) = d(\breve{\mu}(u,i), \breve{\mu}(w,j)) \tag{2.8}$$

where $\breve{\mu}(u,i)$ and $\breve{\mu}(w,j)$ represent the average of all word-sense vectors for u and w. As for LocalSim, we can use the original MaxSimC instead, since they work under the same assumptions [140].

2.3.2 Related Work in Word Embeddings

It is undeniable that word2vec's contributions with continuous skip-gram and CBOW from Mikolov et al. [109, 110] brought a legion of new publications to NLP, or more specifically, to the word embeddings field. Its popularity is due to, among other things, its efficient log-linear neural network language model and its low dimensionality vector representation. Both approaches produce vector representations of words, and those with similar contexts tend to have similar values. This theory is firstly introduced in the *Distributional Hypothesis* [64] and later refined by Firth [52], which states that "*a word is characterized by the company it keeps*". In the CBOW training model, one tries to predict a word given its neighboring context, while skip-gram does the converse, predicting the context given a target word. As influential as word2vec, Global Vectors (GloVe) [130] builds a co-occurrence matrix of the words in a corpus, which is based on word frequency. While word2vec is focused on fixed context windows to derive its vectors, GloVe takes advantage of a global perspective. As for their results, both word2Vec and GloVe present similar results in various NLP tasks [161, 75, 34]. Our approaches make use of word2vec's predictive model to train a disambiguated corpus into specific word-sense vectors that can be applied to several NLP tasks. This allows us to deal with one of the most important problems in traditional word embeddings techniques, the one vector representation per word property.

Extending word2vec's techniques (skip-gram and CBOW), Paragraph Vectors (PV) is an unsupervised framework that learns continuous distributed vector representations for any size of text portion (e.g. phrases, paragraphs, documents) [87, 36]. This technique alleviates the inability of word2vec to embed documents as a unique object. Differently than word2vec, PV produces a fixed length n-dimensional vector representation for each entire textual segment, instead of just the words in the corpus. Le and Mikolov [87]'s algorithm is also available in two different forms: Distributed Bag-of-Words of Paragraph Vectors (PV-DBOW) and Distributed Memory Model of Paragraph Vectors (PV-DM). The PV-DBOW works with the same principle as the skip-gram model, in which one tries to predict the context given a target word. In this approach, the context words are not considered as input, but instead, the model predicts words randomly sampled from the paragraph in the output. For this reason, there is no need to keep track of the word vectors of the context, causing this model to store less data. The PV-DM training model is similar to the CBOW approach in which the context is used to predict a target word. However, in both training models, an extra feature vector representing the text segment (called paragraph-id) is added to the context sliding window. The paragraph-id and word vectors are then averaged or concatenated to predict the target word. This forces the model to remember the missing words from the context when performing the prediction of these words. The paragraph-id vector is shared throughout the entire document for each sliding window, representing the meaning of the document as a whole.

Following the opposite direction of paragraph vectors, some publications move from a direct document representation to a sub-word one. In fastText [17], they extend the skip-gram model by proposing a word representation obtained from a sum of the n-grams of its constituent sub-word vectors. For example, using $n = 3$, the word *kiwi* would be represented as $\{ki, kiw, iwi, wi\}$ and the word *kiwi* itself, as a special case. Adopting a similar method, Peters et al. [131] propose to represent words from their constituent characters with Embeddings from Language Models (ELMo). ELMo uses a two layer deep bidirectional Language Model (biLM) with character convolutions as a linear function of their internal states. Due to its architecture, ELMo does not keep word vectors in their models, but only characters. This allows ELMo to handle any Out-of-Vocabulary (OOV) words, since its vectors are computed as an average of all biLM layers. More recently, the Universal Sentence Encoder (USE) [29] is proposed with two encoding models to represent any text as vectors, one focused on accuracy (transformer architecture) and the other on inference (deep averaging network). The former builds a sub-graph that uses attention mechanisms to compute the context representations of words in a text, considering their ordering and identity. The latter, average words and bi-grams embeddings, which are later used in a feed-forward deep neural network to produce new embeddings. In Section 4.3 we compare our proposed techniques, trained in a simple word2vec implementation, against the aforementioned state-of-the-art word embeddings models.

Encouraged by the robustness of word embeddings, Oele and Noord [124] combine word-sense, context, and word-definition embeddings to support their disambiguation system. They extend Lesk's algorithm [89] in two different scenarios using AutoExtend [145] as their embedding training algorithm, the first using a distributional thesauri, and the second using WordNet hierarchical structure. In [129], they propose a multi-stage system that learns single vector word representations, calculates word similarity graphs, infers word-senses using ego-network clustering, and aggregates word vectors with their prospective word-sense vectors. In contrast to these approaches, we use only single vector word embeddings to support our disambiguation process a single time. Once our annotated corpus is trained using a traditional word embeddings implementation, we can perform our disambiguation step considering the specific word-sense embeddings directly. In addition, we do not rely on any extra parameters, other than those required in a standard word2vec implementation. The WSD technique proposed in Section 3.2 is inspired by Ruas and Grosky [148] approach, which produces word-sense representations for a given word based on its context. Even though disambiguation is a crucial component in our techniques, the presented experiments and discussions focus more on how the combination of WSD and word embeddings can be mutual beneficial in the word similarity and document classification tasks [75]. We do have future plans to compare our WSD technique with alternative methods and see how this affects the overall experimental results in Sections 4.1 and 4.3, but for now this is beyond our scope. For more details of the WSD field, we suggest [120]'s survey, in which supervised, unsupervised, and knowledge-based approaches are discussed in depth.

2.3.3 Related Work in Multi-Sense Embeddings

Nearly all publications using single vector word embeddings suffer from the same problem, in that words having multiple senses are represented through a unique vector. In other words, polysemy and homonymy are not handled properly. For example, in the sentence "This *club* is great!" it is not clear if the term *club* is related to the sense of *baseball club*, *clubhouse*, *golf club*, or any other sense. Systems that use standard word embeddings, like word2vec or GloVe, will most likely represent all possible meanings for the term *club* in one single dimensional vector. Section 2.3.1 provides more information on how multi-sense embeddings are used and measured in NLP tasks.

Some researchers try to solve this representation limitation producing separate vectors for each word-sense. Even though the number of publications in this area is still small, their early findings demonstrate encouraging results in many NLP challenges [91]. As explained in Section 2.3.1, [73, 140] are one of the earliest publications that discuss multi-sense embeddings and how to properly measure them. Huang et al. [73] introduce a neural network language model capable of distinguishing the semantics of words, considering their global (entire document) and local (surrounding words) contexts. Reisinger and Mooney [140] follow a probabilistic approach to produce a multi-prototype vector space model, using word-sense discovery to evaluate a word's context. They set a number, K, of clusters to represent the different contexts where the word is used. We, on the other had, combine the prior knowledge of WordNet and word embeddings to extract the many meanings of a word in an unsupervised manner. Since we produce a vector representation for each word-sense, the global meaning of a word in a document is the average of all senses for that word. This way, we do not need to rely on cluster parameters, which would increase the complexity of our approach. Trask et al. [173] extend [73]'s model by leveraging supervised NLP labels, instead of relying on unsupervised clustering techniques to produce specific word-sense vectors. We follow a similar idea and let the words define their own senses according to the context where they are located. However, our approach also takes advantage of the lexical structure in WordNet, which helps us to identify the implicit relationships between the words. In addition, our model's results can be fed into a word embeddings technique and re-used to improve itself recurrently, with respect to the disambiguation and embeddings steps (Section 3.2.2).

Other techniques also take advantage of probabilistic models to learn their own representation for each sense. Tian et al. [172] design an efficient expectation maximization algorithm integrated with the skip-gram model to avoid the issues of clustering-based approaches. Neelakantan et al. [122] propose a modification for the skip-gram model in which they introduce the Multi-Sense Skip-Gram (MSSG). Their technique performs word-sense discrimination and embedding simultaneously, improving its training time. In the MSSG version, they assume a specific number of senses for each word, while in the Non-Parametric Multi-Sense Skip-Gram (NP-MSSG) this number varies, depending on the word. As in the NP-MSSG model, our approach also does not limit the number of word-senses for each word, but we use a CBOW implementation instead of the skip-gram training model to produce our word-sense embeddings. We also take advantage of WordNet's semantic network to help our system to better identify the possible senses for each word. Other publications expand pre-trained single word embeddings to obtain word-sense vectors [80, 133].

In multi-sense embeddings approaches the use of lexical resources to improve their performance in NLP tasks is quite common. WordNet[7], ConceptNet[8] and BabelNet[9] are examples of popular choices to help obtain more refined word-sense vectors. Based on BabelNet, the system of Iacobacci et al. [74] learns word-sense embeddings for word similarity and relation similarity tasks, moving from a word to a sense embedding representation. Our choice for WordNet is supported by its open source policy under any circumstances, which as in ConceptNet, is very attractive. However, WordNet is fully integrated with the Natural Language Toolkit (NLTK) in Python, which is heavily used in the implementation of our proposed techniques, making its choice preferable over other lexicographic resources, for now. Rothe and Schütze [145, 146] also use WordNet in their *AutoExtend* to produce token embeddings from a set of synonyms (i.e. synsets) and lexemes, using a pre-existing word embeddings model. Similar to our model, their approach is independent of any word-type representation, so it can be easily translated to other learning techniques. They assume that a word can be represented by the sum of its lexemes, so their methodology puts words, lexemes, and synsets in the same vector space. As in [145], we explore the use of synsets as

[7]https://wordnet.princeton.edu
[8]http://conceptnet.io
[9]https://babelnet.org

well, but beyond that we take into account their respective glosses[10], which is not considered in their model and aggregates solid information to the disambiguation process. As a result, our semantic representation obtains better results when the context is available in the word similarity task (Section 4.1.5). In the iterative version of MSSA (Section 3.2.2), we can also use our produced vectors to improve the disambiguation step. Additionally, our annotated corpus is self-contained with respect to its representation in the lexical database. In other words, from the annotation results or the word embeddings model keys, one can retrieve the lexeme, synset, word or any other information available in WordNet for a specific word (Section 3.2.4).

Continuing with the multi-sense embeddings approaches, Camacho-Collados et al. [27] propose a powerful semantic vector representation called NASARI which is extended to a multilingual scenario. Their system uses BabelNet as a lexical database and provides vector representations of concepts in several different languages in a unified semantic space. This approach combined with all lexicons incorporated in BabelNet gives NASARI a robust architecture. As in our approach, SensEmbed [74] produces word-sense vectors based on a disambiguated and annotated corpus. However, their disambiguation process relies on Babelfy [117], which combines WSD and entity linking to build a dense graph representation of sense candidates of each word, using BabelNet as a backbone. More recently, Mancini et al. [99] associate words to the most connected senses in a sentence to produce their embeddings. In their approach, sentences are parsed and only word-senses with more than a certain number of connections with other words in the same sentence are selected. These connections illustrate the relationships (edges) between synsets (nodes) in BabelNet. In the end, both word and sense embeddings are represented in the same vector space. Differing from all these approaches, we only rely on the training corpus available for the disambiguation step in combination with WordNet. No other multi-lexicographic databases are used nor extra hyperparameters considered to incorporate external information

[10]https://wordnet.princeton.edu/documentation/wngloss7wn

about the word-sense. In addition, our techniques (Sections 3.2.1 and 3.2.3) produce word-sense embeddings that can be used to improve the WSD process in a recurrent manner (Section 3.2.2).

The system of Chen et al. [32] performs WSD on the words and uses them to learn word-sense representations from the relevant occurrences through two approaches: L2R (left to right) and S2C (simple to complex). In L2R, they disambiguate words from left to right, whereas S2C selects only word-senses that reach a certain similarity threshold to represent a word. We believe that considering just one order in the disambiguation step or only specific word-senses leads to a poor and biased representation of semantics. In MSSA, we explore the contextual effects of all senses available for a word and its neighbors in a bi-directional fashion. This prevents us from ignoring possible candidates even if they are not that frequent in the text. Chen et al. [31] use WordNet's glosses to produce word-sense vectors via a CNN, which are used as input into an MSSG variant. Their approach uses the sentence in the glosses as positive training data and replaces random words (controlled by a parameter) to create negative training data, in order to minimize a ranking loss objective. Our technique, on the other hand, takes a much simpler approach, averaging the vectors of all the words in the glosses and using this representation to disambiguate word-sense candidates. In addition, there is no extra training or hyperparameter adjustments other than those required in a standard word2vec implementation.

2.3.4 Related Work in Document Classification and Embeddings

In a different direction, some researchers [43, 83, 53, 165] try to improve the quality of pre-trained word embedding models instead of generating a specific model for each training corpus. Sinoara et al. [165] propose to enhance document embeddings representations, gathering knowledge from different word embedding models. As a pre-processing step, they use Babelfy [117] to transform each word in a dataset into synsets, which have their vectors retrieved from a pre-trained embeddings model obtained from NASARI [27]. In case their NASARI embeddings model does not have a vector for a specific synset, they look for it in a traditional pre-trained word2vec model. The main idea behind their approach is to use NASARI as a multilingual word embeddings model, allowing them to work with any kind of language covered in this model. They compare their approach against the traditional word2vec model, LDA [16], and classic BOW. Their reported results show an incremental improvement over the baselines on the document classification task. Our approach for lexical chains (Section 3.3), only uses one word embeddings model to produce its vectors, which is obtained directly from the corpus in where they are derived. Moreover, our lexical chains connect multiple words that share a common semantic relation in a sentence, reducing the number of tokens that need to be embedded.

In [53], they propose a technique to generate document representations called Bag-of-Meta-Words (BoMW). To build the BoMW they represent each document as an average of its constituent word vectors, which are obtained through a pre-trained word embeddings model (e.g. word2vec, GloVe). Then, they map each vector into a different feature space and sum the vectors from this new space to represent the document. They propose two approaches for the mapping, namely Naïve Interval Meta-Word Model (NIM) and Features Combination Meta-Word Model (FCM). The former is based on the discretization of features from the initial word embedding, while the latter clusters related features under the same group. They compare their results on three datasets against BOW, neural networks models (e.g. convolutional neural network, recurrent neural networks), and an average of word embedding models. Their approach shows an improvement in accuracy over the baselines on the document classification task.

Some techniques make use of context information in the document to incorporate semantic aspects into traditional word embeddings models. In [83], they build content tree word embeddings that captures local information from the words in a document. Their word vectors are calculated as a weighted average of a word's vector and its immediate parent. They assume that the context, represented by a word parent node, has influence over its neighboring words. The insertion of a new word into an existing content tree only happens if there is a high correlation between all nodes in that tree and the new word. Enríquez et al. [43] explore the complementary information of BOW and word2vec to represent their document. Their technique is based on a simple voting system that averages the confidence values returned from BOW and word2vec to classify a document either in a negative or positive class. They conclude that BOW provides the best representation over word2vec, while their combination improves the overall results.

To the best of our knowledge, this is the first work that combines lexical chains and word embeddings applied to the document classification problem. We expect the proposed algorithms to produce a robust semantic representation through the use of WSD and lexical chains. Furthermore, our lexical chains are built using the most relevant synset attributes in the lexical database, as opposed to just hypernyms and hyponyms, commonly used in the literature [5, 6, 179, 58, 100, 164]. The main idea is to bring the semantic relations of lexical chains to traditional word embeddings techniques, leveraging their vector representation and improving the overall result in NLP problems, such as word similarity and document classification.

CHAPTER III

Exploring Multi-Sense Embeddings and Lexical Chains

The meanings of words in a text are largely dependent on their context in the document in which they appear, just as particular situations serve as connections to interpret human personality. The meaning of a sentence in a document is more easily determined if its constituent words exhibit cohesion with respect to their individual semantics. In this chapter, we present the proposed techniques and algorithms for finding the degree of cohesion among words and documents in three different layers, as follows. Section 3.1 provides an overview for the entire system and how its internal components connect with each other. In Sections 3.2 and 3.3 the constituent elements for the proposed architecture are explored in depth for a better understanding of their particular strengths and weaknesses.

3.1 Semantic Architecture Representation

The overall architecture proposed in this work consists in extracting semantic features from a collection of documents, to help solve problems involving NLU. The entire scenario is composed of four major stages, as Figure 3.1 shows: Most Suitable Sense Annotation (MSSA) algorithms, Lexical Chains (LC) algorithms, model generation, and natural language problems. The MSSA algorithms (Section 3.2) are responsible for disambiguating and annotating each word according to their most suitable sense, considering the effects of their neighboring words. As a result, a given word corpus is transformed into a specific synset corpus. In the lexical chains algorithms (Section 3.3), the produced synset corpus is used as an input so that the semantic relations between synsets can be extracted. During the model generation step, the output of both MSSA and the lexical chains algorithms are used to train a synset embeddings model in a word2vec implementation, so that synset and lexical chains vectors can be produced. Finally, the vector representation of synsets and lexical chains are used as features to solve NLP tasks, such as word similarity and document classification.

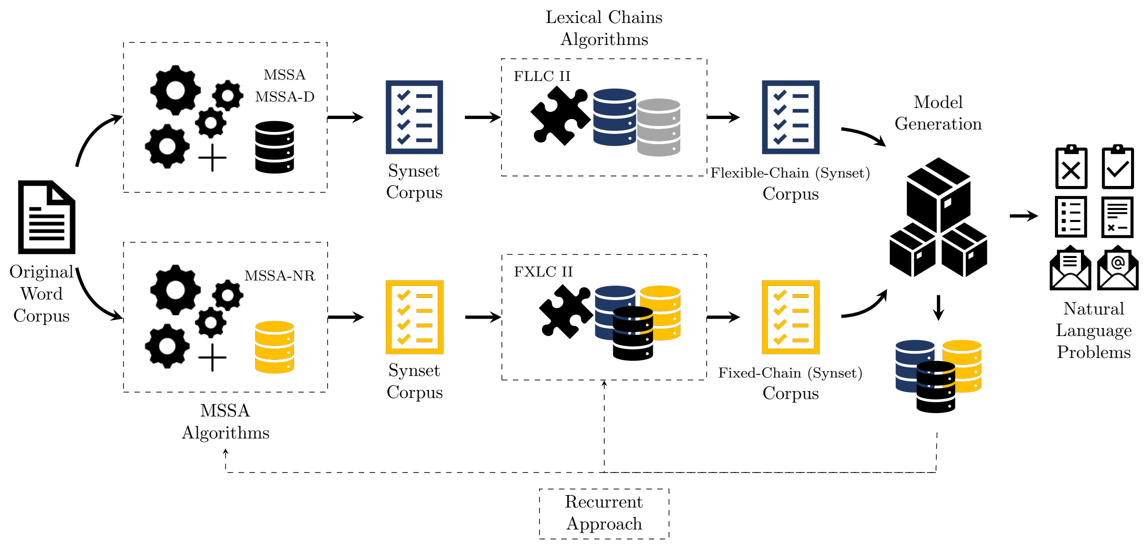


Figure 3.1: System architecture for extracting semantic features using MSSA and lexical chains algorithms.

The MSSA step is comprised of three techniques: Most Suitable Sense Annotation (MSSA), Most Suitable Sense Annotation - Dijkistra (MSSA-D), and Most Suitable Sense Annotation - N Refined (MSSA-NR). These combine the benefits of word embeddings and word-sense disambiguation in a mutual self-improving system. The approaches in this phase can make use of two types of embeddings, word-based (black) or synset-based (yellow). In addition, the prior knowledge from an external lexical database is also incorporated in the process. For the lexical chains step, two other algorithms are proposed, called: Flexible Lexical Chains II (FLLC II) and Fixed Lexical Chains II (FXLC II). While the former works with a synset-based embeddings model (blue) and a lexical database (gray), the latter can work with any kind of pre-trained embeddings model (blue, yellow, black), as long as it matches its input representation. The lexical chains algorithms find the semantic relations between consecutive tokens in a dynamic way (FLLC II) or a pre-defined structured manner (FXLC II).

The model generation activity can be performed using any of the outputs produced during each stage in the proposed techniques. Using a standard word embeddings implementation (e.g. word2vec, GloVe) a synset-based vector representation is obtained and can be used recurrently in any of the previous processes.

For each technique, specific NLP downstream tasks are selected to validate the proposed algorithms. The MSSA variations are first tested in a word similarity task (Section 4.1), and the ones with the best performance are carried out to be evaluated in the document similarity task, along with the LC algorithms (Section 4.3). Additional experiments for the early versions of the proposed techniques can be found in Section 6.3.

The next sections of this chapter are used to explain in detail the proposed approaches and their internal components.

3.2 Synset Disambiguation, Annotation, and Embeddings

The main idea of our process is to have a modular system with two independent tasks: (i) disambiguation followed by annotation, and (ii) token embeddings training. This configuration allows us to incorporate more robust techniques in the future, especially for the training step. The disambiguation and annotation module require nothing more than a lexical database, and a compatible token embeddings model to transform word-based documents into word-sense-based documents. As for the embeddings training module, any word embeddings algorithm that can represent tokens in a vector space is suitable. In the following sections we explain the details of our approach, illustrated in Figure 3.2.

In the first task, we process a collection of articles (documents) from two Wikipedia Dumps to transform each word in the corpus into a synset using WordNet as our lexical resource [48, 112]. This is done through one of the proposed algorithms: MSSA

(Section 3.2.1), MSSA-NR (Section 3.2.2), or MSSA-D (Section 3.2.3). In the second task, we use a word2vec implementation [109, 110] to train this synset corpus and obtain n-dimensional vectors for each word-sense (multi-sense embeddings).

In their initial form, both MSSA and MSSA-D use Google News vectors[1] to help disambiguate the word-senses in the corpus. MSSA works locally, trying to choose the best representation for a word, given its context window. MSSA-D on the other hand, has a more global perspective, since it considers the most similar word-senses from the first to the last word in a document.

For the word embeddings training module. Once the synset embeddings models are available, we can feed them to the system again, using the output vectors from the previous pass, and improve the disambiguation step in either the MSSA or the MSSA-D algorithms, relieving them from the original Google News vectors' dependency. We call this approach MSSA-NR, where N is the number of feedback iterations used. This recurrent characteristic is not explored in any of the related works (Section 2.3) nor the compared systems in the experiments (Section 4.1). Different from other systems [32, 31, 145], our method has only one training phase and does not rely on any extra hyperparameters, other than those required in the original word2vec implementation. In addition, since all proposed MSSA approaches are performed in the raw text directly and prior to any training model, they can be easily incorporated into any NLP pipeline, independently of the task. In other words, MSSA would work the same way as any common pre-processing activity (e.g stemming, stopwords removal, lowercase).

3.2.1 Most Suitable Sense Annotation (MSSA)

As Ruas and Grosky [148] present, each evaluated word w_i takes into consideration its context, represented by its surrounding neighboring words, w_{i-1} and w_{i+1}, as Algorithm 1 shows. We also use WordNet as our lexical database to extract all synsets from each word in

[1]https://code.google.com/archive/p/word2vec/

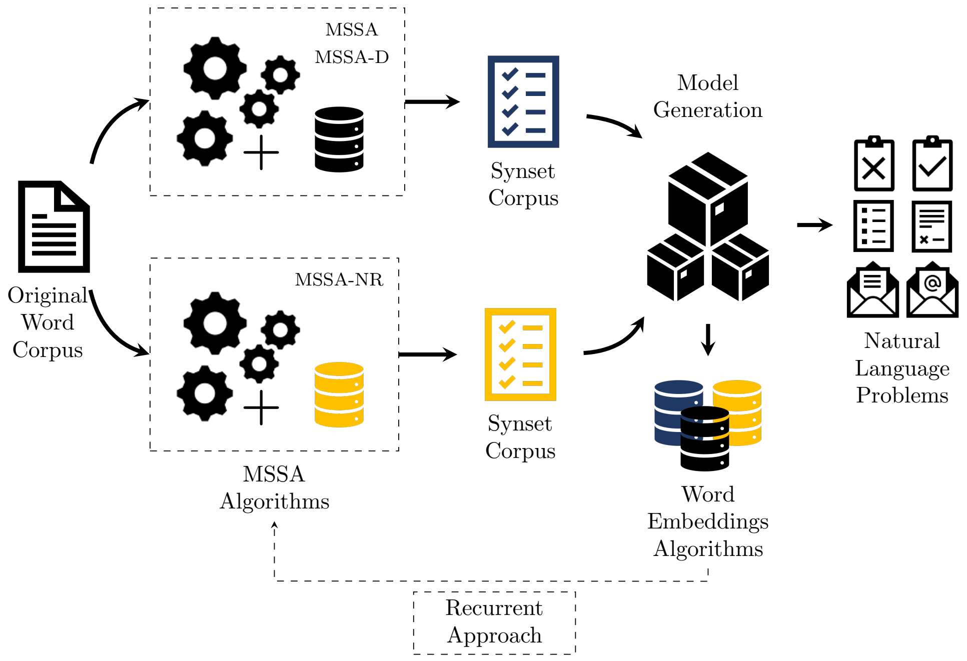


Figure 3.2: System architecture of MSSA, MSSA-D and MSSA-NR.

the text, but unlike [148], our algorithm works for any word and POS mapped in WordNet, not just for nouns. In our approach, all text is first pre-processed, normalizing all tokens in lowercase, removing punctuation, html tags, numbers, common English stopwords, and discarding all words not present in WordNet. The list of common stopwords used is obtained directly through the NLTK library in Python. After this initial data cleaning, we extract all pairs of synsets and glosses for each word w_i in a sliding context window of 3 words (lines 3:13). Our context sliding window is similar to the one used in CBOW [109], which uses the context to predict a given word. However, since our algorithm considers all synsets from w_i, w_{i-1}, and w_{i+1}, we currently limit this word context window to restrict the necessary number of comparisons, so as to infer the most suitable meaning for w_i. It is in our plans to incorporate a larger context without compromising the overall performance for this step. Next, after removing common English stopwords from the glosses, we retrieve and average the embeddings from the remaining tokens in each gloss, which we call *gloss-average-vector*, using Google News pre-trained vectors[2]. If there are no remaining tokens in the gloss or no

[2]https://code.google.com/archive/p/word2vec/

vectors in the model, an empty vector will be assigned for that synset-gloss pair. However, this scenario is very unlikely, since the words in the glosses have their vector extracted from a model trained on a huge corpus. This process is done for all synset-glosses for each element s_c ($current\_candidates$), s_f ($former\_candidates$), and s_l ($latter\_candidates$) (lines 15:18), where M, N and P represent the total number of available synset-glosses per synset, respectively. After the gloss-average-vectors for each synset in a particular position of the sliding window are obtained, we calculate the cosine similarity of all synsets of the $current\_candidates$ against those of the $former\_candidates$ and the $latter\_candidates$, returning the synset for $current$ (in each case) with the highest score, as lines 19 and 20 describe. Finally, we add the synset with the highest value to our list of tokens, in order to represent this occurrence of w_i in our new synset corpus (line 21). It is important to mention that the first and the last words in our corpus are treated differently (lines 8 and 11), since they do not have a complete context window available, similar to the sliding context window in word2vec.

Algorithm 1 Most Suitable Sense Annotation (MSSA)

```
Require: $d = \{w_i, ..., w_n\} : w_i \in$ lexical database (WordNet)
Require: $tm$ = trained word embedding model
Require: $ld$ = lexical database
 1: function MSSA($d$, $tm$, $ld$)
 2:     list_of_tokens = $\emptyset$
 3:     for $i = 0$ to $n$ do
 4:         current = synset-glosses($w_i$, $ld$)
 5:         if $i \neq 0 \wedge i \neq n$ then
 6:             former = synset-glosses($w_{i-1}$, $ld$)
 7:             latter = synset-glosses($w_{i+1}$, $ld$)
 8:         else if $i = 0$ then
 9:             former = $\emptyset$
10:             latter = synset-glosses($w_{i+1}$, $ld$)
11:         else
12:             former = synset-glosses($w_{i-1}$, $ld$)
13:             latter = $\emptyset$
14:         current_candidates = $\emptyset$, former_candidates = $\emptyset$ and latter_candidates = $\emptyset$
15:         for $s_c \in \{current\}$, $s_f \in \{former\}$ and $s_l \in \{latter\}$ do    ▷ where $0 \leq c \leq M$,
    $0 \leq f \leq N$ and $0 \leq l \leq P$
16:             Add gloss-avg-vec($s_c$, $tm$) to current_candidates
17:             Add gloss-avg-vec($s_f$, $tm$) to former_candidates
18:             Add gloss-avg-vec($s_l$, $tm$) to latter_candidates
19:         $u = argmax_{s_{c1}}$ {cosine-similarity($current\_candidates$,$former\_candidates$)}
20:         $w = argmax_{s_{c2}}$ {cosine-similarity($current\_candidates$,$latter\_candidates$)}
21:         Add the synset ($s_{c1}$ or $s_{c2}$) with the highest produced cosine similarity to list_of_tokens

22:     return list_of_tokens
```

In the initial configuration, we use Google News vectors as our standard word embeddings model (tm in lines 15:18), which is trained over 100 billion words and contains 300-dimensional vectors for 3 million unique words and phrases [110]. This approach can also work recurrently, using the current synset embeddings to be fed back into our system, so that in the next iteration we use our previously calculated vectors of synsets to disambiguate the word-senses in the corpus. In this modality, it is not necessary to calculate the gloss-average-vector for each synset-gloss again, since we can use the synset embeddings directly to disambiguate our training corpus.

3.2.2 Most Suitable Sense Annotation N Refined (MSSA-NR)

As mentioned earlier, once we have trained our model based on synset tokens, we can use these output synsets vectors directly for another pass of our algorithm, bypassing the gloss-average-vector calculation. As we do not need to calculate the gloss-average-vector for each recurrence after the first one, each future pass will take less time than the first pass. We hypothesize that using a disambiguated and granular embeddings training model will produce a more refined one. The algorithm for this approach is similar to the one presented in Section 3.2.1, so we are still using the same cleaned training corpus composed of Wikipedia articles, but some of the steps are slightly different.

We identify this recurrent approach as MSSA-NR, where N represents the number of feedback iterations used. Algorithm 2 starts in a similar fashion to Algorithm 1, as we also use WordNet as our lexical database and still work with the same sliding context window for the words. The main difference occurs between lines 3 and 13, where, since our embeddings consist of synsets, we do not need to extract the pairs of synset-glosses and calculate the gloss-average-vector for each synset. Instead, we just extract all synsets available in WordNet for w_i ($current$), w_{i-1} ($former$), w_{i+1} ($latter$) and directly retrieve their respective vector embeddings from the synset model trained (lines 3:13), where Q, R and S represent their total number of available synsets per word. Since MSSA-NR is using an embeddings model on the same corpus on which it was first generated, all the words will have at least one synset mapped, so there is no risk of not finding a vector for a given word-sense. After we retrieve the vector values for all synsets in the sliding window, we calculate the similarity of $current\_candidates$ against $former\_candidates$ and $latter\_candidate$, returning the synsets for $current\_candidates$ with the highest value in each case (lines 19 and 20). As in MSSA, we also select the synset with the highest score to represent w_i in our new synset corpus (line 21)

Because we are using the word-sense embeddings from our previous pass, senses that never were selected to represent any word in the original corpus will not have a vector representation in our model. As a consequence, in the next iteration, these word-senses do not have to be verified, since they were not embedded in the first place. The hope is that, over many passes, the non-used word-senses are dropped and results will converge to some stable synset-value representation of our corpus. This will also contribute to a faster processing time, if compared to the plain MSSA approach, considering that the number of word-senses is reduced on each pass until it stabilizes. We can stop the process after a finite number of passes, when we are satisfied that the results do not change much, or when the cost incurred for running another pass of the algorithm is too high to justify another disambiguation and annotation round. More details about the overall complexity are provided in Section 3.2.5.

Algorithm 2 Most Suitable Sense Annotation N Refined (MSSA-NR)

```
Require: $d = \{w_i, ..., w_n\} : w_i \in$ lexical database (WordNet)
Require: $tsm$ = trained synset embedding model
Require: $ld$ = lexical database
 1: function MSSA-NR($d$, $tsm$, $ld$)    ▷ where $d$ - document containing words $w_n$, $tsm$ - trained synset embedding model, $ld$ - lexical data base
 2:     list_of_tokens = $\emptyset$
 3:     for $i = 0$ to $n$ do
 4:         current = synsets($w_i$, $ld$)
 5:         if $i \neq 0 \wedge i \neq n$ then
 6:             former = synsets($w_{i-1}$, $ld$)
 7:             latter = synsets($w_{i+1}$, $ld$)
 8:         else if $i = 0$ then
 9:             former = $\emptyset$
10:             latter = synsets($w_{i+1}$, $ld$)
11:         else
12:             former = synsets($w_{i-1}$, $ld$)
13:             latter = $\emptyset$
14:         current_candidates = $\emptyset$, former_candidates = $\emptyset$ and latter_candidates = $\emptyset$
15:         for $s_c \in \{current\}$, $s_f \in \{former\}$ and $s_l \in \{latter\}$ do    ▷ where $0 \leq c \leq Q$, $0 \leq f \leq R$ and $0 \leq l \leq S$
16:             Add synset-vec($s_c$, $tsm$) to current_candidates
17:             Add synset-vec($s_f$, $tsm$) to former_candidates
18:             Add synset-vec($s_l$, $tsm$) to latter_candidates
19:         $u = argmax_{s_{c1}}\{$cosine-similarity($current\_candidates$,$former\_candidates$)$\}$
20:         $w = argmax_{s_{c2}}\{$cosine-similarity($current\_candidates$,$latter\_candidates$)$\}$
21:         Add the synset ($s_{c1}$ or $s_{c2}$) with the highest produced cosine similarity to list_of_tokens
22:     return list_of_tokens
```

3.2.3 Most Suitable Sense Annotation - Dijkstra (MSSA-D)

We also propose another variation for the MSSA algorithm, in which we model the documents in the corpus as a graph $Doc_k(N, E)$, where Doc_k is the set of k documents; N is the set of nodes, represented as word-senses (synsets) and E is the set of edges associating two nodes in document k. Inspired by Dijkstra's algorithm [40], we use a modified version of it to minimize the overall cost of moving from one node (synset) to another, for all the words in the document. The weights on the edges are the *cosine distance* (1 - *cosine similarity*) between the gloss-average-vector of two sequential word-senses. All the steps in the MSSA-D design are the same as the ones presented in Section 3.2.1 for MSSA (Algorithm 1), with the exception that there is no sliding context window for the disambiguation part. Different from MSSA, in MSSA-D we analyze the disambiguation problem globally, looking for the shortest distance from one word-sense to the next. Figure 3.3 illustrates a toy example of five words in which the highlighted path has the lowest cost, considering their word-senses $\omega_{n,m}$, where n is the associated word position and m its respective sense. In the end, the objective of this algorithm is the same as the ones presented in Sections 3.2.1 and 3.2.2, transforming a training corpus composed by words into a corpus of synsets to be trained in word2vec.

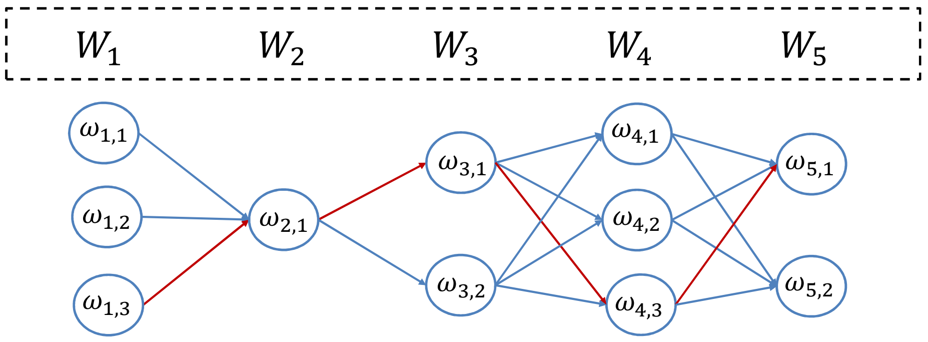


Figure 3.3: MSSA-D illustration of the shortest path from w_1 to w_5 through their respective word-senses.

As in MSSA, it is also possible to apply the MSSA-NR recurrent methodology into MSSA-D and reuse the produced synset embeddings instead of the Google News vectors.

Considering this approach, there is, again, no need to calculate the gloss-average-vector for each word-sense. Instead, we can directly use the synset vectors available. In Section 4.1, we describe the different setups used in our experiments to explore all of our techniques.

3.2.4 From Synset to Embeddings (Synset2Vec)

After all words are properly processed into synsets, we use them as input in a word2vec implementation with CBOW as the training algorithm. This choice is justified, due to its popularity among the compared systems, with its reported superiority in performance, and its ability in capturing the context of words in large datasets [17, 109, 185, 186].

In all MSSA variations, the goal is to transform a word-based document into a synset-based one. This will allow us to properly represent a word with multiple senses. Since the disambiguation step might consider all the senses of a word, its cost grows rapidly with the size of the documents and number of available senses per word. For a small to medium size training corpus, this is not a barrier, but for larger ones, such as a Wikipedia Dump used in our paper, this can demand a high amount of processing time. On the other hand, the number of tokens to be trained in word embeddings algorithms is reduced, since only words that exist in WordNet are considered. In addition, the disambiguation process only needs to be executed once and it can be done in parallel for all MSSA techniques. Once the annotated synset training corpus is performed, one can use it in any desired activity, such as word similarity, document classification, text summarization, and train a word embeddings model.

To keep our vectors interpretable - as pointed out in [125] - across different platforms, we represent each word token as a key in the following format: $word\#synset\_offset\#pos$, where $word$ is the word itself, normalized in lowercase; $synset\_offset$ is an 8 digit, zero-filled decimal integer that corresponds to a unique word-sense, and pos is a part-of-speech tag (e.g. n for nouns, v for verbs, a for adjective, s for adjective satellite and r for adverb)[3]. Since we have independent tasks for annotation and word embeddings training, if more robust techniques are proposed in the future, we can easily incorporate them.

3.2.5 Complexity Analysis

In this section we provide a detailed explanation on how to compute the average time complexity of each of our algorithms, MSSA, MSSA-NR, and MSSA-D.

For each of these algorithms, we have the same pre-processing step; namely, to associate a vector with each gloss in the current version of WordNet. However, this is done only once per WordNet version. In the current version of WordNet, assume that there are S synsets, each having one gloss. For a synset S_i, let g_i be the number of words in its gloss, after eliminating common English stopwords. Then the average number, G, of words/gloss is described as $G = \frac{(g_i + \cdots + g_S)}{S}$. The vector associated with each word in a gloss is assumed to be the Google News vector (i.e. pre-trained word embeddings model) associated with that word and we assume that this vector can be found in $\theta(1)$ time. Given the Google vectors for each word in a given gloss, the vector associated with this gloss is just the average of the vectors from the pre-trained model. Thus, for synset S_i, the time complexity of computing the vector associated with its gloss is $\theta(g_i)$. We then have that the time complexity for computing the vectors associated with all glosses in WordNet is $\theta(g_i + \cdots + g_S) = \theta(SG)$.

Considering MSSA, let us calculate the time complexity of computing the disambiguated synset corpus from the original one. Let W be the number of words in the corpus. There are, on the average 2.89 synsets per word in WordNet[4]. Thus, $2.89W$ is a good approximation for the number of synsets processed by our algorithm. Given that each synset has one gloss vector and defining gv_i as the number of gloss vectors processed for word w_i of the corpus, we then get that $(gv_1 + \cdots + gv_W) = 2.89W$. Based on lines 14:21 in Algorithm 1, the number of processed vectors are illustrated in Equation 3.1.

[3]https://wordnet.princeton.edu/documentation/wndb5wn
[4]https://wordnet.princeton.edu/documentation/21-wnstats7wn

$$\begin{aligned}(gv_1gv_2 + gv_2(gv_1 + gv_3) + gv_3(gv_2 + gv_4) + gv_{W-1}(gv_{W-2} + gv_W)+ \\ (gv_{W-1}gv_W) \leq (gv_1 + \cdots + gv_W)^2 \leq 2.89^2W^2 = \theta(W^2)\end{aligned} \tag{3.1}$$

Thus, we compute $O(W^2)$ cosine measures overall in the computation of the corresponding synset corpus. Assuming that $w2v$ (word2vec) is the running time of the standard approach to calculate the Google vectors using the CBOW approach, we have that the time complexity of MSSA is $O(W^2) + w2v$.

Now, let us consider MSSA-NR. By the argument above, it is seen that the first pass of MSSA-NR has time complexity $O(W^2) + w2v$. For each succeeding pass, we follow the same process as above, but with the synset vectors from the previous pass replacing the corresponding gloss vector. In this case, for N passes, the time complexity is $O(NW^2) + N * w2v$.

Finally, for MSSA-D, the underlying graph has V vertices and E edges. As shown above, $2.89W$ is a good approximation for the number synsets processed by our algorithm. Thus, $V = 2.89W$. Realizing that gv_i is also equal to the number of synsets processed for word w_i of the corpus, we have that the number of edges can be described as Equation 3.2 shows.

$$\begin{aligned}E = gv_1gv_2 + gv_2gv_3 + \ldots gv_{W-1}gv_W + (gv_{W-1}gv_W) \\ \leq (gv_1 + \cdots + gv_W)^2 \leq 2.89^2W^2\end{aligned} \tag{3.2}$$

Thus, since the time complexity of Dijkstra's Algorithm is $O(V^2)$, $O(V^2 + E\log V)$ or $O(E + V\log V)$, depending on its implementation, we have that the time complexity of MSSA-D is $O(W^2)$.

3.3 Extending Lexical Chains

As explained in Section 2.2.1, lexical chains are built according to a series of relationships between words in a text document. In the seminal work of Morris and Hirst [118], they consider an external thesaurus (Roget's Thesaurus [143]) as their lexical database to extract these relations. A lexical chain is formed considering a sequence of words $\{w_1, w_2, \ldots, w_n\}$ appearing in this order, such that any two consecutive words w_i, w_{i+1} possess the following properties[5] [118]:

- Two words share one common category in their index;
- The category of one of these words points to the other word;
- One of the words belongs to the other word's entry or category;
- Two words are semantically related; and
- Their categories agree to a common category.

As for the quality of a built chain, three factors can also be considered to measure how strong a lexical chain is [118]: *reiteration*, *length*, and *density*. Reiteration illustrates how often a lexical chain occurs, the length is related to the number of tokens on it, and density is the ratio of words in the lexical chain to the words in the text document. Since our main objective is to evaluate how the proposed lexical chains perform in a downstream task, we leave the calculation of their strength to future work. Thus, we consider the five characteristics defined in [118] to build our lexical chains and validate their quality in the document classification problem.

In this collection of algorithms, we propose a novel approach to capture the semantic relationship between tokens from a text. Our techniques combine the use of word embeddings, lexical chains, and the prior knowledge of lexical databases, to derive the relations between words. This is done through two algorithms: Flexible Lexical Chains II (FLLC II) (Section 3.3.1) and Fixed Lexical Chains II (FXLC II) (Section 3.3.2). Both algorithms are inspired by the approaches proposed in [148, 149]. While the lexical chains explored in [148, 149] consider only one POS (nouns) and hypernyms, ours, on the other hand, are

[5]Where category, indexes, and pointers are attributes in the lexical database considered.

able to deal with any POS tag, incorporate word embeddings, and also include 19 other lexical synset attributes (e.g. hyponyms, meronyms) from the English WordNet [48]. As a result, we leverage the semantic representation of words, sentences, paragraphs, and entire documents through the use of lexical chains.

The main goal of FLLC II and FXLC II is to represent a collection of words using their semantic values in a more concise and robust way. Even though FLLC II and FXLC II outputs are the same, they explore different aspects when capturing the lexical cohesion in a text. In the FLLC II version, the semantic sets (lexical chains) are assembled dynamically according to the semantic content of each token evaluated and the relationship with its adjacent neighbors. As long as there is any semantic affinity (i.e. a semantic relation that connects two words in a lexical database) they should be integrated into one single entity that represents a single concept. If a word without any semantic affinity with the current chain being built presents itself, a new lexical chain must be started so a new concept can be captured. On the other hand, in the FXLC II approach text documents are broken down into pre-defined chunks, with C words each, that will hopefully describe the semantic content of them. Different from the FLLC II algorithm, the FXLC II technique groups a certain number of words into the same structure, regardless of the existence of semantic affinity or not.

An overview for the lexical chains process is illustrated in Figure 3.4. In both cases, FLLC II and FXLC II, the constructed chains are represented through one of their constituent elements, which is selected considering their vector representation in a pre-trained token embeddings model (e.g. word2vec). Later, we feed the output of our techniques into a word embeddings algorithm which produces a lexical chain embeddings model that is used in the document classification task.

The FLLC II and FXLC II techniques are, at their core, reducing the number of valid elements (words) in a document. Given the aforementioned characteristics, we can classify the FLLC II algorithm as a soft-dimension reduction technique, while FXLC II is characterized

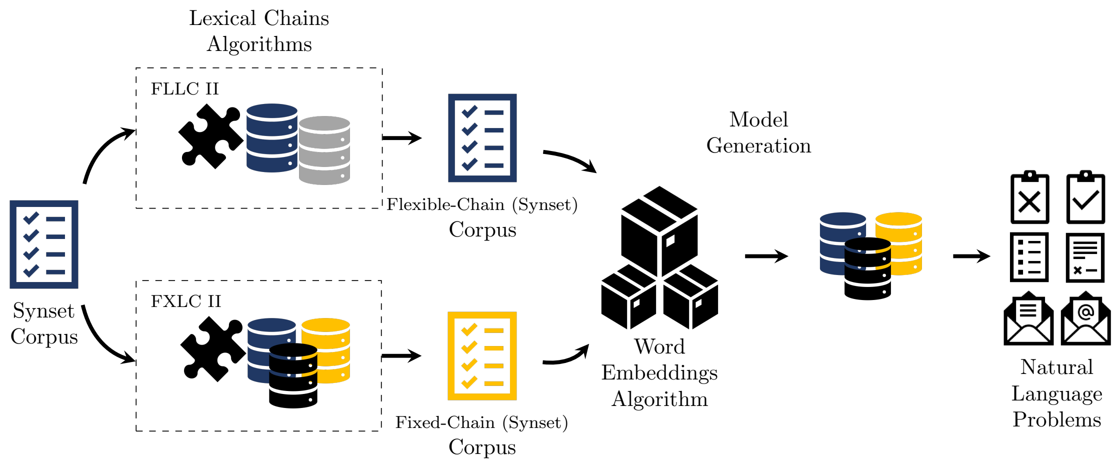

Lexical Chains Algorithms
FLLC II
FXLC II
Synset Corpus
Flexible-Chain (Synset) Corpus
Fixed-Chain (Synset) Corpus
Model Generation
Word Embeddings Algorithm
Natural Language Problems

Figure 3.4: System architecture for building lexical chains.

as a hard-dimension reduction one. Another important difference between the proposed techniques is with respect to their databases and how they are used. FLLC II chains are generated using a lexical database (e.g. WordNet) and a pre-trained token embeddings model represented by the gray and blue diagrams in Figure 3.4 respectively. While the former is responsible for providing the semantic relationship between the tokens in the document, the latter helps us to decide which token will represent our chain. In both algorithms, the blue diagram represents a trained synset embeddings model[6]. In the FXLC II technique, since we do not group tokens directly considering their semantic affinity, we do not require a lexical database with word relationships, such as WordNet. Instead, we just need a compatible pre-trained token embedding model, with respect to the document being processed, to build our chains. Therefore, FXLC II can be exported to other scenarios using any pre-trained token embeddings model (e.g. words, synsets, xml-tags), as long as it contains the same token elements from the documents. In the next sections, both techniques are explored in depth, followed by a simple toy example illustrating their operation.

[6]A word embedding algorithm using a synset corpus.

3.3.1 Flexible Lexical Chains II (FLLC II)

The FLLC II algorithm works building semantic sequences of words (or any other tokens) that present any level of semantic similarity between them. The decision of incorporating, or not, a new word into a chain is dynamic and based on 19 lexical synset attributes extracted from WordNet [48].

We extend Ruas and Grosky [148] flexible chain algorithm to use all POS and also incorporate word embeddings containing the vector representation of the tokens in a document d. As illustrated in Algorithm 3, we require a document represented as synsets [1], a lexical database (e.g. WordNet, BabelNet) ld, and a pre-trained synset embeddings model tsm. To transform a word-based document into a synset-based one, we apply the MSSA algorithm (Section 3.2). In a few words, the MSSA algorithm works disambiguating a given word, with respect to its word-senses and considering the effect of its immediate neighbors.

Once we have all the words in a document represented as synsets, we start building our flexible chains from the first to the last token. Initially, we start our current chain using S_1 (first synset in the document) to initialize the synset list that will be used to represent the current chain, named $current\_chain.synsets$, and a set of related synsets that are used to map the semantic relation between consecutive synsets, called $current\_chain.related$ (lines 3:4). The synsets retrieved in $get\_related\_syns(S_i, ld)$ (including S_i) form a collection of 19 synset attributes in the lexical database ld, which in our case is WordNet.

For each new synset evaluated, S_i, we extract their related synsets (including S_i), called new_rel, and verify if there are any common synsets with the related synsets in the chain being built ($current\_chain.related$) (lines 5:7). In case the intersection between $current\_chain.related$ and new_rel is not empty, we add S_i and new_rel to $current\_chain.synset$ and $current\_chain.related$ respectively (lines 8:9). Otherwise, it means there are no common related synsets between the current chain and S_i, so we understand that the current chain must be proper represented and added to the list of flexible chains (line 11). Thus, we find the synset with the highest cosine similarity against the average of all synset vectors in $current\_chain.synsets$. The average of all synsets in $current\_chain.synsets$ is calculated considering a pre-trained synset embeddings model (tsm) for the function $get\_best\_rep(current\_chain.synset, tsm)$. The synset embedding model (tsm) used is produced using the entire English Wikipedia Dump from 2010 [160] parsed into synsets through the MSSA algorithm in a word2vec implementation. After representing and including the current chain in the list of flexible chains we start to build a new chain, but now considering S_i (lines 12:13), the same method we use to start our algorithm.

After we iterate over all synsets in d, we also verify if there is any element in the current chain that was not added yet to our flexible chains list (line 14). This step mitigates the problem in which all synsets are combined in one single chain or the last synset S_n in the document is semantically related to the current chain being built. At the end of the FLLC II algorithm, we return a list of synsets, that represent all the lexical chains found in a document d (line 16).

Algorithm 3 Flexible Lexical Chain II Algorithm (FLLC II)

```
Require: $d = \{S_1, \dots, S_n\} : S_i \in ld$
Require: $tsm$ = trained synset embedding model
Require: $ld$ = lexical database
 1: function FLLC II($d$, $tsm$, $ld$)
 2:     flexible_chains_list = $\emptyset$
 3:     current_chain.$synsets$ = $[S_1]$
 4:     current_chain.$related$ = $\{$get_related_syns($S_1$, $ld$)$\}$
 5:     for $i$ = 2 to $n$ do
 6:         new_rel = $\{$get_related_syns($S_i$, $ld$)$\}$
 7:         if current_chain.$related$ $\cap$ new_rel not $\emptyset$ then
 8:             Add $S_i$ to current_chain.$synsets$
 9:             Add new_rel to current_chain.$related$
10:         else
11:             Add get_best_repr($current\_chain.synset$, $tsm$) to flexible_chains_list
12:             current_chain.$synsets$ = $[S_i]$
13:             current_chain.$related$ = $\{$get_related_syns($S_i$, $ld$)$\}$
14:     if current_chain.$synsets$ not $\emptyset$ then
15:         Add get_best_repr($current\_chain.synsets$, $tsm$) to flexible_chains_list
16:     return flexible_chains_list
```

The proposed algorithm for flexible chains improves its predecessor [148] in several aspects. First, we consider all POS when building our chains, instead of just nouns. Second, in addition to hypernyms, we also consider 18 additional attributes in WordNet for each synset evaluated (total of 19). This is also an improvement from most lexical chains systems, that often focus on hypernyms, hyponyms, and meronyms only. Third, our version of lexical chains uses the transfer knowledge from an external training task using word embeddings learned from the entire English Wikipedia.

3.3.2 Fixed Lexical Chains II (FXLC II)

The same way as in the FLLC II algorithm (Section 3.3.1), the FXLC II technique also builds its chains using a list of synsets. However, the FXLC II algorithm has a more general approach, as shown in Algorithm 4. The lexical chains in FXLC II are defined beforehand and do not require an explicit semantic relation between its synsets. In other words, we enforce the number of synsets for each chain (chunk) throughout the document. Each chunk can be considered to be an average of its constituent synsets, thus resulting in an abstracted document.

We extend Ruas and Grosky [149] algorithm for fixed chains to all POS and incorporate word embeddings containing the vector representation of the tokens in our document. In order to maintain consistency between our algorithms and experiments, we also use the MSSA algorithm (Section 3.2) to produce a list of synsets out of a given document d. Once we have a document d represented as synsets, we create a new document representation called $chunked\_document$, which is composed of chunks C_j of size cs (line 2).

Analogous to Algorithm 3, we also find the synset to represent each fixed lexical chain C_j in $chunked\_document$ (line 5). To find the synset with the highest cosine similarity against the average of all synset vectors in C_j we consider a pre-trained synset embeddings model (tsm) for the function $get\_best\_repr(C_j, tsm)$. We add each synset S_i in C_j, for the highest cosine similarity value, to our list of fixed chains, which is returned at the end of the FXLC II algorithm (line 6).

Algorithm 4 Fixed Lexical Chains II Algorithm (FXLC II)

```
Require: $d = \{S_1, \dots, S_n\} : S_i \in ld$
Require: $cs$ = chunk size for each lexical chain
Require: $tsm$ = trained synset embedding model
1: function FXLC II($d$, $cs$, $tsm$)
2:     chunked_document = split($d$, $cs$)
3:     fixed_chains_list = $\emptyset$
4:     for $j$ = 1 to length($chunked\_document$) do
5:         Add get_best_repr($C_j$, $tsm$) to fixed_chains_list
6:     return fixed_chains_list
```

The proposed technique for fixed chains also surpasses its predecessor [149] in some aspects. As in the FLLC II algorithm, we also consider all POS when building our chains, rather than just nouns. In addition, the FXLC II technique does not rely on traditional distance measures [177, 89, 181, 141, 77, 88, 93] to calculate how far our synsets are from each other to represent each chunk. Instead, based on the information provided by synset vectors, derived from Wikipedia, we find the closest semantic candidate in a chain with respect to all its inner elements.

Since the FXLC II approach ignores the semantic affinity between synsets and groups them for each chunk C_j, this approach can be extended to other document representations as well. Therefore, as long as the pair document-tokens and word embedding model have the same representation, FXLC II can be applied. We leave this investigation for future projects.

3.3.3 From Lexical Chains to Embeddings (Chains2Vec)

After our documents are transformed into lexical chains representations, which are also composed of synsets, we use them as input in a word2vec implementation. These documents integrate our external corpus that is used to produce a word embeddings model based on synsets of lexical chains.

The main idea concerning FLLC II and FXLC II is to obtain a better semantic representation for words from a large collection of documents that will hopefully generalize well enough to be applied to any NLP downstream task or real-world problem. In order to keep our vectors easy to interpret to other systems using synsets, we also represent each token in our corpus using the same format as the one in Section 3.2.4, when using the MSSA technique: $word\#synset\_offset\#pos$, where $word$ is the word itself, normalized in lowercase; $synset\_offset$ is an 8 digit, zero-filled decimal integer that corresponds to a unique word-sense, and pos is a part-of-speech tag (e.g. n for nouns, v for verbs, a for adjective, s for adjective satellite and r for adverb)[7]. Since we are using an equivalent notation, the synset embeddings models produced with FLLC II and FXLC II results can be incorporated in other systems with the same format as well.

For now, we currently explore the document classification task, but we believe that our chains can also be applied in other domains. Originally, the MSSA vectors were generated targeting the word similarity task, but we also use and extend them to the document classification task. In the following section, we provide a toy example of how FLLC II and FXLC II are used to produce our lexical chains.

3.3.4 Building Lexical Chains

Considering the algorithms introduced in Sections 3.3.1 and 3.3.2 we provide an illustrative example of how our lexical chains are built. Let us consider the following sentence:

Beets, carrots, and potatoes are grandma and grandpa's favorite dish for lunch!

As explained in Algorithms 3 and 4, we require a document composed of synsets to build our lexical chains. Before extracting the most suitable synset for each relevant token, we lowercase all the words, remove all punctuation and common English stopwords. After

[7]https://wordnet.princeton.edu/documentation/wndb5wn

pre-processing the sentence, we end up with the following list of tokens {*beets, carrots, potatoes, grandma, grandpa, favorite, dish, lunch*}. Next, we apply the MSSA algorithm (Section 3.2) to obtain the proper synset for each word in that list, as shown in Table 3.1. To make our toy example clearer, we reference the synsets for each word using the notation in row *Filtered Synset Sentence*. For each synset *Synset('word.pos.number')* we name it as *word.pos.number*, where *word* is the word itself, *pos* is the part-of-speech tag used, and *number* is the synset number in WordNet.

Table 3.1: Sentence example transformation from words into synset.

Type	Sentence
Original Sentence	Beets, carrots, and potatoes are grandma and grandpa 's favorite dish for lunch!
Pre-Processed Sentence	{beets, carrots, potatoes, grandma, grandpa, favorite, dish, lunch}
Synset Sentence	{Synset('beet.n.02'), Synset('carrot.n.03'), Synset('potato.n.01'), Synset('grandma.n.01'), Synset('grandfather.n.01'), Synset('favorite.n.01'), Synset('dish.n.02'), Synset('lunch.n.01')}
Filtered Synset Sentence	{beet.n.02, carrot.n.03, potato.n.01, grandma.n.01, grandfather.n.01, favorite.n.01, dish.n.02, lunch.n.01}

Once we have a specific synset for each token, we can start to build our chains. For the FLLC II algorithm, we extract all related synsets from each synset in the *Filtered Synset Sentence* (Table 3.1). The related synsets are other synsets extracted from 19 attributes in WordNet [1], such as hypernyms, homonyms, and meronyms to name a few. In Table 3.2, we illustrate some of the extracted synsets, in which a "-" represents the absence of a related synset.

In Table 3.2, it is possible to notice that some synsets have an overlap with respect to their related synsets. For example, the synsets {*beet.n.02, carrots.n.03, potatoes.n.01*} have *root_vegetable.n.01* as a common relation, which provides a semantic connection between them. The same characteristic is also present when considering *grandparent.n.01* for the synsets {*grandma.n.01, grandfather.n.01*}. For the cases where there are no common related synsets, a new chain must be built, as in the case of *favorite.n.01*, which has no

Table 3.2: Sample of related synsets extracted from WordNet.

Main Synsets	Related Synset 01	Related Synset 02	Related Synset 03
beet.n.02	root_vegetable.n.01	sugar_beet.n.02	beetroot.n.01
carrot.n.03	carrot.n.02	root_vegetable.n.01	-
potato.n.01	baked_potato.n.01	french_fries.n.01	root_vegetable.n.01
grandma.n.01	grandparent.n.01	nan.n.01	-
grandfather.n.01	grandparent.n.01	-	-
favorite.n.01	choice.n.01	-	-
dish.n.02	nutriment.n.01	meal.n.01	ingredient.n.03
lunch.n.01	meal.n.01	business_lunch.n.01	-

semantic relation with any of its adjacent neighbors. For the FXLC II algorithm, we take a different approach and group the synsets according to an arbitrary chunk size cs. In Table 3.3, we illustrate how the lexical chains are grouped for the FLLC II and FXLC II algorithms, considering $cs = 3$ for the latter. The chains are tagged in order and according to the algorithm used, where (A) represents the chains from FLLC II and (B) the chains from FXLC II. In case cs is greater than the number of available synsets in the document, that chain will be smaller.

Table 3.3: Lexical chains construction.

Lexical Chain Type	Lexical Chain
A - FLLC II	{A1:{beet.n.02, carrot.n.03, potato.n.01}, A2:{grandma.n.01, grandfather.n.01}, A3:{favorite.n.01}, A4:{dish.n.02, lunch.n.01}}
B - FXLC II	{B1:{beet.n.02, carrot.n.03, potato.n.01}, B2:{grandma.n.01, grandfather.n.01, favorite.n.01}, B3:{dish.n.02, lunch.n.01}}

Now that we have grouped the possible synsets to represent our chains, let us assume we have a pre-trained synset embeddings model for all the tokens in our corpus. For the sake of simplicity, our model has only five dimensions as Table 3.4 shows.

The next step is to represent each lexical chain considering the vectors of its inner elements. Two tasks are necessary to find the best representation for our chains: (i) calculate the average for each sub-chain, considering their vector values, in Table 3.3 and (ii) select the synset with the highest cosine similarity within each sub-chain with respect to their average calculated in (i). Table 3.5 shows the average vector calculated for each

Table 3.4: Fictional pre-trained synset embeddings model.

Synsets	Dim-01	Dim-02	Dim-03	Dim-04	Dim-05
beet.n.02	0.926	0.620	-0.278	-0.380	0.717
carrot.n.03	-0.525	0.361	0.839	-0.989	-0.420
potato.n.01	-0.675	-0.172	1.000	0.964	0.868
grandma.n.01	0.159	-0.795	0.663	0.405	0.617
grandfather.n.01	-0.035	0.469	-0.183	0.659	0.361
favorite.n.01	0.121	0.581	-0.603	0.148	0.046
dish.n.02	0.667	-0.257	-0.379	0.300	0.723
lunch.n.01	-0.874	0.458	0.945	0.717	0.314

sub-chain in Table 3.3.

Table 3.5: Vector average for lexical chains *centroids*.

Lexical Chain Type	Centroid	Dim-01	Dim-02	Dim-03	Dim-04	Dim-05
A - FLLC II	A1	-0.091	0.270	0.520	-0.135	0.388
	A2	0.062	-0.163	0.240	0.532	0.489
	A3	0.121	0.581	-0.603	0.148	0.046
	A4	-0.104	0.101	0.283	0.509	0.519
B - FXLC II	B1	-0.091	0.270	0.520	-0.135	0.388
	B2	0.082	0.085	-0.041	0.404	0.341
	B3	-0.104	0.101	0.283	0.509	0.519

Table 3.6 shows the cosine similarity between the elements of each chain and their respective average synset vector (centroid). Bold values represent the highest similarity value for that centroid with respect to a specific synset in the sub-chain. The closer to 1, the most similar a synset is to its centroid, and a better candidate to represent a chain it is. On the other hand, the closer to 0, the most dissimilar a synset is to its centroid, and the worst candidate it is. It is no surprise that if a chain is composed of one synset that synset will have the value of 1.0.

In Table 3.7 we show the resulting lexical chains for both FLLC II and FXLC II algorithms considering the highest cosine similarity values from Table 3.6. Even though this illustrates a simple example, we can see how the configuration of our chains can change depending on how their synsets are organized. In FLLC II the chain {*grandma.n.01, grandfather.n.01*} has *grandma.n.01* as a representative since it has the highest cosine sim-

Table 3.6: Cosine similarity between lexical chains elements and their *centroids*.

FLLC II			FXLC II		
| Synset	Centroid	CosineSim	Synset	Centroid	CosineSim
beet.n.02	A1	0.264	beet.n.02	B1	0.264
carrot.n.03	A1	0.509	carrot.n.03	B1	0.509
potato.n.01	A1	**0.578**	potato.n.01	B1	**0.578**
grandma.n.01	A2	**0.815**	grandma.n.01	B2	0.420
grandfather.n.01	A2	0.572	grandfather.n.01	B2	**0.882**
favorite.n.01	A3	**1.000**	favorite.n.01	B2	0.341
dish.n.02	A4	0.365	dish.n.02	B3	0.365
lunch.n.01	A4	**0.746**	lunch.n.01	B3	**0.746**

ilarity value. However, when considering FXLC II, the synset *favorite.n.01* gets merged to {*grandma.n.01, grandfather.n.01*} as a result of the selected chunk size. Thus, *grandfather.n.01* is chosen as a new synset representative for our chain based on its cosine similarity value.

Table 3.7: Final chains for FLLC II and FXLC II algorithms.

Lexical Chain Type	Lexical Chain
FLLC II	{potato.n.01, grandma.n.01, lunch.n.01}
FXLC II	{potato.n.01, grandfather.n.01, lunch.n.01}

Once the synset input corpus is transformed to either flexible or fixed lexical chains, we feed them into a word2vec implementation, as described in Section 3.3.3. As a result, we produce a synset (chains) embeddings model that can be used in any NLP downstream task. For the lexical chains algorithms, we explore the document classification task to validate our techniques, detailed in Section 4.3.

CHAPTER IV

Experiments and Validation Tasks

For each core contribution presented in Chapter III, specific NLP downstream tasks are selected to validate the proposed algorithms. The MSSA algorithms are applied in the word similarity task for six different datasets and compared against more than 15 word embeddings techniques. As for the lexical chains algorithms, the document classification task is explored considering six state-of-the-art approaches and six benchmark datasets. For the variations of MSSA with the best results in the word similarity task, we also include them in the document classification experiments, thus, providing a more detailed scenario to compare the proposed techniques under the same circumstances.

4.1 Word Similarity Task

We design a series of experiments for the word similarity task to evaluate how our algorithms compare against other approaches in the literature. In the next sections, we present and discuss the main characteristics of the training corpus, benchmarks, and compared systems.

4.1.1 Training Corpus

We apply the MSSA algorithms to two datasets as a training process, transforming their words into synsets using the English WordNet 3.0 [48], also known as Princeton WordNet, as our lexical database. The datasets are Wikipedia Dumps consisting of wiki articles from

April 2010 (WD10) [160] and January 2018 (WD18). Table 4.1 shows the details for both training corpora after they are cleaned (Section 3.2.1).

Table 4.1: Dataset token details. WD10 - English Wikipedia Dump 2010 (April); WD18 - English Wikipedia Dump 2018 (January).

POS	**Words** (10^6)		**Synsets**	
	WD10	**WD18**	**WD10**	**WD18**
Nouns	299.41	463.31	55731	56546
Verbs	130.14	161.96	11975	12237
Adverbs	27.25	31.17	3091	3056
Adjectives	75.77	104.03	15512	15798
Total	532.57	760.47	86309	87637

4.1.2 Hyperparameters, Setup and Details

Once all words in the training corpora are processed into synsets, we use a word2vec implementation to produce our synset embeddings. The hyperparameters for the word embeddings training are set as follows: CBOW for the training algorithm, window size of 15, minimum word count of 10, hierarchical softmax, and vectors size of 300 and 1000 dimensions. If not specified, all the other hyperparameters are used with their default values[1]. Our system is implemented in Python 3.6, with NLTK 3.2.5 and using the *gensim* 3.4.0 [137] library.

In our experiments, we evaluate our approach with several systems, described in Sections 4.1.4 and 4.1.5, using two different training corpora (WD10 and WD18) for the word similarity task. In a second-level analysis, we also explore the properties of our models separately, over different perspectives. For WD10, we discuss the effects of the number of iterations on our recurrent model MSSA-NR, with $-N$ ranging from 0 to 2, where $N = 0$ characterizes the initial scenario using Google News vectors (MSSA) and $N \geq 1$ the iterative one (synset vectors), as illustrated in Figure 3.2. For WD18, we investigate which

[1]https://radimrehurek.com/gensim/models/word2vec.html

of our representations of word-senses performs better, the one considering a local context (MSSA) or the global one (MSSA-D). The comparison of our recurrent model (MSSA-NR) against the MSSA-D algorithm is not explored in the proposed experiments, but we plan to include it in future work. However, to analyze how our synset embeddings are affected by the timestamp difference in the Wikipedia snapshots, we do compare the results of MSSA for both training corpora, WD10 and WD18. The standard number of dimensions used in our experiment is 300, with no specific label for MSSA, and 1000, which is indicated with $-T$ next to the algorithm's name.

The differences between metric names, benchmarks, datasets and hyperparameters make it difficult to perform a direct comparison between all available systems. We try to alleviate this situation by explaining the reason behind our choices for the components in our architecture. In the disambiguation step, we use WordNet [48] as our lexical database, due to its robustness and popularity for this task. Princeton WordNet (or English WordNet) is the most used resource for WSD in English and it is available in more than 70 different idioms [120]. WordNet is also free of charge (for any purpose), can be accessed without any restriction, and is fully integrated with NLTK in Python, making its use preferable over other lexicographic resources, at least for now.

As for the word training embeddings step, we chose word2vec [110] over other popular techniques, such as GloVe [130], fastText [17], and ELMo [131] because of word2vec's resource-friendly implementation, popularity, and robustness in several NLP tasks [74, 75, 91, 99, 133, 145]. In addition, GloVe embeddings are based on the co-occurrence probabilities of the words in a document encoded in a word-context co-occurrence matrix (counting), while word2vec's embeddings are built using prediction models (CBOW or skip-gram), which are closer to our objective. While GloVe requires the entire matrix to be loaded into memory, making its consumption of memory RAM quadratic in its input size, word2vec works with linear memory usage, facilitating the training part of our system. As for fastText, its approach uses word substrings (n-grams) to produce embeddings, in addition to complete words, as in word2vec. The results comparing word2vec and fastText (without n-grams) models are almost equivalent, but some claim that fastText exhibits better performance in syntactic tasks, in comparison with word2vec, which is more adequate for semantic representations [76]. Since our model is focused on the semantic aspects of each word-sense and WordNet is not able to provide valid synsets for many of the produced n-grams (e.g. kiwi - kiw, iwi), word2vec is a natural choice. In ELMo, they compute their word vectors as the average of their characters representations, obtained through a two-layer biLM. This would bring even more granularity to the sub-word embeddings proposed in fastText, as they consider each character in a word has their own n-dimension vector representation. Another factor that prevents us from using ELMo, for now, is its expensive training process[2]. We also consider the recently published USE [29] from Google, but their implementation does not allow it to be trained in a new corpus such as ours (synset-based), only to use their pre-calculated vectors.

For the training corpora, WD10 [160] is commonly used in many systems [31, 73, 74, 91, 97, 122] and WD18 is introduced as a variation in our experiments to analyze the behavior of our own approaches.

Recent publications have pointed out some problems (e.g. model overfitting, subjectivity) in using word similarity tasks to evaluate word embeddings models [14, 47]. We try to mitigate this situation by carefully illustrating some aspects, such as the general idea of the proposed architecture, a detailed description of the components used in the system, the training corpus specification, hyperparameters' definitions, and comparison of our approaches in different training scenarios. We also apply our models to the most popular benchmarks available, without changing their original structure, and categorize all referenced results according to the correct metrics (AvgSim, AvgSimC, MaxSim, MaxSimC/LocalSim, and GlobalSim) defined in seminal publications [73, 140]. Unfortunately, many authors do not describe either what exact metric they use in their experiments, nor specify which one is

[2]https://github.com/allenai/bilm-tf

used in their referenced results. It is common to notice systems being compared under different scopes; this and other issues make our evaluation harder for some specific systems. We try to alleviate such situations, providing as many details as possible for the experiments, metrics, and artifacts used.

The results presented in Sections 4.1.4 and 4.1.5 are organized in three blocks for each benchmark (Tables 4.2:4.7), divided by a double break line and ordered as follows:

1. **Single-sense embeddings**: traditional word embeddings where all word-senses are collapsed into one vector representation per word. Approaches that concatenate a word vector with their senses are also included;

2. **Multi-sense embeddings**: each word-sense has a specific vector representation. Approaches that have a vector for both the word and their senses separately are also included;

3. **MSSA embeddings**: all our proposed models, for multi-sense embeddings.

With the exception of MSSG [122], Chen et al. [32], and Convolutional Neural Network Multi-Sense Skip-Gram (CNN-MSSG) [31], which are trained using the skip-gram model, all the compared systems either use CBOW or an independent approach of word embeddings (e.g. GloVe). Results not reported in the referenced authors are marked as "-" for the given metrics.

All proposed algorithms (MSSA, MSSA-D, MSSA-NR) and generated models used in this paper are available in a public repository[3].

4.1.3 Benchmark Details for Word Similarity Task

The experiments are divided into two major categories, based on the datasets' characteristics: No Context Word Similarity (NCWS) and Context Word Similarity (CWS). All datasets are widely used in the word similarity task for the compared systems. The former

[3]https://github.com/truas/MSSA

(1 to 5) groups' benchmarks that provide similarity scores for word pairs in isolation, while the latter (6) provides a collection of word pairs with their similarity scores accompanied with sentence examples of their use. These sentences are used to illustrate a context where each word compared is applied. The benchmarks used are described as follows:

1. **RG65**: 65 noun pairs. The similarity scale ranges from 0 to 4 [152];

2. **MC28**: 28 pairs of nouns that were chosen to cover high, intermediate, and low levels of similarity in RG65. This is the same set of words in MC30 [113], except for two words not present in WordNet. The similarity scale ranges from 0 to 4 [141];

3. **WordSim353**: 353 noun pairs divided into two sets of English word pairs, the first set with 153 word pairs and the second with 200 [51]. The original dataset is later re-organized [3], claiming that this dataset does not make any distinction between similarity and relatedness. We use the original version published [51]. The similarity scale ranges from 0 to 10;

4. **MEN**: 3,000 word pairs, randomly selected from words that occur at least 700 times in the ukWaC and Wacky corpora[4] combined, and at least 50 times in the *ESP Game*. The similarity scale ranges from 0 to 50 [23];

5. **SimLex999**: 666 noun-noun pairs, 222 verb-verb pairs, and 111 adjective-adjective pairs. the similarity scale ranges from 0 to 10 [69]; and

6. **SCWS - Stanford Context Word Similarity**: 2,003 word pairs and their sentential contexts, consisting of 1328 noun-noun pairs, 399 verb-verb pairs, 140 verb-noun, 97 adjective-adjective, 30 noun-adjective, 9 verb-adjective, and 241 same-word pairs. The similarity scale ranges from 0 to 10 [73].

We try to keep our basic configuration as close as possible to recent previous publications, so we consider the cosine similarity as our distance measure and report the Spearman

[4]http://wacky.sslmit.unibo.it/doku.php?id=corpora

correlation value (ρ) in our experiments. To guarantee a common scenario between all benchmarks, we normalized their similarity scale to an interval of [-1, 1]. Very few publications report results for both Spearman and Pearson correlation values, so we adopt only the former to minimize the differences between our comparisons. Thus, more systems can be included in the word similarity task experiments. The results reported in our experiments, for all model variations, have high significant Spearman order correlation, with a $p-value$ under 0.001, another characteristic that most publications often do not mention.

4.1.4 No Context Word Similarity

In this section, we evaluate our model against popular approaches available for 5 benchmarks datasets: RG65, MEN, WordSim353, SimLex999 and MC28. We compare our results with: Chen et al. [32], Retro (using Glove with 6 billion words and WordNet with all synsets) [46], Huang et al. [73], SensEmbed [74] (400 dimensions), SW2V (variations using BabelNet and WordNet with UMBC and Wikipedia Dump from 2014) [99], word2vec (using UMBC and WD14) [99], word2vec [110], MSSG and NP-MSSG (for 50 and 300 dimensions) [122], GloVe (using 6 and 42 billion words) [130], Pruned-TF-IDF [139], and DeConf [133]. If not specified, the compared systems use low-dimensional vectors with 300 dimensions each. All of them also use cosine similarity to calculate the distance of words in each benchmark, except for SensEmbed, which uses the *Tanimoto* distance [169] for their vector comparison. The Tanimoto coefficient is commonly used for binary attributes in vectors, while cosine similarity is applied mainly to non-binary vectors, when their magnitudes are not relevant. In addition, SensEmbed also introduces what they call a *graph vicinity factor*, an argument created to adjust the final similarity score based on the information only present in BabelNet. Even though NASARI [27] achieves impressive results, its comparison with most systems is compromised since they use ad-hoc variations of the traditional benchmarks in their original report. When considering the SimLex999 benchmark, only the noun-noun pairs are evaluated, discarding the other POS (verb-verb and adjective-adjective). Alternative versions for the MC28 and WordSim353 benchmarks are also used, even though the original versions are more common in the literature. For MC28, they consider its earlier version MC30 [113], while for WordSim353 they consider the similarity dataset described in [3]. These and other minor aspects would make the comparison against other systems more restrictive and unrealistic. Thus, we leave their results out of our experiments.

Tables 4.2 and 4.3 show the results of MSSA against several models for the RG65 and MEN benchmarks, respectively. In both experiments, SensEmbed and DeConf-Sense present the highest results for the AvgSim and MaxSim metrics, followed by one of our models. SensEmbed builds its vectors using BabelNet as its disambiguation backbone, through Babelfy. BabelNet is composed of several different resources[5], including specific lexicons (e.g. GeoNames, Wikiquote, Microsoft Terminology). After the disambiguation step, they train a word2vec model with 400 dimensions. As mentioned above, they also introduce what is called a *graph vicinity factor*, a coefficient that combines the structural knowledge from BabelNet's semantic network and the distributional representation of sense embeddings. This factor multiplies AvgSim (Equation 2.4) and GloSim (Equation 2.8) scores to re-adjust the similarity measure [74]. DeConf-Sense, as in our models, relies on less resources to produce its word-sense embeddings. Their approach uses traditional single-sense embeddings (e.g. Google News Vectors) and divides them into separate word-sense vectors according to WordNet's semantic network. Also, they use the Personalized PageRank [66] to calculate the semantic relatedness between two synsets in their core [133]. All multi-sense embeddings systems surpass single-sense ones, in which, for the GloSim metric, MSSA-2R-T and MSSA (WD10) have the highest results for RG65 and MEN datasets.

Even though our models do not perform as well as DeConf-Sense for MaxSim, our approach is able to be trained recurrently, improving the quality of its vectors. In RG65,

[5]https://babelnet.org/about

we start with ρ = 0.857 for MSSA (WD10) and move to ρ = 0.872 with MSSA-1R, for the MaxSim as shown in Table 4.2. We try to increase the number of iterations, but unfortunately MSSA-2R does not produce better vectors as one would expect. Further investigation is necessary to evaluate if MSSA-2R reached its limit or is stuck in a local maximum. Since the disambiguation step is costly for us, at this point, we leave the exploration of more iterations for future experiments. If this process is performed in parallel, we can increase and investigate higher values for N (e.g. 5, 10, 100). The comparison with SensEmbed is compromised since their model has many differences with the others, including the metric used. However, even using a simpler lexical database (WordNet) our models obtained competitive ρ values for MSSA-1R, MSSA-T, and MSSA-2R-T.

The increase of dimensionality seems to have a positive effect in most word embeddings models, including ours. WD10 and WD18 models present improvements when each of their models is compared with its 1000-dimensional version, as Tables 4.2 (RG65) and 4.3 (MEN) show. For WD10, the increase, is on average, 1.61% for both benchmarks in total, while for WD18 it is 0.48% for RG65 and 0.28% for MEN. Looking only at MSSA, it is hard to affirm that more words would necessarily represent a better result, for if that were true, Glove-42B and Retro-G6B in Table 4.3 should have more competitive scores, since they are trained over 42 and 6 billion words, respectively. The performance of WD18 for MSSA and MSSA-D is not clear for the global and local contextual aspects, since their results do not improve consistently for all metrics. However, MSSA does obtain better scores for RG65 and MEN. We do fine-tune the hyperparameters of these models in non-reported results, but this just reinforces the findings of [14, 47] with respect to model overfitting for specific benchmarks. For this reason, we keep the same configuration between our models among all experiments so we can evaluate how well they generalize.

The results reported for the SW2V algorithm [99] in Table 4.3 (MEN), show an interesting behavior with respect to the lexical database used. Their ρ varies no more than 0.01 when we compare the models using the same corpus (UMBC or WD14), which indicates

Table 4.2: Spearman correlation score (ρ) for the RG65 benchmark. Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
GloVe-42B	-	-	0.829
GloVe-6B	-	-	0.778
Retro-G6B	-	-	0.767
Retro-G6B-WN	-	-	0.842
word2vec	-	-	0.754
DeConf-Sense	-	**0.896**	-
DeConf-Word	-	0.761	-
SensEmbed	**0.871**	0.894	-
SW2V-Shallow	-	0.740	-
SW2V-Babelfy	-	0.700	-
MSSA(WD10)	0.779	0.857	0.830
MSSA-1R(WD10)	0.795	0.872	0.825
MSSA-2R(WD10)	0.814	0.869	0.858
MSSA-T(WD10)	0.783	0.878	0.845
MSSA-1R-T(WD10)	0.825	0.871	0.856
MSSA-2R-T(WD10)	0.822	0.878	**0.859**
MSSA(WD18)	0.828	0.794	0.821
MSSA-D(WD18)	0.801	0.826	0.817
MSSA-T(WD18)	0.776	0.847	0.816
MSSA-D-T(WD18)	0.795	0.839	0.835

that the BabelNet (-BN) variation is as robust as WordNet (-WN) to capture the semantic relationships in this dataset. This finding is also present in our results as well, as our ρ score fluctuates around the same range with a slightly superior performance for MSSA (WD18), with ρ = 0.769.

In Table 4.4 (WordSim353), all results perform worse than the single-sense embeddings of GloVe [130] for the GloSim metric. However, to reach this score they process 42 billion tokens, while, when considering just 6 billion tokens, its performance decreases 13.30%. We, on the other hand, with a little less than 540 million tokens for WD10, can obtain superior results with MSSA-2R and MSSA-2R-T. Even though Pruned-TF-IDF [139] follows with a competitive ρ score, their model does not use low-dimensional vectors,

Table 4.3: Spearman correlation score (ρ) for the MEN benchmark. The results of Chen et al. [32] and word2vec are reported in Mancini et al. [99] (MSSG/NP-MSSG). Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
Retro-G6B	-	-	0.737
Retro-G6B-WN-All	-	-	0.759
word2vec(UMBC)	-	0.750	-
word2vec(WD14)	-	0.720	-
Chen et al.(2014)	-	0.620	-
DeConf-Sense	-	**0.786**	-
DeConf-Word	-	0.732	-
SensEmbed	**0.805**	0.779	-
SW2V-BN-UMBC	-	0.750	-
SW2V-WN-UMBC	-	0.760	-
SW2V-BN-WD14	-	0.730	-
SW2V-WN-WD14	-	0.720	-
MSSA(WD10)	0.751	0.745	0.760
MSSA-1R(WD10)	0.781	0.751	0.790
MSSA-2R(WD10)	0.777	0.737	0.788
MSSA-T(WD10)	0.778	0.753	0.785
MSSA-1R-T(WD10)	0.783	0.747	0.791
MSSA-2R-T(WD10)	0.785	0.744	**0.795**
MSSA(WD18)	0.745	0.769	0.775
MSSA-D(WD18)	0.768	0.716	0.765
MSSA-T(WD18)	0.769	0.749	0.776
MSSA-D-T(WD18)	0.772	0.717	0.767

which makes its direct comparison problematic. In addition, their model relies on several parameter adjustments (e.g. pruning cutoff, feature weighting, number of prototypes, feature representation). In contrast, our model works independently of any parameters other than those required in a word2vec implementation.

For the MaxSim metric we observe that our initial model MSSA(WD18) obtains equal results when compared to SensEmbed, and better results when considering SW2V in their two forms, $-Shallow$ and $-Babelfy$. Because of our model's simplicity we highlight MSSA(WD18) instead of SensEmbed for this metric. SW2V and SensEmbed models use

BabelNet in their disambiguation process, while ours only uses WordNet, which BabelNet incorporates completely. In addition, most of our models for WD10 and WD18 also present superior scores against SW2V-Babelfy.

Under more similar characteristics, MSSG and NP-MSSG models of [122] present closer results to our systems for both AvgSim and GloSim, as Table 4.4 shows. They also produce multi-sense embeddings, based on word-senses that are learnt jointly with the word vector itself. MSSG and NP-MSSG only differ in the number of senses a word can have, which is similar to what we accomplish with MSSA. Their training time of 6 hours for MSSG-300d and 5 hours for NP-MSSG are comparable with our synset embeddings (Section 3.2.4) step. However, unlike MSSA, which requires a disambiguation process prior to the embeddings one, their model does these tasks at the same time and with strong competitive results. For GloSim, MSSG-300d and NP-MSSG-300d present ρ of 0.692 and 0.691 respectively, while MSSA-T (WD18) and MSSA-D-T has values of 0.692 and 0.693, respectively. This shows that, as for the amount of words processed, the number of dimensions does not necessarily provide better results. In general, our models using 300 dimensions obtain better performance than those with a higher dimensionality. The same behavior is observed when we consider the Spearman values using AvgSim for MSSG-300d, MSSA-1R-T, and MSSA-D-T. Wihtin the same dimensionality, the results decrease when we move from the base to the first iteration in MSSA-NR (i.e. from $N = 0$ to $N = 1$), but they improve for the second iteration ($N = 2$). Now, considering the same iteration pass, for different dimensions (300 and 1000d), we perform slightly worse, as Table 4.4 shows.

As explained in Section 4.1.3, the WordSim353 benchmark is composed of two separate datasets, the first with 153 word pairs and the second with 200. According to Agirre et al. [3], this benchmark conflates two distinct aspects of linguistics: similarity and relatedness. Thus, some authors [74, 99] take this into account and use the updated version of this dataset. In Table 4 of [74], they report their results for the WordSim353 dataset without any distinction or reference to the structure split in [3], so we assume they are making use of the original version proposed in [51]. Our expectation for the global model (MSSA-D) context superiority is not achieved for the WordSim353 dataset. If we only analyze the performance of both MSSA (WD18) and MSSA-D, we observe the inconsistency of the results, with the former approach showing better results than the latter, for MaxSim and GloSim. It is in our plans to investigate which subcategory can be better explored in our models for this benchmark, but for now, we keep WordSim353 as one single set, so more systems can be compared.

Table 4.4: Spearman correlation score (ρ) for the WordSim353 benchmark. Huang et al. [73] results are reported in Neelakantan et al. [122] (MSSG/NP-MSSG). Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
GloVe-42B	-	-	**0.759**
GloVe-6B	-	-	0.658
Retro-G6B	-	-	0.605
Retro-G6B-WN-All	-	-	0.612
Huang et al. (2012)	0.642		0.228
MSSG-50d	0.642	-	0.606
MSSG-300d	0.709	-	0.692
NP-MSSG-50d	0.624	-	0.615
NP-MSSG-300d	0.686	-	0.691
Pruned-TF-IDF	-	-	0.734
SensEmbed	**0.779**	0.714	-
SW2V-Shallow	-	0.710	-
SW2V-Babelfy	-	0.630	-
MSSA(WD10)	0.725	0.702	0.727
MSSA-1R(WD10)	0.711	0.661	0.712
MSSA-2R(WD10)	0.730	0.662	**0.737**
MSSA-T(WD10)	0.712	0.669	0.721
MSSA-1R-T(WD10)	0.708	0.666	0.716
MSSA-2R-T(WD10)	0.729	0.667	**0.737**
MSSA(WD18)	0.663	**0.714**	0.712
MSSA-D(WD18)	0.708	0.626	0.702
MSSA-T(WD18)	0.694	0.637	0.692
MSSA-D-T(WD18)	0.702	0.623	0.693

The last two NCWS benchmarks, SimLex999 and MC28, are particularly challenging, for distinct reasons. For SimLex999, in Table 4.5, we identify a consistent improvement with respect to the increase in dimensionality between models of the same configuration using WD10, but not for WD18. Results using the recurrent models (MSSA-NR) and their 1000-dimensional versions also present consistent improvement for WD10. However, on average our models perform poorly regardless of their configuration for all metrics, while DeConf-Sense holds the best results for MaxSim. The average Spearman correlation values for this dataset seems to be low in all publications, rarely surpassing $\rho = 0.50$. Even in our unreported models, we do not have satisfactory results. The same behavior is observed when we try to apply our model in no-nouns dominant benchmarks, such as: YP130 [183] and SimVerb3500 [57]. For the former, our Spearman scores achieve on average $\rho = 0.563$ (Avgsim), while for the latter, $\rho = 0.243$ (MaxSim). Our suspicion is that our models, as with most compared systems, are not robust enough to deal with datasets of this nature (i.e. mainly, not nouns). It seems verbs, adjectives, and adverbs-based benchmarks need a more specific approach to deal with their characteristics properly. As in Table 4.3 (MEN), the results of the SW2V algorithm [99] for the MaxSim metric presents little or no variation in their ρ score, when considering the same training corpus under different lexical databases, as Table 4.5 (SimLex999) shows. Likewise, our MSSA trained in the WD18 corpus obtains the best results among our models.

For MC28, reported in Table 4.6, the lack of recent publications makes it hard to draw any strong conclusions about compared models. We do obtain the same results as SensEmbed, but with a much simpler architecture and less resources, since all of our algorithms only use WordNet as its lexical resource. If we examine the Association for Computational Linguistics (ACL) state-of-the-art Wiki[6] we obtain the third best result considering the human upper bound as the gold standard. Since MC28 is a subset of RG65, our models present similar results, but with slightly better values on average, for both the WD10 and WD18 training corpora.

[6]https://aclweb.org/aclwiki/MC-28_Test_Collection_(State_of_the_art)

Table 4.5: Spearman correlation score (ρ) for the SimLex999 benchmark. Chen et al. [32] and word2wec results are reported in Mancini et al. [99] (MSSG/NP-MSSG). Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
word2vec(UMBC)	-	0.390	-
word2vec(WD14)	-	0.380	-
Chen et al.(2014)	-	0.430	-
DeConf-Sense	-	**0.517**	-
DeConf-Word	-	0.443	-
SW2V-BN-UMBC	-	0.470	-
SW2V-WN-UMBC	-	0.450	-
SW2V-BN-WD14	-	0.430	-
SW2V-WN-WD14	-	0.430	-
MSSA(WD10)	0.427	0.368	0.396
MSSA-1R(WD10)	0.438	0.369	0.405
MSSA-2R(WD10)	0.440	0.369	0.408
MSSA-T(WD10)	0.456	0.393	0.432
MSSA-1R-T(WD10)	0.468	0.394	**0.441**
MSSA-2R-T(WD10)	**0.469**	0.385	0.439
MSSA(WD18)	0.375	0.438	0.404
MSSA-D(WD18)	0.401	0.351	0.374
MSSA-T(WD18)	0.460	0.389	0.430
MSSA-D-T(WD18)	0.425	0.372	0.391

In all benchmarks, except SimLex999, our proposed models give competitive results within the top three positions. Considering the MaxSim and GloSim metrics, we report either the first or second highest Spearman correlation values in all experiments. Our models are strongly based on the context surrounding the word we disambiguate, so we believe that the more information about a word's surroundings we have, the more accurate our representations will be. Nonetheless, our models perform very well when applied to benchmarks considering the similarity of word pairs without any extra context information. In addition, we are able to obtain better results than several systems based on more complex architectures and lexical databases. MSSA's single training phase minimizes the hyperparameter adjustments to the word2vec implementation only, which makes it easier to replicate. Moreover,

Table 4.6: Spearman correlation score for the MC28 benchmark. Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
GloVe-42B	-	-	0.836
GloVe-6B	-	-	0.727
SenseEmbed	-	0.880	-
MSSA(WD10)	0.833	0.862	0.842
MSSA-1R(WD10)	0.825	0.883	0.843
MSSA-2R(WD10)	0.829	0.849	0.847
MSSA-T(WD10)	**0.845**	**0.888**	**0.875**
MSSA-1R-T(WD10)	0.841	0.883	0.862
MSSA-2R-T(WD10)	0.801	0.866	0.836
MSSA(WD18)	0.775	0.799	0.792
MSSA-D(WD18)	0.835	0.807	0.829
MSSA-T(WD18)	0.796	0.834	0.818
MSSA-D-T(WD18)	0.801	0.833	0.821

our modular configuration displays an appealing layout for us to apply the systems that obtain the best results throughout the experiments for the word similarity task. Considering the NCWS category, MSSA shows superior results over MSSA-D, making us suspect that a local context sliding window seems to be more adequate to extract semantic features of a corpus to build their embeddings. As in other compared systems, we also observe that the increase of dimensionality is most likely to improve the overall performance for the word similarity task, but this is not always true. The increase of dimensionality usually adds more computational time to the word embeddings step, which might not be worth the effort, given the small differences when compared to lower-dimensional vector models.

4.1.5 Context Word Similarity

In this section the results of AutoExtend (Synsets) [145] and CNN-VMSSG [31] are also incorporated. For the Stanford Context Word Similarity (SCWS) benchmark, we do not report the results for the MaxSim metric, since almost all publications do not explore them as well. As explained in Section 4.1.3, the SCWS dataset provides word pairs, their similarity score, and a given specific context for each word in a sentence, alleviating dubious interpretations [73].

Table 4.7 shows DeConf [133], NP-MSSG-300d [122] and MSSA (WD10) with the highest Spearman correlation values, in descending order for AvgSim. However, when considering MaxSimC and GloSim, the results of MSSA-1R and MSSA(WD10) give state-of-the-art scores, respectively. It seems that the extra information about a word's context indeed helps our model to better select a word-sense, but using all the remaining metrics for the word similarity task does not produce good results. CNN-VMSSG presents the second highest results for MaxSimC and GloSim after our models, but their approach relies on two training steps, one for the CNN part and one for the MSSG, instead of just one training step, like ours. Since in our disambiguation step MSSA looks for a word-sense with the highest similarity of its word (w_i) against its neighbors (w_{i-1} and w_{i+1}), the score obtained in a metric that reflects the maximum similarity of a word-sense, given its sentence context, is expected. This encourages us to apply our models to tasks in which sentences are compared, instead of just words. Thus we extend our work for document classification problems, where each entity (document) has a collection of tokens to build their embeddings.

The increase in the word-vector dimensionality has the same behavior as in previous experiments for the NCWS datasets, with the exception of MSSA (WD18) which has an increase of almost 8% on its score for AvgSim, if compared with MSSA-D. The recurrent model (MSSA-NR), considering the WD10 corpus, and the global context approach (MSSA-D) for WD18, seem to have little in the overall score when compared with their initial models (MSSA).

For AvgSimC, our models do not present competitive results, while DeConf and MSSG-300d are able to produce top scores. Since our approach is oriented towards the higher similarity between word-senses and their context, perhaps a different scheme to select prospective word-senses could improve our system. It would also be interesting to apply the top-ranked algorithms to our model and compare their performance for all metrics. Most of the published results do not report their findings for all metrics in [73, 140], making their direct comparison arduous. DeConf is designed to use pre-trained single dimensional vectors to produce their multi-sense embeddings, so our approach is not easily applicable, as we produce a vector for each word-sense directly. NP-MSSG [122] and SW2V [99] on the other hand, offer the necessary flexibility to use our annotated corpus to produce new embeddings.

Table 4.7: Spearman correlation score (ρ) for the SCWS benchmark. Huang et al. [73] results are reported in Neelakantan et al. [122] (MSSG/NP-MSSG). Highest results reported in **bold** face.

Models	**Avg Sim**	**Avg SimC**	**Max SimC**	**Glo Sim**
GloVe-42B	-	-	-	0.596
GloVe-6B	-	-	-	0.539
AutoExtend	0.626	0.637	-	-
Chen et al. (2014)	0.662	0.689	-	0.642
CNN-VMSSG	0.657	0.664	0.611	0.663
DeConf-Sense	**0.708**	**0.715**	-	-
Huang et al. (2012)	0.628	0.657	0.261	0.586
MSSG-50d	0.642	0.669	0.492	0.621
MSSG-300d	0.672	0.693	0.573	0.653
NP-MSSG-50d	0.640	0.661	0.503	0.623
NP-MSSG-300d	0.673	0.691	0.598	0.655
Pruned-TF-IDF	0.604	0.605	-	0.625
SensEmbed	-	0.624	0.589	-
MSSA(WD10)	0.667	0.581	0.637	**0.667**
MSSA-1R(WD10)	0.660	0.581	**0.639**	0.659
MSSA-2R(WD10)	0.665	0.593	0.631	0.665
MSSA-T(WD10)	0.659	0.590	0.617	0.664
MSSA-1R-T(WD10)	0.655	0.594	0.623	0.658
MSSA-2R-T(WD10)	0.661	0.604	0.617	0.664
MSSA(WD18)	0.593	0.569	**0.639**	0.651
MSSA-D(WD18)	0.640	0.557	0.613	0.640
MSSA-T(WD18)	0.649	0.588	0.617	0.654
MSSA-D-T(WD18)	0.638	0.570	0.597	0.639

4.2 Further Discussions and Limitations on MSSA

In this section, we try to provide a deeper discussion about the main aspects of our techniques, while pointing out their strengths and limitations. We also present alternatives that can be taken into account to mitigate these limitations.

The main objective of our algorithms is to properly transform word-based documents into synset-based ones that can be used in systems or tasks dealing with semantic representation at some level. For this, we use WordNet to identify possible word-senses of a given word. Unfortunately, this forces us only to work with formal texts (i.e. free of colloquial English, slang and typos). Traditional word embeddings techniques are derived directly from the raw text, which can be either an advantage or disadvantage, depending on which task is selected for its validation. If we move to a document classification task for documents based on informal text collections (e.g. user comments in a blog, movies reviews), our approaches would probably work poorly. On the other hand, we believe if the documents considered make use of formal English (e.g. scientific paper abstracts, news articles) our techniques might be adequate.

Another aspect is WordNet structure itself, with respect to the number of its available synsets and its idiom-version. Currently, in version 3.0., WordNet has 155,287 unique strings mapped, with 117,659 synsets, leaving a reasonable amount of words out, if we consider the entire English vocabulary. An alternative lexical database to be considered is BabelNet [121], which is composed of several different resources[7] (including WordNet) and specific lexicons (e.g. GeoNames[8], Wikiquote[9], Microsoft Terminology[10]). Since we are using the English version of WordNet (also known as Princeton WordNet) our system currently does not apply to other languages in its current version. The language aspect is not a barrier to BabelNet since it is a multilingual lexical database. However, it is

[7]https://babelnet.org/about

[8]http://verbs.colorado.edu/ mpalmer/projects/verbnet.html

[9]https://www.wikiquote.org

[10]http://www.microsoft.com/Language/en-US/Terminology.aspx

important to mention that there are other idiom versions[11] of WordNet available [1] that can be incorporated in our techniques.

From the experiments presented in Sections 4.1.4 and 4.1.5, we notice that our technique is very sensitive to the presence of verbs and adjectives POS, as Table 4.5 shows. Even though, we do not explicitly report the results obtained using YP130 [183] and SimVerb3500 [57] datasets, our overall performance for these are not satisfactory. However, when using datasets that provide context to the words being evaluated (SCWS), our techniques show competitive results (Table 4.7). This encourages us to explore tasks in which we can use larger context to support our decision, such as document classification, sentiment analysis, and plagiarism detection.

Our modular and flexible architecture (Figure 3.2), provides an interesting setup that can be applied to any expert system using natural text as its input. This is because the disambiguation and annotation steps work as a pre-processing phase, and can be applied to the raw text directly, in order to obtain a more precise semantic representation for a given word. Nevertheless, since MSSA considers all word-senses available in WordNet, it might not be the best option for very large training sets. In our experiments, it takes us approximately four full days to transform all documents in the English Wikipedia Dump of 2010 [160] in their synset versions. However, if available, one can parallelize the processing of the words into synsets since our MSSA, MSSA-NR, and MSSA-D are performed on a document level, individually. Moreover, the word to synset transformation task only needs to be performed once. After the annotated synset training corpus is derived, one can use it in any activity (e.g. NLP tasks, word embeddings training models). In addition, since words that do not exist in WordNet are automatically discarded, this reduces the time of any word embeddings technique applied to the translated training corpus. As with the experiments in Section 4.1, this can lead to good or bad results, depending on the datasets used for validation.

[11]http://globalwordnet.org/wordnets-in-the-world/

Differing from MSSA, which evaluates all senses for each word and requires the calculation of gloss vectors, the MSSA-NR algorithm only considers the word-senses that are actually embedded, in the first place. Therefore, the non-used word-senses are dropped, reducing the amount of comparisons required for each context window. In this case, the average complexity for MSSA-NR is smaller than the one in MSSA. Approaches like MSSG [122] fix the number of possible word-senses available and obtain a faster disambiguation process than MSSA. The gain in speed however, comes with a price of removing word-senses that could provide a better semantic representation for words.

During our experiments, we can observe that MSSA-D is highly affected by the number of words in a document, probably because of its global search. Dijkstra's algorithm does perform a blind search for all the available paths in the graph, that goes from the word-senses of the first word to the last. MSSA and MSSA-NR, on the other hand, have a local search approach and deal with the word-senses inside each context window, one at a time. Thus their processing time is, at some level, dependent on the number of word-senses in the context.

In our pipeline, we train a synset vector representation model from scratch, so we do not take advantage of pre-trained models (e.g. Google News, GloVe). Approaches like [133, 145] follow this direction and do save some time in the disambiguation task. However, they are not able to produce their own vector representation directly from the training corpus. In addition, their approaches require some parameter tuning, adding a certain complexity to the system as a whole. In contrast, our techniques can be applied directly to any training corpus, resulting in a new representation that can be transferred to several different problems. For example, we can apply the MSSA to better represent sentences provided to chat bots in support systems and improve the quality of their answers. Additionally, MSSA, MSSA-NR, and MSSA-D are completely unsupervised, so they do not rely on any parameters other than those required in a word embeddings algorithm.

Finally, another important aspect explored in our approach is the recurrent aspect of

MSSA-NR. To the best of our knowledge, the ability to iteratively use the produced synset vectors to improve the word sense disambiguation task and provide a more refined synset annotation is not explored in any of the compared systems. This opens many new directions on how word embeddings and word sense disambiguation can mutually benefit from each other.

4.3 Document Classification Task

In this section, we explain all details and constraints in which our experiments for document classification are performed, so one can reproduce them as close to our configuration as possible. First, we describe the different aspects of each dataset used, their characteristics, references, and availability. Second, we provide an accurate description of the chosen metrics, machine learning classifiers, and hyperparameters adopted. In addition, we also include the procedures considered to fine-tune all classifiers and discuss the main aspects of state-the-art systems compared to our proposed techniques.

4.3.1 Datasets Details

Our experiments consider 6 different datasets with specific characteristics which impose a particular challenge on each classification. Among these datasets, 4 of them represent benchmark datasets widely used on text classification problems and the other 2 are extracted from scientific papers abstracts in biology. Table 4.8 describes the main characteristics of each dataset.

Ohsumed. This dataset is composed of medical scientific paper abstracts of MEDLINE from 1991 [79]. 56,984 documents are unevenly divided into 23 classes, so some categories have more documents than others. The labels of this corpus are Medical Subject Headings (MeSH) from the cardiovascular diseases group. This dataset also has several different versions available, so we choose the one in which all documents are about cardiovascular diseases[12].

20Newsgroups. This dataset is another well-known collection in text classification problems [85, 114, 78]. 18,846 documents are split into 20 categories (e.g. sport, religion, computer) with a different number of documents in each category. We consider the "bydate" organization[13], in which the original dataset containing 20,000 documents has some duplicates and headers removed.

Reuters-21578. This dataset is one of the most popular datasets in the text classification task. 12,902 documents composed of news articles from the Reuters Newswire from 1987 are divided into 90 classes. However, we select the largest 10 categories in the dataset, resulting in 9,980 documents in total[14].

BBC. This dataset is composed of news from the BBC News website from 2004 to 2005 [59]. 2,225 documents are divided into five topic areas (e.g. sports, news, politics). Several different versions are available for this dataset, but we use the raw text files one[15].

ScyGenes. This corpus is composed of abstracts of scientific papers about the yeast Saccharomyces cerevisiae [106, 107]. This is a small, but well curated dataset with 1,114 documents divided into 7 categories. Each category represents the name of a gene of this yeast.

ScyClusters. This dataset is a variation of the ScyGenes corpus and composed of papers containing the yeast Saccharomyces cerevisiae [106, 107]. 1,655 abstracts are divided into 7 categories as well. However, its categories are organized in a different fashion, representing a cluster of the genes with the same biological function.

[12]http://disi.unitn.it/moschitti/corpora.htm
[13]http://qwone.com/ jason/20Newsgroups/
[14]http://disi.unitn.it/moschitti/corpora.htm
[15]http://mlg.ucd.ie/datasets/bbc.html

Table 4.8: Technical details about the datasets after pre-processing

Corpus	Subject	#docs	#classes	#tokens	#synsets
Ohsumed	Medical abstracts	56984	23	64154	36395
20Newsgroups	News	18846	20	129782	43413
Reuters-21578	News	9980	10	24273	21747
BBC	News	2225	5	29126	29151
ScyClusters	Biological abstracts	1655	7	13265	11428
ScyGenes	Biological abstracts	1114	7	10553	10045

Table 4.8 shows the technical aspects of each dataset with respect to their theme, number of documents, classes, tokens, and synsets. We use the term *tokens* instead of words because some features do not represent a proper word itself. As an example, let us consider the token *housd* that does not represent an English word, but it is included in the dataset. Techniques that rely on proper syntax are not able to process typos and malformed word-tokens, but recent word embeddings models, such as ELMo [131] and USE [29] can handle these issues. Column $\#synsets$ shows the number of synsets generated applying the MSSA algorithm (Section 3.2). As a result, the number of synsets is different when compared to the number of tokens, and in most cases, it is smaller. This happens because not every token (e.g. *housd*) is represented in WordNet, which forces them to be dropped since no semantic relation can be extracted.

4.3.2 Machine Learning Classifiers

In order to validate the consistency and robustness of the techniques, we consider five classifiers in our experiments. These classifiers are chosen among the most popular ones in the document classification arena: K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), Logistic Regression (LR), Random Forests (RF), and Naïve Bayes (NB).

***K*-Nearest Neighbors** [4] is a simple yet efficient classifier that uses the *K* closest data points to decide the label of a new unseen instance. K-NN is considered a lazy algorithm since it does not learn any function from the training data, but instead, it memorizes the characteristics of it.

Support Vector Machine [35] is a popular classifier applied in text categorization. SVM has a good generalization of the training dataset when combined with a proper kernel choice. The basic idea behind SVM's algorithm is to find the best hyperplane that better separates the data, after applying a series of transformations to it. These transformations are intended to represent the current data points in a higher-dimensional space, in the hope that, due to data sparsity in high dimensions, a hyperplane might be able to properly separate them.

Logistic Regression classifier [104] applies a linear combination of weighted input variables to predict the output. The main idea is to find a probabilistic relationship between the input features so they can be used to predict the output with the largest probability. Considering the text classification problem, the input variables are the features of documents (usually words) and the output labels their labels. During the training phase, the logistic classifier goal is to learn the best values for the weights of the inputs. Later, these input weights are used on the features of the test documents to predict their labels.

Random Forests [22] creates a collection of decision trees that will together find the classification for a data point. Each decision tree will create rules upon the input variables to decide their label. The main idea behind Random Forests is to combine several simple decision trees into a single classifier to define the final label.

Naïve Bayes [114] is a simple probabilistic algorithm that assumes its variables are conditionally independent. Hence, it calculates the probability of a document to belong to a specific class, based on the conditional independence of its features.

All classifiers have several parameters that need to be fine-tuned to provide the best configuration possible for each technique. For this reason, we perform an extensive grid-search on the training corpus to find out the most suitable hyperparameters for each classifier. With the exception of Reuters-21578 and 20Newsgroups, all data sets do not include a training and test split, so we applied *k*-fold-cross-validation for *k* equals to 10 (10-fold-CV). Table 4.9 shows the grid search configuration considered. Each final classification is performed considering the best hyperparameters found.

Table 4.9: Grid-search configuration parameters.

Classifier	Parameter	Range
K-NN	neighbors	1, 5, 15, 25 ... 95
Logistic Regression	solver	newton-cg, lbfgs, sag, saga
	maximum iteration	500, 1000, 1500
	multi-class	ovr, multinomial
	tolerance	0.01, 0.001, 0.0001, 0.00001
Support Vector Machine	kernel	linear, radial bases function, polynomial
	gamma	0.01, 0.001, 0.0001, 0.0001
	polynomial degree	1, 2, 3, 4, 5, 6, 7, 8, 9
	C	1, 10, 100
Random Forest	number of estimators	100, 325, 550, 775, 1000
	maximum features	auto, sqrt
	maximum depth	10, 32, 77, 100, None
	minimum samples split	2, 5, 10
	minimum samples leaf	1, 2, 4

We evaluate our models in the text classification task through their classification accuracy as metric. Accuracy measures how many documents were correctly classified during the task. This metric is defined as shown in Equation 4.1.

$$\text{Accuracy} = \frac{\text{total number of correct predictions}}{\text{total number of predictions}} \tag{4.1}$$

The classification results are obtained considering either 10-fold-CV or training and test split, depending on the availability for the dataset. The classification performance for all machine learning classifiers and datasets are also validated in the Friedman test and Nemenyi's post-hoc test for a p-value of less than 0.05, certifying the statistical significance of our experiments [37].

4.3.3 Word Embedding Models Characteristics

This section presents the main characteristics and details of the compared systems used in the document classification task (Section 4.3.6). In total, we compare our techniques against 6 state-of-the-art approaches that use transfer learning at some level in their implementation. Table 4.10 summarizes the algorithms used in our experiments, named: Latent Dirichlet Allocation (LDA) [16], word2vec [110], Global Vectors (GloVe) [130], fastText [17], Universal Sentence Encoder (USE) [29], Embeddings from Language Models (ELMo) [131], Most Suitable Sense Annotation - N Refined (MSSA-NR) [150] for N equal to 1 (M1R), FLLC II, using the M1R synset output corpus (FL-1R), and FXLC II, with a chunk size equal to 2 and using the M1R synset corpus (FX2-1R).

Table 4.10: Word embeddings used and their main characteristics. * For USE, Cer et al. [29] report its training data as a collection of sources from Wikipedia, web news, web question-answer pages discussion forums and Stanford Natural Language Inference corpus.

Algorithm	Main Characteristics	Training Corpus	Dimensions
LDA	Probability distribution	Wikipedia Dump 2010	300
word2vec	Continuous Bag-of-Words (CBOW)	Google News	300
GloVe	Word-word co-occurrence matrix	Wikipedia Dump 2014 + Gigaword 5	300
fastText	Skip-gram	Wikipedia Dump 2017 + UMBC	300
USE	Deep Average Network	Various sources*	512
ELMo	Bidirectional Long Short Term Memory	1 Billion Word Benchmark	1024
M1R	Most Suitable Sense Annotation - NR	Wikipedia Dump 2010	300
FL-1R	Flexible Lexical Chains II + CBOW	Wikipedia Dump 2010 (M1R)	300
FX2-M1R	Fixed Lexical Chains II + CBOW	Wikipedia Dump 2010 (M1R)	300

Our experiments are set up to use the same baseline structure, in which they transform each word in our documents into a vector representation using an external word embeddings trained model as a dictionary. Once we obtain these word vectors, we average them to represent the entire document as one single entity. The compared systems in our experiments can be categorized into two major groups: (i) pre-trained word embeddings models and (ii) explicitly trained word embeddings models. In the first category (i), we use models that are accessed directly as a black-box: word2vec[16], GloVe[17], fastText[18], USE[19], and ELMo[20]. In the second group (ii), we train the word embeddings models from scratch, for the following techniques: LDA, M1R, FL-1R, and FX2-1R. With the exception of USE and ELMo, that only provide their pre-trained models with 1024 and 512 dimensions respectively, all other techniques have a 300 dimension word vector representation. The training corpus considered and the other parameters for our trained word embeddings models are the same as the ones described in Section 4.1.2. We also compare our techniques with a BOW approach in a separate experiment using the same classifiers and the original datasets (i.e. words). In addition to the presented models explored in the document classification task, we also include variations of our techniques under a different scope to evaluate their characteristics more carefully.

4.3.4 Document Embeddings Models Characteristics

We also compare our proposed approaches against techniques that provide document embeddings explicitly. In order to maintain a standard structure between our experiments, we consider the same datasets described in Section 4.3.1 for the document classification task. For this scenario, we use the same training corpus in which the experiments considering word embeddings are performed (Wikipedia Dump from 2010). However, instead of obtaining the document representation through a word2vec training model, we consider a PV [87] implementation, also referenced as doc2vec. We compare the document-based version of our techniques (M1R, FL-1R, and FX2-1R) against three baselines, named: PV-DBOW, ELMo, and USE. The PV-DBOW is produced considering the raw words from our external corpus (WD10), which are also used to derive the vectors for our techniques. Different from

[16]https://code.google.com/archive/p/word2vec/
[17]https://nlp.stanford.edu/projects/glove/
[18]https://fasttext.cc/docs/en/english-vectors.html
[19]https://tfhub.dev/google/universal-sentence-encoder/2
[20]https://tfhub.dev/google/elmo/2

the word embeddings representation in which we average the constituent word vectors to represent a document, ELMo and USE are modified to produce document vectors directly. In other words, for each Wikipedia article used as an input, we produce document vectors of fixed length.

Since the training process for entire documents uses a specific encoder, the hyperparameters considered are also different from a traditional word2vec implementation. The PV-DBOW (word base corpus), M1R, FL-1R, and FX2-1R resulting corpora are trained considering: negative sampling of 10^{-5}, PV-DBOW training model, window size of 15, minimum count of 5, 300 dimensions, trained word-vectors in skip-gram fashion ($dbow\_words$), averaged word vectors (dm_mean), and 30 epochs. Parameters not mentioned use the default values in the $gensim$[21] API. As for ELMo and USE, we access their pre-trained models in the same way as Section 4.3.3 describes, but we encode the documents entirely. Since the focus of this task is on word embeddings, we do not include the main characteristics of PV in Table 4.10, but their results for the document classification task are detailed in Table 4.13.

4.3.5 Experiment Configuration

We divide our experiments into two distinct perspectives, the first for document classification (Section 4.3.6) and the second for lexical chains behavior analysis (Section 4.3.7).

In the first perspective, we present the results for all variations of our models against a traditional BOW approach. In this scenario, we are able to analyze the different chunk sizes for our lexical chains and how the recurrent characteristic of MSSA affects our built lexical chains. This evaluation also supports the choice of which of our models' configuration should be selected for the comparison against state-of-the-art systems. Thus, we compare our techniques (FL-1R, FX2-1R) with different word and document embeddings approaches that also use transfer learning. In the second experiment perspective, we provide a deeper

[21]https://radimrehurek.com/gensim/models/doc2vec.html

assessment on how our techniques behave for each dataset used. The idea is to provide a different perspective on two specific aspects: (a) chunk size for the FXLC II variations and (b) the effects of the recurrent models from MSSA in both, FLLC II and FXLC II.

All proposed models are evaluated considering the document classification task for the datasets presented in Section 4.3.1. The same pre-processing steps are performed in all datasets and techniques, so we can guarantee more consistency and uniformity in our experiments. For pre-processing, we lowercase all words in the document and remove all common English stopwords using the PyPi library (v. 2018.7.23). Each document is then represented as the average of its constituent word vectors using a word embeddings model, or directly, using the entire document for the document embeddings variation. Averaging the word vectors of a document to obtain its representation is a common strategy adopted in several publications in the document classification task [87, 36, 86, 165].

Figure 4.1 presents a high-level perspective of the entire process during the classification task in our experiments. For each document in a corpus, we retrieve its word vectors from a pre-trained word embeddings model, average them, and build a document vector. Next, we feed them to several machine learning classifiers and evaluate their accuracy. The same process is performed when considering the document embeddings. While the compared systems derive vectors directly from the words in each dataset, our proposed techniques require a disambiguated synset corpus. Thus, two additional steps prior to the machine learning classifiers are necessary. First, we disambiguate the words from each document to obtain their respective synsets, using the MSSA algorithm. Second, using the lexical chains models (FLLC II and FXLC II) created from WD10, we derive the synset vectors.

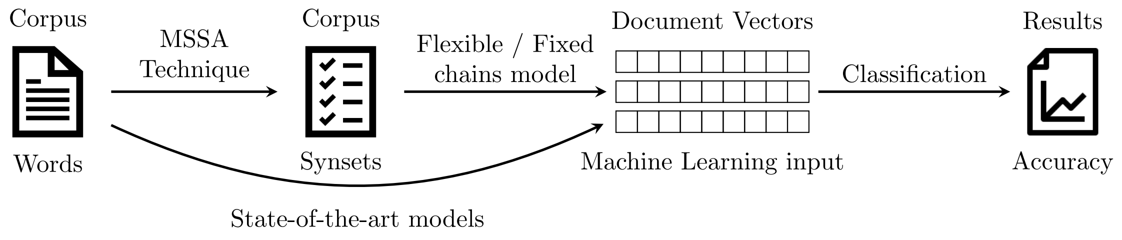


Figure 4.1: Workflow of document classification.

We compare our findings against classical and state-of-the-art word embedding techniques that consider transfer knowledge in their approach. For traditional word embeddings techniques, we use LDA [16], word2vec [109], and GloVe [130]. For the state-of-the-art word embedding techniques, we include three other approaches, named fastText [17], USE [29], and ELMo [131]. As for traditional document embeddings, we use PV-DBOW [87], ELMo [131], and USE [29] to compare with our techniques, considering the details described in Section 4.3.4. Furthermore, we also provide a comparison of our results against a traditional BOW technique, which relies only on statistical information of the training corpus. Even though BOW does not necessarily use transfer learning from a prior task, it is probably the most used technique to represent a collection of documents in NLP. In addition, the BOW technique suffers from the curse of dimensionality with respect to the number of words in the considered vocabulary. In this manner, depending on the size of the dataset, its adoption might not be suitable. To mitigate the dimensionality problem and allow a fair comparison among the compared systems, we created a BOW representation considering the top 300 features (words), ordered by term frequency, and applying tf-idf as its weighting scheme [128]. The only two models that do not follow this vector representation are ELMo and USE, with 1024 and 512 dimensions respectively.

Another important step when using document embeddings through a PV implementation (i.e. doc2vec) is the generation of document vectors for unseen data when using *gensim*. Different from word2vec, that either retrieves or not an n-dimensional vector for a given word, PV tries to infer its vectors for each document using its pre-trained model. In other words, we can say there are two training steps when considering doc2vec: (i) external document embeddings training and (ii) inferred task oriented embeddings. The former, follows the same training steps pattern as word2vec with respect to its specific parameters (Section 4.3.4). As for the latter, we produce a vector representation for an unseen document based on the pre-trained model obtained in (i) using the routine $infer\_vectors$ in $gensim$ [137]. Thus, specific hyperparameters have to be adjusted. Our inferred vectors, for each document in the datasets used for the document classification task, are produced with the following hyperparameters: $alpha = 10^{-4}$, $min\_alpha = 10^{-6}$, and 300 $epochs$. These values are based on extensive empirical experiments studying the correlation between their values and the accuracy in sample datasets. We justify the choice of PV-DBOW over PV-DM because of its superiority in the semantic similarity task [86]. All training models using doc2vec ($gensim$ version of PV) consider the same hyperparameter configuration. Attributes not mentioned are used with their default values in $gensim$[22].

4.3.6 Document Classification Task Results

Tables 4.11 and 4.12 present the results of our experiments considering the document classification task for the word embeddings techniques. Table 4.13 illustrates the results using the document embeddings setup. The results are organized as follows. Each block illustrates the results applying all classifiers in a specific dataset. Each column represents a different word/document embeddings model benchmark to compare against our techniques. Values in **bold** represent the best results in a row and underlined values the best results in a column for a specific dataset.

As explained in Section 3.3, our proposed techniques are built considering a synset corpus, which for the presented experiments is obtained through the MSSA algorithm. In one variation of MSSA, called MSSA-NR, it is possible to control the number of times a produced synset embeddings model is used to refine the disambiguation step. Thus, we can generate a more robust semantic representation that can be used to train a new enhanced synset embeddings model. In MSSA-NR, we consider $0 \leq N \leq 2$, where N represents how many iterations the recurrent process is performed. For the FXLC II algorithm (Section 3.3.2), we also consider the sizes of 2, 4, and 8 for the number of synsets in each chain ($chunk\_size$). In order to evaluate the best configuration for the proposed techniques, we first compare them with a traditional BOW with a tf-idf weighting scheme of 300

[22]https://radimrehurek.com/gensim/models/doc2vec.html

dimensions. We keep only 300 dimensions so all models can be compared under similar constraints. The focus of our approaches and all compared systems is towards transfer learning from an external corpora, scalability, and dense vector representations; none of which BOW shares. BOW needs to be executed directly in the training corpus, it has a sparse representation, with respect to the considered vocabulary, and does not generalize well in real-world scenarios where many OOV words might lead to poor results. Therefore, instead of using BOW as direct compared system, we use it as a guide to assist us in evaluating which of our approaches should be investigated in details agaisnt state-of-the-art systems.

As Table 4.11 shows, BOW sustains good results for ScyGenes, and a few for ScyClusters. This behavior seems reasonable since ScyGenes and ScyClusters are the smallest datasets considered. These datasets might contain unique keywords among their categories, resulting in a good classification for the BOW approach. On the other hand, as the number of documents increases, our semantic embedding representations start to overcome BOW. We believe this is due to the number of existing words in the corpora; BOW is unable to extract features that lead to a better classification. When considering the largest datasets, our techniques start to improve their results. If we consider each row on Table 4.11, FLLC II and FXLC II together present a better accuracy score in 23 out of 30 cases. Among all of these results, we can observe that FXLC II with a chunk size of 2 and FLLC II built over MSSA-1R have the majority of the best results, named FX2-1R and FL-1R respectively. Thus, we use FX2-1R and FL-1R to compare against the other state-of-the-art models using word embeddings.

Table 4.11 can be represented in a different perspective to highlight the results of our lexical chains. In Figure 4.2, we show the number of times each technique obtains the best results for a given classifier and dataset. As previously mentioned, the techniques using FXLC II with $chunk\_size$ = 2 obtain higher results than the other variations. More specifically, the ones considering MSSA-NR for N = 1 comprise one third of the top-ranked results, thus supporting their choice for comparison with state-of-the-art techniques. The

Table 4.11: Classification accuracy for BOW approach against the proposed techniques for each classifier and dataset. Values in **bold** represent the best result of that row. Underlined values represent the best value for that dataset. K-NN - K Nearest Neighbors; RF - Random Forest; LR - Logistic Regression; SVM - Support Vector Machine; NB - Naïve Bayes.

	BOW	FL-0R	FL-1R	FL-2R	FX2-0R	FX2-1R	FX2-2R	FX4-0R	FX4-1R	FX4-2R	FX8-0R	FX8-1R	FX8-2R
Ohsumed	0.3486	0.4402	0.4412	0.4389	0.4383	0.4416	0.4399	0.4326	0.4376	0.4316	0.4174	0.4252	0.4186
K-NN	0.3187	0.3953	0.3967	0.3936	0.4007	**0.4023**	0.4012	0.3940	0.3993	0.3986	0.3832	0.3890	0.3847
RF	0.3486	0.3407	0.3397	0.3380	0.3486	**0.3540**	0.3495	0.3462	0.3505	0.3483	0.3381	0.3419	0.3374
LR	0.3441	0.4123	0.4169	0.4151	**0.4283**	0.4196	0.4182	0.4097	0.4137	0.4148	0.4032	0.4084	0.4024
SVM	0.3364	0.4402	0.4412	0.4389	0.4383	**0.4416**	0.4399	0.4326	0.4376	0.4316	0.4174	0.4252	0.4186
NB	0.1662	0.3091	0.3065	0.3079	0.3157	0.3218	**0.3224**	0.3097	0.3199	0.3158	0.3005	0.3075	0.3069
20Newsgroups	0.5458	0.7135	0.7147	0.7167	0.7151	0.7272	0.7196	0.7066	0.7168	0.7059	0.6980	0.6990	0.7106
K-NN	0.4316	0.6312	0.6329	0.6298	0.6353	**0.6447**	0.6371	0.6333	0.6377	0.6321	0.6175	0.6251	0.6224
RF	0.5458	0.6701	0.6742	0.6642	0.6706	**0.6802**	0.6792	0.6722	0.6706	0.6749	0.6601	0.6689	0.6595
LR	0.5252	0.7067	0.7147	0.7131	0.7151	**0.7272**	0.7187	0.7066	0.7168	0.7059	0.6980	0.6986	0.7069
SVM	0.5096	0.7135	0.7114	0.7167	0.7035	0.7192	**0.7196**	0.6994	0.7115	0.7038	0.6974	0.6990	0.7106
NB	0.3946	0.5860	0.5884	0.5890	0.5821	0.5916	0.5839	0.5883	**0.5928**	0.5834	0.5755	0.5817	0.5801
Reuters-21578	0.8683	0.8719	0.8726	0.8719	0.8701	0.8708	0.8664	0.8672	0.8690	0.8672	0.8586	0.8633	0.8644
K-NN	0.8378	**0.8558**	0.8554	0.8539	0.8489	0.8518	0.8493	0.8443	0.8468	0.8425	0.8328	0.8410	0.8428
RF	0.8561	0.8543	0.8525	0.8504	0.8522	0.8550	0.8532	0.8492	**0.8558**	0.8540	0.8407	0.8443	0.8428
LR	0.8683	0.8698	**0.8726**	0.8676	0.8672	0.8708	0.8637	0.8622	0.8644	0.8647	0.8540	0.8607	0.8626
SVM	0.6232	**0.8719**	0.8640	**0.8719**	0.8701	0.8637	0.8664	0.8672	0.8690	0.8672	0.8586	0.8633	0.8644
NB	0.7718	0.8116	**0.8120**	0.8044	0.7937	0.7980	0.7998	0.7973	0.8027	0.7987	0.7923	0.7962	0.7933
BBC	0.9524	0.9784	0.9784	0.9771	0.9775	0.9757	0.9784	0.9771	0.9766	0.9771	0.9744	0.9753	0.9739
K-NN	0.9097	0.9604	0.9600	0.9627	0.9627	0.9573	0.9627	0.9595	0.9623	**0.9650**	0.9627	0.9591	0.9609
RF	0.9421	0.9645	0.9667	0.9636	0.9686	**0.9703**	0.9667	0.9667	0.9667	0.9649	0.9658	0.9654	0.9672
LR	0.9524	**0.9784**	**0.9784**	0.9753	0.9757	0.9757	**0.9784**	0.9757	0.9766	0.9748	0.9730	0.9753	0.9739
SVM	0.9510	**0.9780**	0.9771	0.9771	0.9775	0.9766	0.9775	0.9771	0.9753	0.9771	0.9744	0.9748	0.9717
NB	0.9137	0.9474	0.9456	0.9465	0.9447	0.9469	**0.9501**	0.9402	0.9469	0.9461	0.9452	0.9411	0.9452
ScyClusters	0.6997	0.6930	0.6822	0.6827	0.6990	0.6936	0.6991	0.6738	0.6723	0.6833	0.6749	0.6625	0.6669
K-NN	**0.6470**	0.6430	0.6165	0.6340	0.6283	0.6400	0.6424	0.6146	0.6322	0.6216	0.6061	0.6085	0.6162
RF	**0.6997**	0.5975	0.5940	0.5885	0.6077	0.6073	0.6065	0.6053	0.5968	0.6077	0.5908	0.5830	0.5914
LR	0.6616	0.6845	0.6822	0.6827	**0.6990**	0.6936	0.6916	0.6670	0.6711	0.6833	0.6693	0.6506	0.6669
SVM	0.6610	0.6930	0.6701	0.6815	0.6802	0.6905	**0.6991**	0.6738	0.6723	0.6678	0.6749	0.6625	0.6681
NB	0.5329	0.5354	**0.5385**	0.5324	0.5233	0.5329	0.5348	0.5069	0.5154	0.5311	0.4955	0.5059	0.5051
ScyGenes	0.9767	0.8474	0.8521	0.8457	0.8502	0.8456	0.8420	0.8394	0.8494	0.8411	0.8160	0.8330	0.8294
K-NN	**0.9094**	0.7829	0.7750	0.7783	0.7856	0.7903	0.7731	0.7551	0.7758	0.7785	0.7461	0.7675	0.7642
RF	**0.9767**	0.7727	0.7603	0.7597	0.7756	0.7659	0.7674	0.7612	0.7694	0.7711	0.7549	0.7624	0.7523
LR	**0.9309**	0.8454	0.8521	0.8457	0.8502	0.8456	0.8420	0.8394	0.8494	0.8411	0.8160	0.8330	0.8294
SVM	**0.9282**	0.8474	0.8475	0.8450	0.8494	0.8402	0.8413	0.8279	0.8402	0.8267	0.8100	0.8213	0.8234
NB	**0.9229**	0.6858	0.6895	0.6967	0.7021	0.7047	0.7048	0.6894	0.7083	0.7001	0.6803	0.6860	0.6866

fact that FXLC II for $chunk\_size$ = 8 does not give the best results in all circumstances tells us that the dimensionality reduction in these cases loses too much semantic information for the classification task.

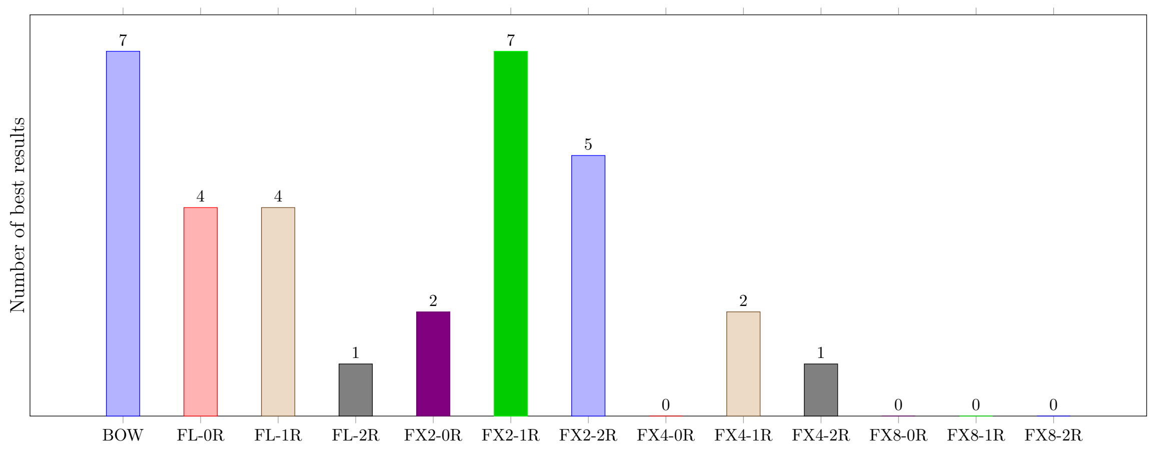


Figure 4.2: Document classification ranking considering BOW and lexical chains techniques.

In Table 4.12, we present the best results of our Chains2Vec embedding models against state-of-the-art techniques that also use word embeddings. Considering the Ohsumed dataset, our approaches outperform all the other techniques for all classifiers baselines, which have a considerably inferior result. The Ohsumed text collection represents a difficult real-world scenario because of its size, number of classes, and constituent words [184]. In this dataset, our fixed lexical chains of size 2 (FX2-1R) achieves the best results for all classifiers. For the 20Newsgroups dataset, our techniques overcome the baselines for the K-NN, RF and NB, while fastText is superior considering LR and SVM.

In a similar manner to the BOW rank in Figure 4.2, we also provide a comparison of our proposed techniques against state-of-the-art ones. Figure 4.3 illustrates how many times each approach is able to achieve the best results considering all classifiers and datasets. In 21 out of 30 experiments (70%), MSSA and the lexical chains algorithms outperform the compared systems. In particular, FX2-1R achieves the highest score in 16 of the 21 cases. We believe that the recent word embeddings techniques, such as USE and ELMo perform poorly because of their ability to handle any possible input, even OOV words. Approaches that work with more significant features are able to provide a better semantic representation. Additionally, the produced synset embeddings models produced through our techniques are at least 75% smaller than the other systems, with 250MB only. GloVe is the smallest,

Table 4.12: Classification accuracy for word embeddings models against proposed techniques for each classifier and dataset. Values in **bold** represent the best result of that row. Underlined values represent the best value for that dataset. K-NN - K Nearest Neighbors; RF - Random Forest; LR - Logistic Regression; SVM - Support Vector Machine; NB - Naïve Bayes.

	LDA	word2vec	GloVe	fastText	USE	ELMo	M1R	FL-1R	FX2-1R
Ohsumed	0.2262	0.4223	0.4136	0.4324	0.3009	0.4172	0.4357	0.4412	0.4416
K-NN	0.2138	0.3822	0.3731	0.3912	0.3009	0.3209	0.3975	0.3961	**0.4023**
RF	0.1689	0.3302	0.3258	0.3447	0.2899	0.2851	0.3411	0.3397	**0.3540**
LR	0.2262	0.3981	0.4136	0.4171	0.2792	0.4172	0.4185	0.4169	**0.4196**
SVM	0.2056	0.4223	0.3286	0.4324	0.2542	0.3194	0.4357	0.4412	**0.4416**
NB	0.0558	0.2820	0.2771	0.2785	0.1947	0.1865	0.3118	0.3065	**0.3218**
20Newsgroups	0.6340	0.7110	0.7153	0.7485	0.6476	0.6895	0.7201	0.7147	0.7272
K-NN	0.5013	0.5737	0.5425	0.6134	0.5588	0.4574	0.6320	0.6329	**0.6447**
RF	0.6340	0.6300	0.6386	0.6794	0.6476	0.5389	0.6753	0.6742	**0.6802**
LR	0.5498	0.6657	0.7152	**0.7288**	0.5447	0.6895	0.7167	0.7147	0.7272
SVM	0.5388	0.7110	0.7153	**0.7485**	0.5171	0.4981	0.7201	0.7114	0.7192
NB	0.4624	0.4426	0.4133	0.5104	0.4401	0.2677	0.5895	0.5884	**0.5916**
Reuters-21578	0.8260	0.8805	0.8830	0.8802	0.8278	0.8780	0.8719	0.8726	0.8708
K-NN	0.7919	**0.8680**	0.8640	0.8593	0.8267	0.8436	0.8568	0.8554	0.8518
RF	0.8260	0.8561	**0.8619**	0.8536	0.8278	0.8281	0.8550	0.8525	0.8550
LR	0.7875	0.8698	**0.8776**	0.8705	0.7370	0.8751	0.8651	0.8726	0.8708
SVM	0.8041	0.8805	**0.8830**	0.8802	0.7951	0.8780	0.8719	0.8640	0.8637
NB	0.6024	0.7740	**0.8224**	0.8034	0.7725	0.7603	0.8009	0.8120	0.7980
BBC	0.9552	0.9708	0.9784	0.9766	0.9672	0.9743	0.9780	0.9784	0.9766
K-NN	0.9304	0.9591	**0.9622**	0.9618	0.9577	0.9497	0.9596	0.9600	0.9573
RF	0.9552	0.9532	0.9631	0.9663	0.9672	0.9573	0.9681	0.9667	**0.9703**
LR	0.9241	0.9478	0.9690	0.9681	0.9370	0.9649	0.9766	**0.9784**	0.9757
SVM	0.9295	0.9708	**0.9784**	0.9766	0.9474	0.9743	0.9780	0.9771	0.9766
NB	0.8680	0.9218	0.9483	0.9469	0.9442	0.9195	**0.9501**	0.9456	0.9469
ScyClusters	0.4814	0.6410	0.6391	0.6645	0.4966	0.6612	0.7027	0.6822	0.6936
K-NN	0.4137	0.5919	0.5903	0.5777	0.4835	0.5675	0.6267	0.6165	**0.6400**
RF	0.4814	0.5479	0.5469	0.5890	0.4966	0.5317	0.5889	0.5940	**0.6073**
LR	0.3692	0.5879	0.6168	0.6162	0.3475	0.6612	0.6839	0.6822	**0.6936**
SVM	0.3439	0.6410	0.6391	0.6645	0.3439	0.3650	**0.7027**	0.6701	0.6905
NB	0.2362	0.4769	0.5185	0.4790	0.3910	0.4670	0.5373	**0.5385**	0.5329
ScyGenes	0.6104	0.7988	0.7961	0.8301	0.6460	0.8761	0.8556	0.8521	0.8456
K-NN	0.4573	0.7085	0.7120	0.7281	0.6105	0.7480	0.7866	0.7750	**0.7903**
RF	0.6104	0.6849	0.7363	0.7308	0.6460	0.7423	0.7649	0.7603	**0.7659**
LR	0.3488	0.6346	0.7746	0.7477	0.3044	0.8509	**0.8556**	0.8521	0.8456
SVM	0.3471	0.7988	0.7961	0.8301	0.2442	**0.8761**	0.8538	0.8475	0.8402
NB	0.3766	0.6326	0.6743	0.6249	0.5725	0.7028	0.6977	0.6895	**0.7047**

with 1GB and word2vec (Google News) is the largest with 3.5GB. In the case of document embeddings (PV-DBOW), the models trained using raw words for WD10 have more than 6.3GB.

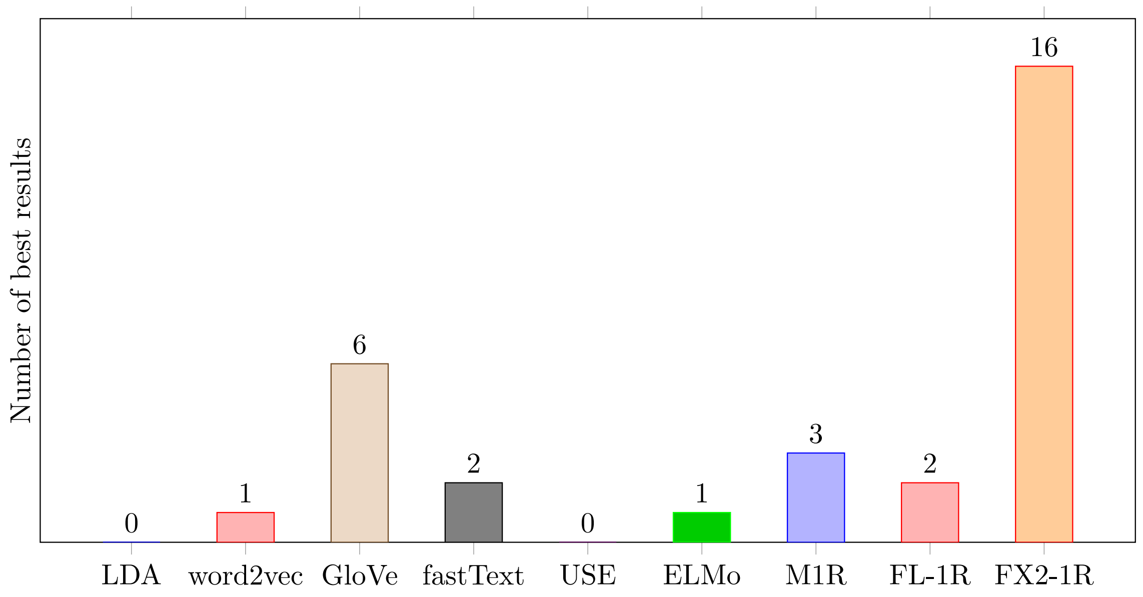


Figure 4.3: Document classification ranking considering state-of-the-art for word embeddings and lexical chains techniques.

Considering the Reuters-21578 dataset, GloVe achieves the best results in 4 out of 5 classifiers. This finding makes sense if we consider how GloVe uses a co-occurrence approach to build its vectors. Several documents in this dataset are composed of short phrases, which prevents our techniques from deriving a good semantic representation. As a consequence, our chains end up with poor accuracy. However, we are still able to outperform techniques that can embed any given token, such as ELMo and USE. On BBC, the synset embedding models present the highest accuracy in 4 of the 5 classifiers, in which our lexical chains representation has the best scores for RF and LR. The documents composing BBC dataset are extracted from BBC News articles and are written using formal and cohesive English, which is beneficial to us. A cohesive text structure contributes to the semantic representation in our techniques since they rely on the lexical information from on WordNet.

Finally, on ScyClusters and ScyGenes datasets our lexical chains show superior results in 7 out of 10 experiments. If we consider word2vec and FX2-1R, for the K-NN classifier, our technique shows an improvement of 15% and 9% for the ScyClusters and ScyGenes datasets respectively. These results suggest that the semantic relations extracted through our algorithms indeed improve the quality of a standard word embeddings technique. As in the BBC and Oshumed datasets, ScyClusters and ScyGenes are also composed of formal, cohesive and typo-free English documents. As a result, the synset techniques, especially the proposed ones, are able to properly extract the semantic relations within the documents.

A natural direction after training the many variations of our techniques, through a word2vec implementation, is to explore different embeddings algorithms. For this, as explained in Section 4.3.4, we use a PV-DBOW [87] implementation to obtain the document vectors from our synset corpus. USE and ELMo are also modified so they can provide vector representations of documents in the datasets directly (i.e. without the need of averaging all its constituent words vectors). As Table 4.13 shows, our techniques do not generalize as well as in word2vec. In fact, most of the best results are achieved with a vanilla PV-DBOW implementation, even when considering the datasets in which our techniques present the best results for the word embeddings experiments (i.e. Oshumed, 20Newsgroups, ScyClusters, and ScyGenes).

In a overall perspective, if we consider MSSA-NR, FLLC II, and FXLC II (chunk size of 2 and 4) we do achieve the highest accuracy when comparing with USE and ELMo, as Figure 4.4 illustrates. However, this is not a fair comparison since we are using four variations to justify our results, while the others are being evaluated as single techniques. The experiments using document embeddings techniques give us two broad directions to pursue: (i) explore the behavior of our techniques using character/sub-word level embeddings (e.g. fastText, USE, ELMo) and (ii) improve/modify the current versions of MSSA, FLLC II, and FXLC II to a document level structure. On top of that, we also contemplate other NLP task that we want to explore, such as sentiment analysis, text summarization, and plagiarism detection.

Table 4.13: Classification accuracy for document embeddings models against proposed techniques for each classifier and dataset. Values in **bold** represent the best result of that row. Underlined values represent the best value for that dataset. K-NN - K Nearest Neighbors; RF - Random Forest; LR - Logistic Regression; SVM - Support Vector Machine; NB - Naïve Bayes.

	PV-DBOW	USE	ELMo	M1R	FL-1R	FX2-1R	FX4-1R
Ohsumed	0.4344	0.4001	0.4067	0.4283	0.4289	0.4275	0.4190
K-NN	**0.4103**	0.3822	0.3266	0.3949	0.3946	0.3966	0.3908
RF	0.3213	**0.3330**	0.2903	0.3189	0.3195	0.3202	0.3075
LR	**0.4274**	0.3760	0.4017	0.4215	0.4211	0.4207	0.4138
SVM	**0.4344**	0.4001	0.4067	0.4283	0.4289	0.4275	0.4190
NB	0.2681	**0.2885**	0.2157	0.2541	0.2564	0.2613	0.2568
20Newsgroups	0.7558	0.7551	0.7097	0.7162	0.7164	0.7246	0.7226
K-NN	0.6614	**0.7494**	0.5610	0.6011	0.6024	0.6107	0.5901
RF	0.7178	**0.7460**	0.6193	0.6707	0.6678	0.6830	0.6687
LR	0.7523	**0.7551**	0.7097	0.7162	0.7141	0.7181	0.7203
SVM	**0.7558**	0.7490	0.6968	0.7154	0.7164	0.7246	0.7226
NB	0.4180	**0.7190**	0.4932	0.3981	0.3951	0.3935	0.4021
Reuters-21578	0.8654	0.8575	0.8808	0.8615	0.8604	0.8600	0.8485
K-NN	0.8403	0.8442	**0.8600**	0.8406	0.8493	0.8424	0.8381
RF	0.8449	0.8421	0.8550	0.8500	**0.8557**	0.8550	0.8485
LR	0.8432	0.8496	**0.8790**	0.8295	0.8306	0.8317	0.8353
SVM	0.8654	0.8575	**0.8808**	0.8615	0.8604	0.8600	0.8625
NB	0.6490	0.8026	**0.8249**	0.6411	0.6340	0.6372	0.6458
BBC	0.9753	0.9735	0.9748	0.9762	0.9771	0.9762	0.9744
K-NN	**0.9645**	0.9560	0.9559	0.9556	0.9574	0.9578	0.9551
RF	**0.9744**	0.9677	0.9708	0.9686	0.9690	0.9677	0.9677
LR	0.9753	0.9726	0.9748	0.9739	0.9730	**0.9757**	0.9744
SVM	0.9748	0.9735	0.9739	0.9762	**0.9771**	0.9762	0.9721
NB	0.9537	**0.9591**	0.9564	0.9430	0.9385	0.9407	0.9416
ScyClusters	0.5686	0.4960	0.5418	0.5938	0.5873	0.5892	0.6054
K-NN	0.5244	0.4484	0.4204	0.5476	0.5502	**0.5511**	0.5511
RF	0.5134	0.4337	0.4380	**0.5188**	0.5141	0.5097	0.4851
LR	0.5599	0.4326	0.5418	0.5684	0.5691	**0.5715**	0.5667
SVM	0.5686	0.4960	0.5284	0.5938	0.5873	0.5892	**0.6054**
NB	0.4767	0.3862	0.3133	0.4826	0.4916	0.4964	**0.4970**
ScyGenes	0.8597	0.6333	0.7926	0.8505	0.8502	0.8450	0.8198
K-NN	**0.7766**	0.5766	0.5633	0.7596	0.7507	0.7551	0.7457
RF	**0.7813**	0.5819	0.6167	0.7627	0.7546	0.7479	0.7398
LR	**0.8227**	0.5531	0.7926	0.8156	0.8092	0.7920	0.7839
SVM	**0.8597**	0.6333	0.7635	0.8505	0.8502	0.8450	0.8198
NB	**0.6875**	0.5176	0.5331	0.6671	0.6697	0.6640	0.6632

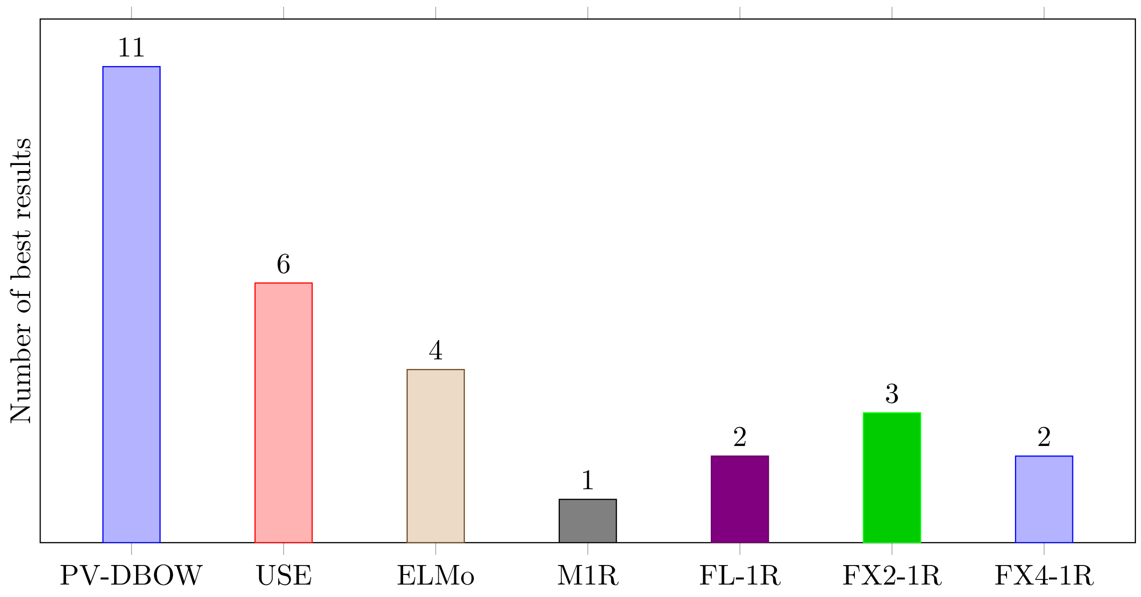
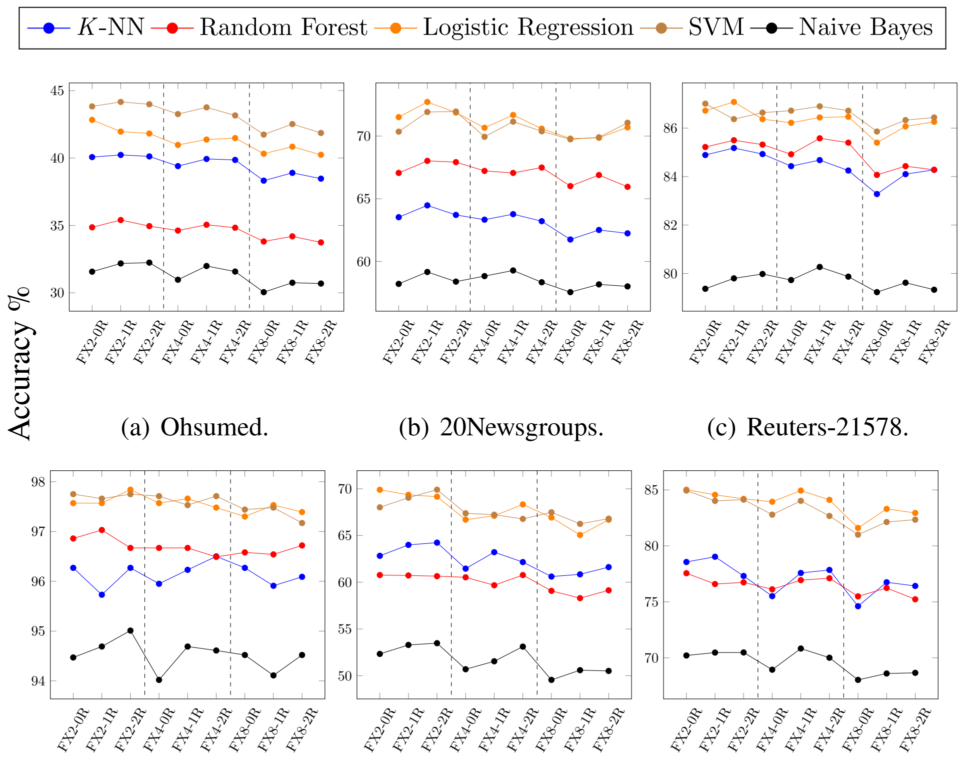


Figure 4.4: Document classification ranking considering state-of-the-art for document embeddings and lexical chains techniques.

4.3.7 Lexical Chains Behavior Analysis

In this section, we provide a deeper investigation of our proposed techniques and all the details of their internal configuration. We still keep the same machine learning classifiers and datasets to maintain consistency in our comparisons. The main idea is to provide a different perspective on how the chunk size of our fixed chains and the recurrent aspect of the models, in which they are based, influence our results. We also provide the same findings for the flexible chains algorithms and for the lexical chains trained using a PV-DBOW implementation. The results in Figures 4.5 and 4.6 show the accuracy over the different variations of our approaches for the six datasets and five classifiers for the word embeddings training model. Figures 4.7 and 4.8 show the same perspective, but considering the document embeddings training model.

For the FXLC II technique, in Figure 4.5, we can see that all models built considering the MSSA-1R (-1R) synset corpus and model present an improvement if compared with their base version (-0R). This means that the refined representation of MSSA-NR indeed improve the quality of the vectors obtained through our chains. However, the same behavior is not true when we perform the recurrent model again (-2R). Even though there is a variation considering the number of iterations in the recurrent model, our fixed chains seem to be more sensitive to the change in their chunk size. In fact, if we consider each block separately (vertical dashed lines), the accuracy for each fixed chain size remains quite stable. Nevertheless, moving from 2 to 8 synsets per chain, the loss of information seems to greatly affect our techniques. In other words, the chunk size parameter in our algorithm is inversely proportional to the quality of semantic representation of our chains. Thus, resulting in a decrease in accuracy when we move from 2 to 8 synsets per chain.



(a) Ohsumed. (b) 20Newsgroups. (c) Reuters-21578.

(d) BBC. (e) ScyClusters. (f) ScyGenes.

Figure 4.5: Accuracy for fixed lexical chains for variable chunk size and number of multiple recurrent passes for word embeddings models.

We also provide the same perspective analysis considering the FLLC II technique, shown in Table 4.6. Since the only configuration applied to the flexible chains is the recurrent aspect, we have a simpler scenario if compared to the FXLC II alternative. However,

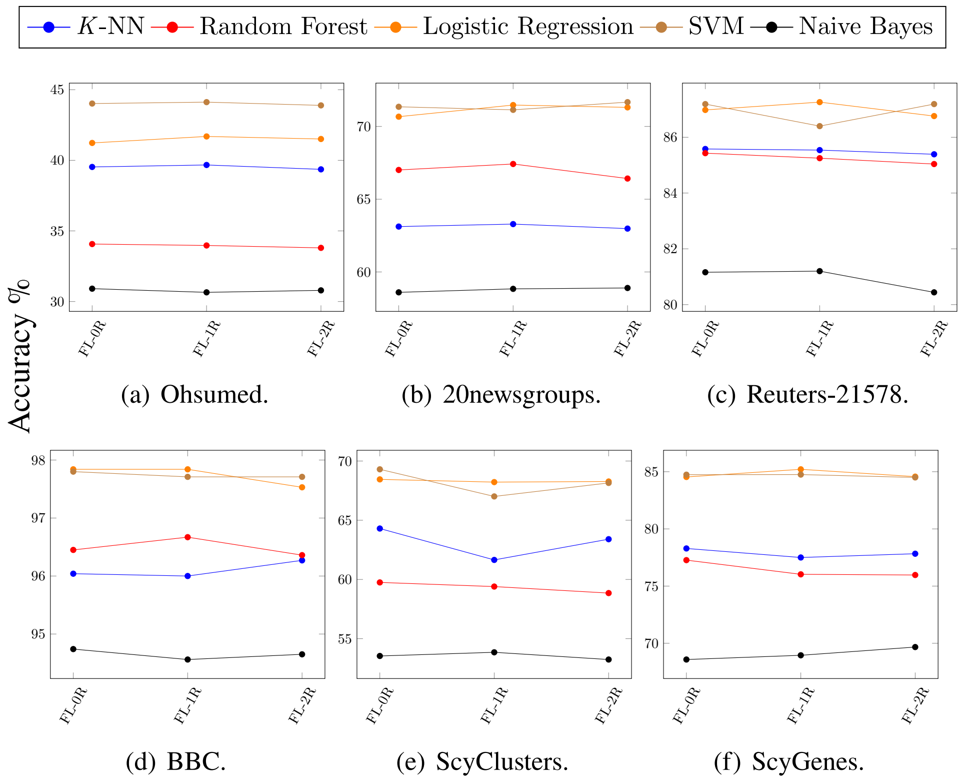


Figure 4.6: Accuracy for flexible lexical chains for multiple recurrent passes for word embeddings models.

the behavior with respect to the recurrent characteristics remains the same, in which little variation in accuracy is perceived. It seems that the enclosed cost of iteratively applying MSSA-NR is only interesting for N=1. Another common aspect for both techniques is the stability concerning the machine learning classifiers. For both, FLLC II and FXLC II, the best results are always between logistic regression and SVM, independent of the dataset, change of chunk size, and recurrence iteration.

The same behavior analysis for the lexical synset corpus trained over a PV-DBOW approach is also included, as Figures 4.8 and 4.8 show. In this analysis we do not depict any of the results of FXLC II for chunk size of 8 because of its poor performance in the document classification experiments for the word embeddings models (Table 4.12). The behavior of fixed chains for document embeddings in Figure 4.7 follows a similar pattern to the ones using word embeddings training models (Figure 4.5), with respect to the recurrent approach MSSA-NR. The accuracy seems to improve when we move from $N = 0$ to 1, but decrease when we perform a second iteration in the algorithm. In addition, the size of the chain seems to affect negatively their semantic representation, in which chains with chunk size of 2 have better performance than those with size of 4. Most datasets present similar characteristics with their word embeddings pair results, except for the ScyGenes dataset, which improves its results when its chunk size and recurrent iteration are increased. In fact, ScyGenes is the only dataset that improves all its results, for every classifier in the second iteration for the lexical chains (-2R). However, the obtained accuracy for ScyGenes is still inferior to its word embeddings pair.

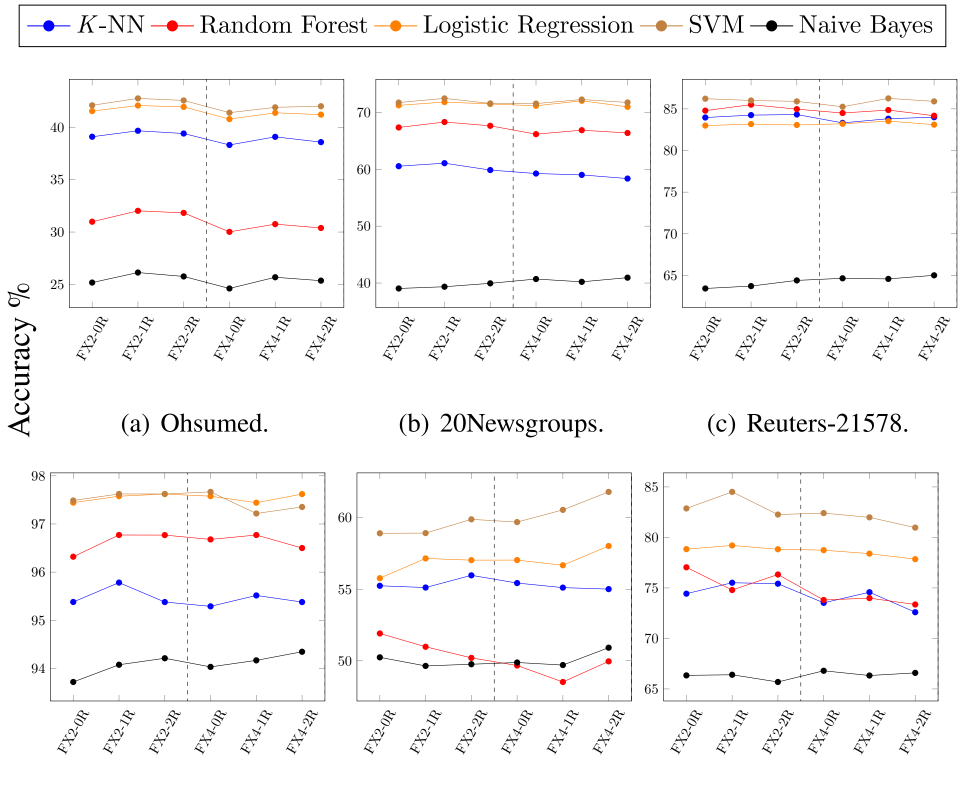


Figure 4.7: Accuracy for fixed lexical chains for variable chunk size and number of multiple recurrent passes for document embbedings models.

As for the flexible chains behavior, considering the document embeddings technique we see little or no significant difference against their word embeddings pair, as Figure 4.8

shows. The number of iterations does not seem to affect the construction of flexible chains to the point where great improvements are perceived. In other words, the cost incurred for running another pass in the training corpus is too high to justify another disambiguation and annotation step. Finally, the performance in the machine learning classifiers between chains trained in word and document embeddings seem to maintain certain stability, in which logistic regression and SVM provide the best results for all variations of our techniques.

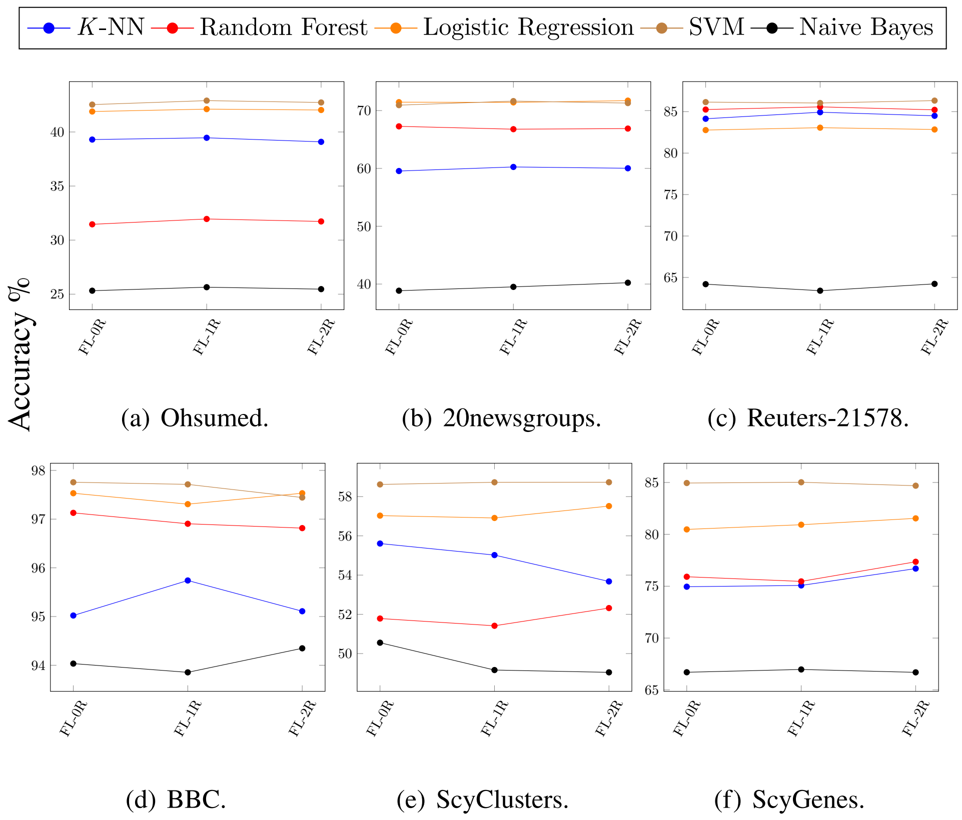


Figure 4.8: Accuracy for flexible lexical chains for multiple recurrent passes for document embeddings models.

4.4 Further Discussions and Limitations on FLLC II and FXLC II

In this section, we try to provide a deeper discussion about the main aspects of our lexical chains techniques, pointing out their strengths, and limitations, while discussing alternatives that can be taken into account to alleviate them.

The main objective of the proposed lexical chains algorithms is to represent synset-token documents through the use of different types of lexical chains. For the FLLC II technique (Section 3.3.1), we use WordNet to identify how the synsets in our documents are semantically connected. This imposes the condition that only relationships mapped in WordNet [1] can be considered, which might not reflect the true semantic value in a document. Other lexical databases, such as ConceptNet [94] and BabelNet [121] might provide alternative structures for the synsets. However, ConceptNet does not assume that words are classified in sets of synonyms, which would require drastic changes in our algorithm. On the other hand, BabelNet uses a similar synset structure to WordNet that could be explored. BabelNet also integrates different resources[23] (including WordNet). Unfortunately, because of their proprietary license, BabelNet's access is not as facilitated as ConceptNet and WordNet. It is important to mention that, for research purposes, BabelNet indexes can be downloaded upon an application request to their company. The application requires affiliation to a research institution or a Ph.D. student status, besides the non-commercial nature of the project.

One might point out that the proposed techniques only use a simple word2vec implementation to embed the synset corpora produced in FLLC II and FXLC II. Although this brings an interesting perspective, we decided to validate our algorithms using a straightforward method before moving to more complex ones, such as fastText [17], ELMo [131], and USE [29]. In fastText they propose to learn word representations as a sum of the n-grams of its constituent sub-words. This can lead to a large number non-existent tokens in WordNet, thus our approaches would not take advantage of these extra computations. In ELMo, the sub-word issue is even stronger since their words vectors are a linear combination of their characters. This would create even more noise for us, and they would not be relevant for the WordNet lexical structure. Another factor that prevents us from using ELMo, for now, is its expensive training process[24]. Closer to the word2vec technique and probably easier

[23]https://babelnet.org/about
[24]https://github.com/allenai/bilm-tf

to incorporate, GloVe [130] is also considered, but since it works building a co-occurrence matrix of words, our training corpus would make its memory consumption infeasible at the moment. In addition, we also examined the recently published USE [29], but their implementation only allows us to retrieve word vectors from their pre-calculated model, not train a new corpus. We note that we are emphasizing the use of new word embeddings techniques since experiments with document embeddings approaches show discouraging results so far (Section 4.3.6).

The synsets used to build our chains and embeddings models are based on proper English. For that reason, our approaches do not generalize that well for documents containing informal English (e.g. colloquial text, slang). An option to mitigate the lack of matches between document tokens and the embeddings model is to incorporate multiple pre-trained word embeddings, similar to what Sinoara et al. [165] adopt. However, this can lead to an overhead as large as the extra pre-trained models considered. This alternative would compromise the lightweight representation of our models. Additionally, if several word embeddings models have the same word, a ranking system would be required to order multiple occurrences of the same words.

Even though we incorporate as much as 19 semantic features in WordNet for the FLLC II algorithm (Section 3.3.1), there are still relationships that can be explored. Nevertheless, during the early stages of our research, many synset's attributes ($related\_synsets$ in Algorithm 3) did not return any associated synset. By decreasing the number of related synsets we achieve better performance during the construction of our flexible chains. In addition to the number of attributes, we do not explore deeper levels of relations for each attribute (e.g. hypernyms of hypernyms). In other words, for each synset in the related synsets we do not investigate their own related synsets.

During our experiments in Section 4.3, we noticed that the best results are obtained with datasets composed of abstracts of papers. These results reinforce the strength of our technique on corpora composed of well-written English. Even so, we still have good performance on other datasets that are composed of news from websites. In 20Newsgroups for example, we have 3 out of 5 best results. During our investigations on the experiments against BOW, we understand that our performance tends to decrease when we use large chunk size chains. A possible explanation may be because with larger fixed chains we have a great dimensionality reduction and consequently, we may be losing too much information in the process.

CHAPTER V

Final Considerations

In this work, we propose a collection of algorithms capable of extracting semantic features from natural language text using lexical chains and multi-sense embeddings. These techniques come to alleviate the *semantic deficiency gap* between data production and consumption, which increases on a daily basis. Two different aspects are explored with the proposed algorithms. The first is concerned with the disambiguation and representation of words given the influence of its immediate neighbors. The second concerns in extracting the semantic relationships between consecutive words dynamically or using pre-defined chunks. While Most Suitable Sense Annotation (MSSA), Most Suitable Sense Annotation - N Refined (MSSA-NR), and Most Suitable Sense Annotation - Dijkistra (MSSA-D) are focuses on the former aspect, Flexible Lexical Chains II (FLLC II) and FXLC II target the second one.

In MSSA, we automatically disambiguate and annotate any text corpus using a sliding context window for each word. We explore how single vector representation limitations can be mitigated if MSSA is applied using a traditional word2vec implementation, producing more robust multi-sense embeddings with minimum hyperparameters tuning. Similarly, MSSA-NR works in a recurrent manner, using pre-trained synset embeddings models in the disambiguation process, instead of a traditional word-based one. While MSSA and MSSA-NR work locally in their sliding context window, MSSA-D annotates its word-senses in a global perspective. MSSA-D produces its word-senses with the objective of minimizing the cost (i.e cosine distance) of moving from the first word to the last one in the text document.

For FLLC II, our lexical chains are built with the assistance of a lexical database (i.e. WordNet) to extract the relations among the constituent synsets of a document. Additionally, FLLC II is able to handle any POS and considers 19 synset attributes from WordNet. In FXLC II, we pre-define a specific number of synsets for each lexical chain. The defined semantic space guarantees the dimensional reduction of our document representation with respect to the number of its synsets. In other words, while FLLC II offers a dynamic approach in capturing the different ideas in a text, FXLC II assumes a more strict methodology and enforces a limit in the number of synsets for each chain, representing text documents more efficiently.

Considering the word similarity task, we perform an extensive comparison with many recent publications and categorize their results according to standard metrics. During the bibliographic review, some shortcomings with respect to the experiments setup, particularly considering the metrics available, are evident in recent contributions in the area. Most publications focus on one or two metrics at the same time, ignoring the others. Hence, it is hard to confirm the superiority of one system over another. However, we try to mitigate this, comparing systems with the greatest number of similarities we can find.

The combination between the proposed MSSA algorithms and word2vec achieves solid results in 6 different benchmarks: RG65, MEN, WordSim353, SimLex999, MC28, and SCWS. In our recurrent model (MSSA-NR), we explore how we can build and improve the produced synset embeddings model iteratively. Word similarity is a downstream task and somewhat independent of whether the produced word-sense embeddings converge or not. Therefore, we believe specific experiments to study the values of our embeddings are still necessary to fully understand their behavior. The other group of our models, using global (MSSA-D) and local (MSSA) context information are also used to build synset embeddings. The former approach finds the most similar word-senses from the first to the last word in a document, while the second approach looks for the most suitable synset given a defined sliding window. Initially, we thought that MSSA-D would produce the best results on average, since it considers the whole document globally. However, if we analyze the results of WD18 only, this is not entirely true. Most of our experiments shows that MSSA obtains better results when compared to MSSA-D. Apparently, features of the local context window are more powerful than those globally obtained. We also confirm that a dimensional increase in the synset embeddings training model proves more effective than a change in the approach itself, but at the cost of some extra computation time. The simplicity of our model makes its use attractive, while the independent components of its architecture allows its extension to other NLP tasks beyond word similarity.

Currently, all MSSA techniques consider a sliding context window of +/- 1 tokens, unigrams or non-stemmed words, but we intend to pursue some extensions, such as keeping common n-grams, having a flexible context sliding window size for MSSA, and different weighting schemes for the context analysis. We will pursue a weighting scheme whereby context words closer to the target word-sense have more importance. In addition, we plan to evaluate higher levels of our recurrent model (MSSA-NR). This seems to be a more certain path to follow than just increasing the dimensionality for each scenario. We also want to explore new alternatives to build semantic representations using MSSA as their base. Finally, we also would like to integrate MSSA with the best-ranked systems evaluated throughout our experiments in the word similarity task.

Since all proposed MSSA techniques are performed on the raw text directly and prior to any word embeddings training model, they can easily be incorporated into any NLP pipeline, independent from the problem or task. This opens new alternatives to different real-world problems and systems that make use of natural language text as their input. In particular, some would directly benefit from a better semantic representation, especially in the expert systems arena. In monitoring social media activity, one could use MSSA to improve the quality of the processed comments about a certain company or product and evaluate its digital reputation (e.g. customer surveys). In chat-bots, MSSA can help intelligent systems to comprehend human textual interaction, leading us to a more human-like perception in general services (e.g. tutoring systems, automated health system, technical support). In the same direction, virtual digital assistants that are able to differentiate the nuances in a human discourse can definitely provide a better service with respect to the need and characteristics required from its users. Another interesting option, closer to academia, would be to explore the semantic signature between authors in scientific papers and Principal Investigators (PI) in research grants. The correlation between scientific papers and research grants awarded would help us to identify more relevant features that lead authors to a high productivity in their careers. In the recommender systems field, more specifically in scientific paper recommendation, one can use our semantic annotation to explore characteristics in articles other than title, abstract, and keywords only. The inner sections of a paper (e.g. introduction, related work, conclusions) are rich in content and would much benefit from our algorithms. In general, any system that requires more semantic features in order to support the decision making process can benefit from the proposed techniques in this work.

The proposed lexical chains techniques are compared against a traditional BOW approach and an additional six state-of-the-art techniques for the document classification task: LDA, word2vec, PV-DBOW, GloVe, fastText, USE, and ELMo. In order to explore the stability of our systems, we evaluate all techniques on six distinct datasets with specific characteristics, which impose a particular challenge on each classification. Furthermore, we consider five machine learning classifiers in our experiments so we can guarantee our findings are not bound to one specific classification method.

The document classification task shows that the proposed techniques for building lexical chains leverage the semantic representation offered in word embeddings. As Table 4.12 demonstrates, FLLC II and FXLC II also often improve the results of MSSA and traditional word2vec, which are used as a building block for both of them. As we use MSSA in the document classification problem, we believe the proposed lexical chains can also be transferred to other NLP downstream tasks that also require a refined semantic representation, such as sentiment analysis, text summarization, and plagiarism detection.

An interesting aspect of our architecture (Figure 3.4) is that all components can use a synset embeddings model in their representation process. That being said, once a synset embeddings model is produced, using the results from FLLC II or FXLC II, we can feed it back to our algorithms and create a more refined output. In this scenario, both techniques (FLLC II and FXLC II) will have their chain representation affected at some level. This is because the values provided using the new synset embeddings models might lead to a different synset closer to the average chain vector (get_best_rep) calculated (i.e centroid). The idea is to apply the recurrent aspect of MSSA-NR to the lexical chains approaches. Currently, we use MSSA-NR results to build our chains, but we do not explore the reuse of the produced synset chains embeddings models over multiple iterations. We believe after many passes, the representation of our chains will get more accurate, better defined, and stable.

Considering that the FLLC II and FXLC II outputs (i.e. synset chain corpus) follow an identical format for their token identification, we can also use them recurrently. If the produced lexical chains corpus is used as input, two distinct behaviors are expected. For the FLLC II algorithm, since the lexical chains are built using synset relationships from the lexical database, their structure will not change. The reason behind this is because once the flexible chains are represented, there is no lexical relationship connecting two separate chains, otherwise they would be placed together to begin with. However, in the FXLC II algorithm, for each iteration we use the output chain synset corpus as an input, the shape and the representation of these chains will drastically change. This behavior is expected if we consider that every time FXLC II is executed, the input corpus is reduced according to chunk size adopted. In case FXLC II is performed enough times, one might even end up with a single synset representing the entire corpus. Needless to say, by doing so, it is also expected that on each iteration, the semantic representation for the FXLC II technique produces a more a general output. The input-output recurrence, as for the synset model embeddings, give us interesting directions to explore in future work.

In this work, we chose to embed our MSSA and lexical chains algorithms using a word2vec implementation. However, as explained in Chapter IV, recently published embeddings techniques (e.g. fastText, USE, ELMo) bring new perspectives into play. We believe our proposed system of combining WSD, lexical chains and embeddings algorithms can leverage the semantic features in these neural network models. We intend to investigate how new bidirectional transformer language models (biLM) can be used in conjunction with our proposed techniques for lexical chains. In particular, three actors show promising findings in the NLP arena that call for our attention, namely ELMo [131], Bidirectional Encoder Representations from Transformers (BERT) [38], and Multi-Task Deep Neural Network (MT-DNN) [96].

5.1 Future Directions

Planning the next steps of this work, a preliminary study on the field of scientific paper mining and recommendation is provided in this section. The idea is to use this material as a starting point and find real world applications in which the developed techniques can be applied.

5.1.1 Scientific Paper Mining and Recommendation

As explained in Chapter I, data production is increasing in a daily basis. In 2025, the forecast is already of 175 zettabytes and it will probably increase in the next couple of years. In academia, this scenario also follows a similar trend. ELSEVIER[1] uploads approximately 420,000 records per year, with more than 130 million publications and 30,000 e-books in its archive. The Institute of Electrical and Electronics Engineers (IEEE)*Xplorer*[2] uploads, every month, around 20,000 new publications and its archive holds more than 4 million

[1]https://www.elsevier.com

[2]http://ieeexplore.ieee.org/

records among journals, conference proceedings and e-books. With more than 100,000 members, the Association for Computing Machinery (ACM)[3] is the largest scientific and educational computing society, with more than 50 journals and 170 conferences. All these groups contribute to the *information explosion*, which makes it harder to stay up-to-date with the most recent advances in any area of science. The task of reading a paper and associating it with similar ones is a tedious and time-consuming exercise, one that every scholar must go through in order to become familiar with a new field, or to stay on top of recent discoveries. This challenge becomes more obvious when the investigated field of research has many publications or the state-of-the-art of the field is yet not well-formed.

Nonetheless, there are information systems (e.g. Mendeley[4], Scopus[5]) that try to facilitate this task, suggesting similar articles based on user preferences and history. These systems make use of several different approaches to relate similar records considering *explicit features*, such as authors and co-authors, publication year and keyword list, title and keywords, abstract and title, and many other combinations. All these features can be directly extracted from publications, or their metadata (e.g. Research Information Systems (RIS)), using techniques such as string-matching, BOW and tf-idf. In fact, tf-idf is used in approximately 70% of the current weighting schemes in scientific paper recommendation systems [9]. Others explore the likelihood of pairs of terms co-occurring together in a corpus or some variation of n-grams (e.g. uni-gram, bi-gram, tri-gram) to group similar articles based on their words. Some explore these aspects through machine learning and/or probabilistic methods to infer related records. Simply calculating the probability of any given pair of words to occur, or expanding the synonyms in both corpus and query, does not extract the meaning behind the content embedded in a document. More specifically, string-based techniques often fail to obtain implicit semantic characteristics from documents. Our techniques try to provide semantic features that can help to relate the dominant interpretation

[3]http://www.acm.org
[4]https://www.mendeley.com
[5]https://www.elsevier.com/solutions/scopus

between many scientific documents. In addition, these features can be used to improve the quality of how to measure scientific production, increasing the number of co-related characteristics in the data mining process.

Given the necessity to explore non-obvious data relationships, one should be aware that, (i) words can have different meanings, even though they are written the same way; (ii) context is important, and one word can affect another at the semantic level, and (iii) sometimes an idea is represented through a series of word-events that work as a concrete semantic representation of the entire intentional concept. In other words, the whole is much more than the sum of the parts, which in this case is represented through consecutive words. The approaches presented in Chapter III extract semantic features of natural language text considering all three points, (i), (ii), and (iii). Thus, providing the perfect scenario to apply our proposed techniques.

The concern of properly measuring science is not a XXI century problem. This is first discussed in the mid 1950's by Garfield [55]. The task of measuring science, or *Scientometrics*, has its origins in the translation of the term "*Naukometrija*"[119]. Most researchers associate the modern use of *Scientometrics* to Price [135], but as Garfield [56] states, referencing his own work from 1955: "*Let me remind you of some historical facts. Price's "Science Since Babylon" (Price, 1961) was published 6 years after my 1955 paper in Science (Garfield, 1955)([55]). The first edition of Little Science, Big Science appeared 2 years later in 1963. The opening page is called a "prologue to a science of science". If Derek was aware of my paper, he did not cite it then.*". In any case, the problem of measuring scientific publications through its many features (e.g. number of citations, number of publications, authors) brings several questions into play, especially about the impact of those publications in the scientific community. Among the many questions present in the field, some relate to topics presented in this work: Is it possible to better analyze and recommend scientific papers based on its semantic features? Is it possible use semantic features to identify hidden patterns in this context? We believe that the answer for these questions is *yes*.

Bollacker et al. [18] propose the field of scientific paper recommender systems with the CiteSeer Project. This is a branch inside Recommender Systems (RSS), which can be defined as a collection of applications and algorithms designed to assist users in the process of decision making, offering support or suggestions to them [142]. These systems have a significant impact in industry and in our lives, such as Amazon[6], Google[7], Netflix[8], and Spotify[9]. Such systems are so profitable that Netflix, in 2009, granted a prize of one millon US dollars to anyone who could improve its recommender system 10%.

The problem of recommending products can be easily applied to other areas. For example, in suggesting scientific papers to scholars, articles are the *products* to be recommended to consumers (i.e. *users*). Bollacker et al. [18] make this clear when they point to the accelerated growth in the number of published research items. The difficulty of selecting relevant publications is only getting harder and requires the assistance of robust techniques. More specifically, they say that "*A problem in the search for current relevant published research is the exponential growth of the literature. The Web makes literature easier to access, but ease of publication encourages an increased publication rate. Additionally, Web based research publications tend to be poorly organized (each institution or researcher may have his or her own organizational scheme), and are spread throughout the Web.*". This just illustrates the accelerated data production issue (Chapter I), and how important it is to develop techniques that can handle such matters in a non-superficial approach.

5.1.2 Related Work in Scientific Paper Mining and Recommendation

Since the first paper on Scientific Paper Mining and Recommendation (SPR) systems [18], the number of articles in this field has grown abruptly. In 2013, Beel et al. [10] release a quantitative literature survey in which 176 research articles about scientific

[6]https://www.amazon.com/
[7]https://www.google.com/
[8]https://www.netflix.com/
[9]https://www.spotify.com/

paper recommendation are evaluated. According to Beel et al. [10], 19% of these articles are not compared against a formal baseline. Their analysis includes several perspectives, such as: system quality (e.g. accuracy, user satisfaction), evaluation methods, baselines, datasets, and etc. Their conclusions show that there are no consensus on how to evaluate and compare scientific paper recommender approaches.

Later, Beel et al. [9] improve their survey from 2013 with a more detailed study. In their updated article, 216 papers are evaluated, showing that 55% of the recommendation approaches apply content-based filtering; 18% use collaborative filtering, and 16% graph-based recommendation. The remaining is distributed among stereotyping, item-centric recommendations, and hybrid recommendations. According to their survey, many SPR systems never leave their *research project* status. Considering findings from 24 recommender systems [9], 8 are still in the prototype stage, 4 offline, 5 are not maintained, and 7 are in used in production. Along with their survey, they also show the most productive authors in the area of scientific paper recommendation, in which C. Lee Giles[10] is number one, A. Geyer-Schulz[11] is number two, M. Hahsler[12] number three, and J. Beel[13] is number four. They again conclude that currently, there is no consensus about what is the most effective and promising approach used for SPR. The lack of details in the algorithms, evaluation criteria, dataset variation, poor experimental design, and neglected quality measures, are some of the possible aspects that contribute to this outcome. Beel and Dinesh [8] discuss their team's experiences, over the six years preceding 2017, regarding their challenges in building, operating and researching three scientific article recommender systems in the digital libraries and reference management context. Their report not only shows their accomplishments, but also the reasons why several of their experiments failed according to their expectations.

In [11], they investigate the problems of choice overload associated with the difficulty

[10]http://clgiles.ist.psu.edu/index.shtml
[11]https://em.iism.kit.edu/team/show_person.php?id=7
[12]http://michael.hahsler.net
[13]https://www.tcd.ie/research/profiles/?profile=beelj

of decision making for scientific paper recommendation in digital libraries. They conduct empirical experiments with the GESIS's digital library Sowiport[14], in which they conclude that the average number of click recommendations increase with the number of items available, but not proportionally. In other words, users might feel overloaded by the number of choices rather quickly. Boratto et al. [19] discuss the current limits in the area of content-based recommender systems and propose an architecture to build semantics-aware content-based recommendation systems.

Paraschiv et al. [126] create and apply a semantic annotation model on a dataset composed of 519 proposal abstracts in an attempt to provide semantic relevant documents as results for information retrieval systems. Their methodology uses social network analysis metrics to compare the generated ranking through two models, one based semantic similarity and the other on co-authorship networks. In [84], they implement and compare two content-based techniques (tf-idf and word embeddings) for paper-to-paper recommendation using PUBMED[15] abstracts between 2010 and 2015. According to their results, the approach using word embeddings is the most accurate in comparison with a more traditional tf-idf approach.

Exploring semantic and bibliometric elements, Rollins et al. [144] propose *Manuscript Matcher* to suggest scientific papers considering the user's title, abstract and citations to perform its recommendations. In their satisfaction survey, with 2,800 users, *Manuscript Matcher* achieves a 65% overall positive evaluation. In an attempt to outperform recommendation systems of articles based on keywords, Achakulvisut et al. [2] develop a tool called *Science Concierge*[16], to recommend papers based on their content. This system takes into consideration the user's vote for relevant and irrelevant documents. They use a corpus of 15,000 posters from the Society for Neuroscience Conference[17] 2015, which

[14]http://sowiport.gesis.org/
[15]https://www.ncbi.nlm.nih.gov/pubmed/
[16]https://github.com/titipata/science_concierge
[17]https://www.sfn.org/annual-meeting/neuroscience-2015

they claim to process within $10ms$. *ArnetMiner*[18] [168], extracts and mines academic information for the user using a probabilistic approach to tackle the name ambiguity problem. Their system creates semantic-based profiles of researchers, integrates academic data from different sources, analyzes, and discovers patterns from this researcher social network. Currently, their system keeps track of more than 127 million researchers, summing up approximately 232 million publications. Other recommender systems (e.g. Mr. Dlib[19], Bibtip[20], Mendeley[21], Docear[22]) are also discussed in [9].

Xia et al. [182] propose a recommendation method which incorporates information on common author relations among articles. Their main motivation is that researchers often search published items from the same author. They use two features: the ratio of the total number of pairwise articles with common author relations to the total number of all pairwise articles for a researcher, and the ratio of the number of occurrences of the most frequent author in articles to the total number of articles for a researcher. Using this approach, they obtain better results than a traditional random walk.

The area of SPR presents a very complex scenario, in which multiple fields (e.g. NLP, machine learning, scientometrics, statistics) work together to achieve the same goal, better recommendation of scientific papers. According to Beel et al. [9], most of these approaches fall in one of the following categories: stereotyping, content-based filtering, collaborative filtering, graph-based, global relevance and hybrid. Currently, there is no consensus about which are the most used, effective, and promising approaches in the SPR field. This situation happens, mainly, because of poor experimental design, with respect to the techniques, datasets, and evaluations used [9]. We believe that the techniques proposed in this work (Chapter III) have much to contribute in extracting semantic features, through word embeddings, word-sense disambiguation, and lexical chains, to be applied to the

[18]http://www.arnetminer.org
[19]http://mr-dlib.org/
[20]http://www.bibtip.com/en
[21]https://www.mendeley.com/
[22]http://www.docear.org/

scientific paper recommendation problem. Since our systems are carefully evaluated, the next step of our research is to move forward to real-world problems and applications.

CHAPTER VI

Early Findings and Contributions

This chapter presents the early versions and experiments of the proposed techniques in Chapter III, and their respective related work.

6.1 Word Sense Disambiguation Techniques

This section presents more information about WSD techniques that do not rely on word embeddings to measure the similarity or dissimilarity between two words. Meng et al. [108] provide more details for the word similarity measures in these sections.

6.1.1 Path-Based Measures: Wu & Palmer

Path-based approaches are represented through functions that describe how two concepts are related (path and position) in the lexical database. Wu and Palmer [181]'s similarity measure is calculated considering the positions of synsets c_i and c_j in the lexical database in relation to the position of the lowest common subsumer between them, as Equation (6.1) shows.

$$sim_{WP}(c_i, c_j) = \frac{2 * depth(lcs(c_i, c_j))}{len(c_i, c_j) + 2 * depth(lcs(c_i, c_j))} \tag{6.1}$$

where c_i and c_j represent the synsets for the words w_i and w_j; $len(c_i, c_j)$ is the length of the shortest path from synset c_i to synset c_j in the lexical structure; $lcs(c_i, c_j)$ is the lowest common subsumer (e.g. hypernym) of synset c_i and synset c_j; and $depth(c_k)$ is the length of the path to synset c_k from the root entity (i.e. initial synset).

6.1.2 Information Content-Based Measures: Jiang & Conrath

IC approaches assume that every synset incorporates some amount of information. Similarity scores in this category are calculated taking into account the IC for each synset, in which similar synsets will share more common information than different ones. Jiang and Conrath [77] use the semantic distance between two synsets c_i and c_j to obtain their semantic similarity, as Equation (6.2) illustrates.

$$dis_{JC}(c_i, c_j) = (IC(c_i) + IC(c_j)) - 2 * IC(lcs(c_i, c_j)) \tag{6.2}$$

where c_i and c_j represent the synsets for words w_i and w_j; $lcs(c_i, c_j)$ is the lowest common subsumer (e.g. hypernym) of synset c_i and synset c_j; and $IC(c_k)$ is the information content calculated for c_k. In this measure, the similarity score is the opposite of the semantic distance, calculated in Equation (6.2). The IC for every synset can be either estimated manually or obtained via pre-computed index[1] [108].

6.1.3 Feature-Based Measures: Tversky

Feature-based measures do not specifically rely on the lexical database nor the subsumers of their synsets. Instead, they use properties from the lexical structure to calculate their similarity scores, such as glosses in WordNet. These measures assume that the more common features two synsets have, and the less uncommon ones they have, the more similar these synsets are. Tversky [177] suggests that semantic similarity is not reciprocal, and thus, that features between subclasses (e.g hyponyms) and superclasses (e.g. hypernyms) should have different contributions to the similarity evaluation, as Equation (6.3) shows.

[1]http://wn-similarity.sourceforge.net/

$$sim_{Tversky}(c_i, c_j)\frac{|c_i \cap c_j|}{|c_i \cap c_j| + k * |c_i/c_j| + (k-1) * |c_j/c_k|} \tag{6.3}$$

where c_i and c_j represent the synsets for words w_i and w_j; k is a constant adjustable and $k \in [0, 1]$.

6.1.4 Hybrid Measures: Zhou

Hybrid measures combine more than one approach, such as those presented so far. Similarity scores in this category can use IC with feature-based measures, IC with path-based measures, and other combinations as well. Some use more than two approaches, such as [41], which incorporates synsets, neighborhood, and semantic features. Zhou et al. [190] calculate their similarity measure based on multiple weighting factors, path characteristics, and IC, as Equation (6.4) shows.

$$\begin{aligned} sim_{Zhou}(c_i, c_j) = 1 - k * (\frac{log(len(c_i, c_j)}{log(2 * (deep_{max} - 1))}) - (1 - k) * ((IC(c_i) + \\ (IC(c_j) - 2 * IC(lcs(c_i, c_j))/2) \end{aligned} \tag{6.4}$$

where c_i and c_j represent the synsets for words w_i and w_j; k is a constant adjustable and $k \in [0, 1]$; $IC(c_p)$ is the information content calculated for c_p; $lcs(c_i, c_j)$ is the lowest common subsumer (e.g. hypernym) of synset c_i and synset c_j; $len(c_i, c_j)$ is the length of the shortest path from synset c_i to synset c_j in the lexical structure; and $deep_{max}$ is the distance of the deepest synset in the structure.

In the early versions of the algorithms in Chapter III, the aforementioned similarity measures and others [92, 88, 141, 93] were carefully examined. The description and experiments for the initial prototypes of our algorithms are presented in Sections 6.2 and 6.3 respectively.

6.2 Best Synset Disambiguation and Lexical Chains Algorithms

Using a combination of diverse types of lexical chains, we develop a text document representation that can be used for semantic document retrieval. Four different semantic representation models are proposed: (i) Best SynsetID (BSID), (ii) Flexible Lexical Chains (FLC), (iii) Flexible to Fixed Lexical Chains (F2F) and (iv) Fixed Lexical Chains (FXLC). For now, let us define a *synset* of a word as a set of synonyms for that word, and a *hypernym* of a word as a set of more general synonyms for that word [48].

In (i) we propose an extension of WSD techniques, in which we extract the best semantic representation of a word, considering the influence of its immediately neighboring words. The motivation for this technique is to prevent that words end up with an inadequate representation, given its multiple synonyms and the effects of its neighbors.

The second type (ii), uses the previous representation to build variable-sized lexical chains that delineate all concepts in a document. Though the algorithm has its complexities, the underlying idea behind the algorithm is quite simple. Proceeding linearly through the text, we convert each successive word to its semantic representation using (i). In parallel, as long as succeeding synsets share some semantic similarity, they will be part of the same set (chain), otherwise a new one must be created to capture a new idea. To illustrate the FLC algorithm, consider the sentence "*the dog and the cat run with the child and her mom in the park this Summer*" as an example. After cleaning the data and applying the Best Synset Disambiguation (BSD) algorithm, we only keep the following list {*dog, cat, child, mom, park, summer*}. The chain starts with BSID(*dog*) as the first element and ID representing the chain under construction (current chain), and evaluates the BSID(*cat*), which has the hypernym *carnivore* in common, so BSID(*cat*) is incorporated into the current chain and BSID(*carnivore*) is set as the ID for the current chain. Next, the ID representing the chain being constructed (BSID(*carnivore*)) is evaluated with the next BSID(*child*), which has the hypernym *organism* in common. The BSID(*child*) is incorporated into the current chain and BSID(*organism*) is set as the new ID for this chain. Next, since BSID(*mom*) has a hypernym in common (*organism*) with the current chain, BSID(*mom*) is also incorporated and the ID representing the chain under construction remains unchanged. The BSID(*park*) and BSID(*summer*) are not incorporated to the chain, as they do not share any common hypernym other than WN's root itself (i.e. *entity*). They also do not have any hypernym in common among themselves, forcing them to have their own single-synset-chain, resulting in the following structure {{*dog, cat, child, mom*}, {*park*}, {*summer*}}, where {*organism*}, {*park*} and {*summer*} represent each flexible chain, respectively.

In (iii), we develop an algorithm to transform FLC (ii) into a fixed size structure chain. We want to mitigate the problem of two or more long flexible chains being separated by single-synset-chain occurrences. All flexible chains in this step have an ID (FLCID) that is assigned to all BSIDs integrating the chain under construction. For example, let us consider the flexible chains {{*dog, cat, puppy*}, {*park*}, {*summer*}, {*dog, cat, puppy*}} represented using the synset IDs {{*animal*}, {*park*}, {*summer*}, {*animal*}}. These IDs are propagated to the BSIDs of the original chain, resulting in the following chain {{*animal, animal, animal*}, {*park*}, {*summer*}, {{*animal, animal, animal*}}, which will be processed into fixed structures. For this task, we divide the BSIDs, represented by FLCIDs, in sets of four units. Considering our example, the new fixed chains have the following structure {{*animal, animal, animal, park*}, {*summer, animal, animal, animal*}}. Both, the first and second chains, have the synset *animal* as the dominant interpretation, therefore the IDs for these fixed chains are readjusted to {{*animal*}, {*animal*}}. In our experiments (Section 6.3), the set size of four provides the most diverse set of chains.

Finally, in (iv), we investigate how fixed lexical structures can be derived directly from a document's semantic representation (i). In this algorithm we divide the BSIDs, for every document, in chunks of size n (c_n), and evaluate what is the synset that best represents each one of these chunks. As in the previous approach (iii), the size of four synsets is chosen, so both techniques can be better compared. For each chain c_n, we extract all hypernyms (including the initial synsets) from all the BSIDs in each chunk and select the dominant synset to represent the entire chain. If there is no dominant BSID, we select the deepest one in the chain using the root of WordNet as the start point. In case there are more than one, one synset us selected randomly, since all of them could represent the given chain. It is important to mention that, hypernyms beyond a certain threshold are not considered in our approach. The closer the root we get, the more common our synsets become, contributing poorly to the semantic diversity of our chains. Therefore, hypernyms with depth below five [71] are discarded.

All these approaches are used to construct high-dimensional vectors corresponding to the document's semantic structure, which are compared with traditional techniques, such as BOW and tf-idf. In the next sections, all cases related to the proposed algorithms will be discussed in detail.

6.2.1 Best Synset Disambiguation Algorithm (BSD)

Prior to constructing lexical chains, we need to capture the most adequate representation for the meaning of words in a document. This is done using and extending WSD algorithms. The product of this task provides what we call a BSID, a higher level of abstraction for all the words in the document, which will be used to build our lexical chains. For this, we follow the definition of lexical chains [118]. The terms used to build our lexical chains are represented through the most suitable semantic value of a word, also known as BSID.

The semantic representation of words is obtained using a lexical database, which in our case is WordNet [48]. WordNet provides a complex structure for the words and their relationships through several different semantic hierarchies. The following, is a brief summary of definitions used in WordNet, necessary to understand our work on WSD and lexical chains:

- **Lemma** - the lowercase word found in WordNet structure. The base form of a word;
- **Synonym** - a one-to-many mapping from concepts to words;

- **Gloss** - consists of a brief definition or sentence use of a synset;
- **Synset** - a set of cognitive synonyms (one or more) of a given word that share common meaning;
- **Synset ID** - a unique ID that represents the entire synset;
- **Sense** - the elements in each synset;
- **Hypernym** - a more general abstraction of a synset, corresponding to a-kind-of relationship. A *human* is a kind of a *mammal*, so *mammal* is the hypernym;
- **Hyponym** - a more specific abstraction of a synset, the opposite of hypernyms;
- **Meronym** - constitutes a "*part of*" relationship. A *hand* is part of an *arm*;
- **Least Common Subsumer** - the most specific synset in the hypernym hierarchy which is an ancestor of given synsets
- **Root** - initial synset in WN, called *entity*.

Most publications in the lexical chains field try to build these structures considering only the words within the document, some use an auxiliary annotated corpus for learning, while others use the most common synset for each word (i.e. the first synset in WordNet for each words). Our approach considers the effects of immediate neighbors for each term w_i evaluated, using all synsets available in the structure and their hypernyms. For each word w_i, with $i = 1, 2, \ldots, n$ there are 0 or more synsets available in WN. In our approach, only nouns within WordNet are considered, so the remaining are discarded. The current version of WordNet used in this project (3.0) has approximately 117,000 synsets, divided into four major categories: 81,000 noun synsets, 13,600 verb synsets, 19,000 adjective synsets, and 3,600 adverb synsets. Since the number of nouns compose almost 70% of all information available in WordNet, we choose to work with this category of synsets.

In addition, nouns allow us to explore interesting relationships between synsets, such as: hypernyms, hyponyms, meronyms, etc.

We represent the BSID of a word w_i analyzing the effects of its predecessor (w_{i-1}) and successor (w_{i+1}), called Former SynsetID (FSID)(w_i)) and Latter SynsetID (LSID)(w_i)) respectively. FSID(w_i) and LSID(w_i) are selected based on the score obtained in all possible combinations between all available synsets of the pairs (w_i, w_{i-1}) and (w_i, w_{i+1}). The synsets for w_i with the highest similarity value in comparison with w_{i-1} and w_{i+1} will be represented as FSID(w_i) and LSID(w_i) respectively. Algorithm 5 illustrates how FSID(w_i) and LSID(w_i) are extracted.

Algorithm 5 Former SynsetID (FSID) and Latter SynsetID (LSID) extraction through WSD.

```
Require: $d = \{w_i, ..., w_n\}$ : $w_i \in$ lexical database (WordNet)
1: function EXTRACT FSID AND LSID($d$)                 ▷ for $d$ - document containing words $w_n$
2:     $FSID\_LSID\_list = \emptyset$
3:     for $i = 0$ to $n$ do
4:         Apply WSD algorithm between $(w_i, w_{i-1})$ and $(w_i, w_{i+1})$
5:         Set $(FSID(w_i), FSID(w_{i-1}))$ as the pair of synsets with the highest score, using $(w_i, w_{i-1})$
6:         Set $(LSID(w_i), LSID(w_{i+1}))$ as the pair of synsets with the highest score, using $(w_i, w_{i+1})$
7:         Add $(FSID(w_i), LSID(w_i))$ to $FSID\_LSID\_list$
8:     return $FSID\_LSID\_list$
```

The similarity value is calculated for all synsets available for each word evaluated and in the end, every word w_i holds two prospective synsets (FSID(w_i) and LSID(w_i)), which represent the synsets with the highest similarity score. These are used to produce the BSID for (w_i).

The semantic similarity score can be calculated using several different types of measures. Meng et al. [108] divide these measures into four main categories: path based, IC based, feature based and hybrid methods. Each one of these categories have several ways of calculating the similarity score (Section 6.1).

After the FSID and LSID for each word w_i is found, it is necessary to find the BSID for this given word. This algorithm mitigates that words with multiple meanings (i.e. polysemy) end up with a bad representation, thus a two-level disambiguation process is performed. In

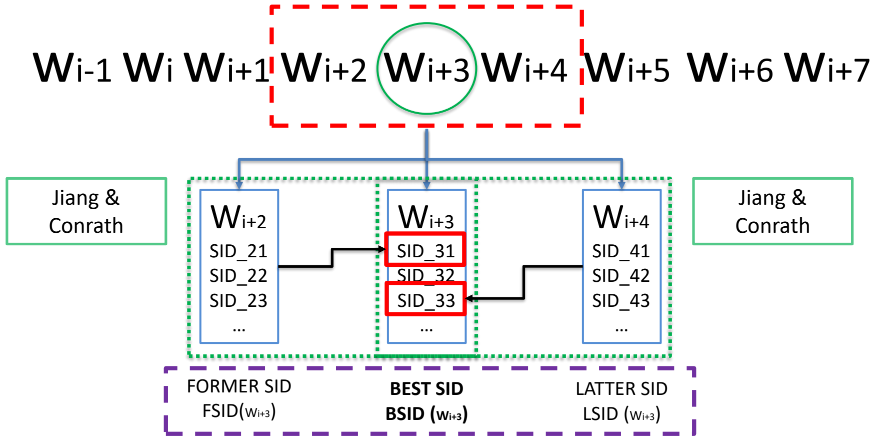


Figure 6.1: BSID through FSID and LSID evaluation process.

the first level, we apply a known WSD technique to obtain prospective pairs of synsets with the highest score considering the context of each word. In th second level, we extend the concept of WSD to synsets so the most suitable synset between FSID and LSID is chosen.

For the task of selecting a representative between FSID and LSID, we propose the BSD algorithm. Three cases are considered prior to its selection: (a) if FSID(w_i) and LSID(w_i) are equal, then BSID(w_i) = FSID(w_i) = LSID(w_i); (b) the lowest common subsumer between FSID(w_i) and LSID(w_i), not beyond a given a depth threshold; and (c), if (b) produces an empty set, the deepest synset among FSID(w_i) and LSID(w_i) is chosen. In case both have the same depth, one is chosen randomly. In (b), we use the depth of 6 (root being the initial point) as the limit to look for common hypernym extraction, which can be adjusted for more detailed or general abstractions. This value is obtained through a series of experimental tests considering factors like: execution time, diversity of synsets, diversity of chains, specificity of synsets, and others. Figure 6.1 illustrates the process of obtaining prospective candidates for BSID(w_i) (red rectangle) based on FSID(w_i) and LSID(w_i).

After these components are produced, the choice for the most suitable representation (BSID(w_i)) is initialized as shown in Algorithm 6.

As we traverse the graph in WordNet, for the lowest common subsumer extraction (b), we consider the first hypernym on each level, for each synset. Since WordNet organizes its

Algorithm 6 Best Synset Disambiguation (BSD) Algorithm.

```
Require: $FSID\_LSID\_list = \{(FSID(w_i), LSID(w_i)), ..., (FSID(w_n), LSID(w_n))\}$ : $w_i \in d$, lexical database (WordNet)
 1: function EXTRACT BEST SYNSETID($d$)                ▷ for $d$ - document containing words $w_n$
 2:     $BSID\_list = \emptyset$
 3:     for $i = 0$ to $n$ do
 4:         Apply WSD algorithm between $(w_i, w_{i-1})$ and $(w_i, w_{i+1})$
 5:         if $FSID(w_i) \equiv LSID(w_i)$ then
 6:             $BSID(w_i) = FSID(w_i) = LSID(w_i)$
 7:         else
 8:             Set hypernym cut-off for $(FSID(w_i), LSID(w_i))$
 9:             Extract the set of $HFSID(w_i)$ for all hypernyms of $FSID(w_i$
10:             Extract the set of $HLSID(w_i)$ for all hypernyms of $LSID(w_i$
11:             Let $\gamma$ = the lowest common subsumer between $HFSID(w_i)$ and $HLSID(w_i)$ using a defined
    cut-off
12:             if $\gamma$ is Not $\emptyset$ then
13:                 $BSID(w_i) = \gamma$
14:             else
15:                 depth1 = depth($HFSID(w_i)$)
16:                 depth1 = depth($HLSID(w_i)$)
17:                 if (depth1 == depth2) then
18:                     $BSID(w_i)$ = $Random(HFSID(w_i), HLSID(w_i))$
19:                 else
20:                     $BSID(w_i)$ = $argmax_{s_{c1}}\{depth(HFSID(w_i), HLSID(w_i))\}$
21:         Add $BSID(w_i)$ to $BSID\_list$
22:     return $BSID\_list$
```

synsets from most to least frequent usage, and we are generalizing the concepts as we move towards the root, it is only natural that we extract a hypernym that will provide the most diffused element with respect to its frequency in the lexical database. In other words, the first hypernym in every upper level will provide greater probability of an intersection with another synset when we build our lexical chains.

6.2.2 Flexible Lexical Chains Algorithm (FLC)

Once all words are annotated with their respective BSID, we can build our lexical chains, which are also used to extract semantic features from documents. To the best of our knowledge, this routine introduces a novel approach, constructing a parameterized flexible lexical chain, considering an adaptive structure of synsets based on multilevels of hypernyms.

We introduce an algorithm called Flexible Lexical Chains (FLC), which extracts these chains, evaluating whether a new synset of a word w_i or its hypernyms, present lexical cohesion among themselves and the current chain under construction. If the evaluated synset has semantic affinity with the chain being built, then this new synset is incorporated to the chain. Otherwise, a new chain is initialized to capture the next semantic representation.

As Algorithm 7 shows, first, we begin a new FLC inserting the first synset for word w_1, from the text into an initial chain structure, calling this first chain FLC(*NCh*), where *NCh* is equal to 1 and BSID(w_1) is the synset ID representing the FLC (FLCID(*NCh*)). For each following synset, BSID(w_i), where $i \in [2, n]$, we verify if BSID(w_i) is equal to FLCID(*NCh*). If they are the same, we just add BSID(w_i) into the current chain. In case they are not the same, we need to investigate if there is any semantic relationship shared between BSID(w_i) and FLCID(*NCh*). This is done through the extraction of all hypernyms from BSID(w_i) and FLCID(*NCh*), called α and β respectively.

Next, we choose the lowest common subsumer (hypernym), called γ, between α and β given a certain cut-off. In our approach, this value is also a parameter that can be changed accordingly. This threshold is to avoid the relatedness between α and β being too general, since all noun synsets in WordNet are connected to the root $entity$. During experimental tests, depth values below 5 or 4 resulted in non-useful chains for our purposes, confirming Hotho et al. [71] findings. If γ does not exist, it means that BSID(w_i) and FLCID(*NCh*) are not semantically related, so the current chain does not absorb the current BSID(w_i). As a result, a new FLC must be initialized so the new lexical cohesion can be captured. In case γ does exist, BSID(w_i) is included into the current FLC and the synset representing it is updated, considering the synset in γ.

Figure 6.2 illustrates how words in a text would compose a flexible lexical chain in a pictorial way. Every w_i represents the BSID(w_i) obtained using the BSD algorithm, while S_j is the FLCID for each chain.

Algorithm 7 Flexible Lexical Chain (FLC) Algorithm.

```
Require: $BSID\_list = \{BSID(w_i), ..., BSID(w_n)\} : w_i \in d$, *lexical database* (WordNet)    ▷ for $i = 1, ..., n$
 1: function EXTRACT FLEXIBLE LEXICAL CHAINS($d$)    ▷ for $d$ - document containing synsets $BSID(w_n)$
 2:     Set NCh = 1
 3:     Define FLC(NCh) as the initial *flexible lexical chain*, containing only $BSID(w_1)$
 4:     Set $BSID(w_1)$ as synset representative of $FLC(NCh)$; $FLCID(NCh)$ = $BSID(w_n)$
 5:     for $i = 2$ to $n$ do
 6:         if $BSID(w_i)$ == $FLCID(NCh)$ then
 7:             Add $BSID(w_i)$ to FLC(NCh)
 8:         else
 9:             Extract the set $\alpha$ of all hypernyms $BSID(w_i)$
10:             Extract the set $\beta$ of all hypernyms $FLCID(NCh)$
11:             if $\gamma$ = $\emptyset$ then
12:                 $NCh = NCh + 1$
13:                 Define $FLC(NCh)$ as the next *flexible lexical chains*, containing only $BSID(w_i)$ and re-initialize $FLCID(NCh)$ to $BSID(w_i)$
14:             else
15:                 Add $BSID(w_i)$ to $FLC(NCh)$
16:                 Set $FLCID(NCh)$ = $\gamma$
17:     return $FLC(j)$    ▷ for $j = \{1, ..., NCh\}$
```

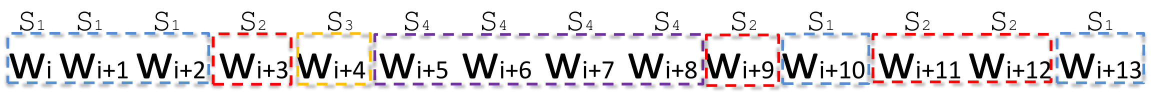

S1 S1 S1 S2 S3 S4 S4 S4 S4 S2 S1 S2 S2 S1
Wi Wi+1 Wi+2 Wi+3 Wi+4 Wi+5 Wi+6 Wi+7 Wi+8 Wi+9 Wi+10 Wi+11 Wi+12 Wi+13

Figure 6.2: Flexible Lexical Chains example construction diagram.

6.2.3 Flexible to Fixed Lexical Chains Algorithm (F2F)

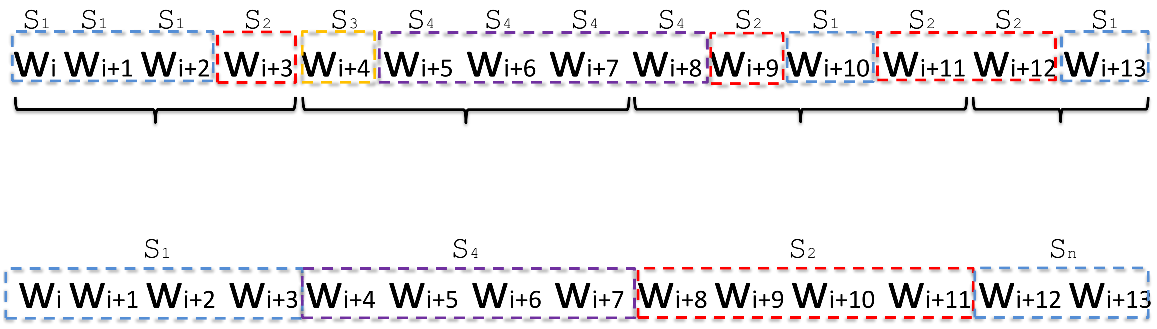
Another proposed algorithm is to produce fixed lexical chains from FLC. For this task, after all FLC are produced, we convert these flexible chains into fixed structures ones.

After pairs of words and synsets are arranged, we divide the document into chunks of size k with respect to the number of synsets, represented as c_{wk}. Each chunk corresponds to a fixed chain. For each c_{wk}, we extract the synset that appears most often in the chunk, called θ, and assign it to represent c_{wk}. If there is more than one synset or all synsets have the same frequency, θ is chosen randomly, as Algorithm 8 shows. In Figure 6.3 we provide a pictorial representation of the process, where each fixed lexical chain is represented using a single synset.

The total number of synsets used in our overall document representation is the union over all documents of the synsets representing all the fixed lexical chains in that docu-

Algorithm 8 Flexible to Fixed Lexical Chain (F2F) Algorithm.

```
Require: $FLC\_list = \{BSID(w_i), ..., BSID(w_n)\} : w_i \in d$, lexical database (WordNet)   ▷ for $i = 1, ..., n$
 1: function EXTRACT FLEXIBLE TO FIXED LEXICAL CHAINS($d$)   ▷ for $d$ - document containing synsets $BSID(w_n)$
 2:     for $i = 1$ to $n$ do
 3:         Set $synset(w_i) = p$   ▷ where $w_i$ occurs in $FLC(k)$ and $p = FLCID(k)$
 4:         Split $d$ into fixed-sized chunks of $k$ words each   ▷ $d = FLC\_list$
 5:     $F2F\_list = \emptyset$
 6:     for $cw_j = 1$ to $NChunks$ do   ▷ where $j = \{1, ..., NChunks\}$
 7:         Let $\{w_{j,1}, ..., w_{j,k}\}$ be the number of word occurrences in chunk $cw_j$
 8:         Let $\theta = \{synset(w_{j,1}), ..., synset(w_{j,k})\}$   ▷ for a given chunk $cw_j$
 9:         Represent chunk $cw_j$ by the dominant synset of $\theta$, $dominant(\theta)$
10:         if $dominant(\theta) = \emptyset$ then
11:             $dominant(\theta) = Random(\theta)$   ▷ where a random synset is selected from chunk $cw_j$ in $\theta$
12:         Add $dominant(\theta)$ to $F2F\_list$
13:     return $F2F\_list$
```



S1 S1 S1 S2 S3 S4 S4 S4 S4 S2 S1 S2 S2 S1
Wi Wi+1 Wi+2 Wi+3 Wi+4 Wi+5 Wi+6 Wi+7 Wi+8 Wi+9 Wi+10 Wi+11 Wi+12 Wi+13
S1 S4 S2 Sn
Wi Wi+1 Wi+2 Wi+3 Wi+4 Wi+5 Wi+6 Wi+7 Wi+8 Wi+9 Wi+10 Wi+11 Wi+12 Wi+13

Figure 6.3: Flexible to Fixed Lexical Chains construction diagram.

ment. Call this value $NSynsets$. Let the synsets used in our document representation be $syn_1, syn_2, \ldots, syn_{NSynsets}$.

6.2.4 Fixed Lexical Chains Algorithm (FXLC)

We also propose an algorithm called Fixed Lexical Chains (FXLC), which extracts lexical chains given a pre-defined number of chunks directly derived from the BSID annotation. Different than the previous algorithm (F2F), FXLC does not use any pre-processed lexical chains. Its construction is entirely based on BSIDs for each word w_i. We develop this technique to compare which lexical chain structure would present better results, the one derived from FLC or the one obtained directly from BSIDs.

As Algorithm 9 shows, for each chain c_d, we extract all hypernyms (including the initial synsets) from all the synsets and call this set λ. Since each chain must be properly represented, only dominant synsets (appearing in at least half of c_d), β, are considered. If there is no dominant synset in the chain, λ is selected instead. Let δ be either λ or β, depending on the preceding condition. Next, we choose a subset of δ, called ε, of those synsets that are not too close to the root ($entity$) in WordNet. This is to prevent providing a too general synset for our chains. If ε is empty, all synsets in δ are considered instead. Let Ω be either δ or ε, depending on the preceding condition.

Algorithm 9 Fixed Lexical Chain (FXLC) Algorithm.

```
Require: $BSID\_list = \{BSID(w_i), ..., BSID(w_n)\}$ : $w_i \in d$, lexical database (WordNet)    ▷ for $i = 1, ..., n$
 1: function EXTRACT FIXED LEXICAL CHAINS($d$)    ▷ for $d$ - document containing synsets $BSID(w_n)$
 2:     Split $d$ into fixed-sized chunks of $k$ words each, named $Chunks\_list$    ▷ where $Chunks\_list = \{c_i, ..., c_j\}$, with $c_i = \{BSID(w_p), ..., BSID(w_k)\}$ for $0 < i < j and 0 < p, k < i$
 3:     $FXLC\_list$ = $\emptyset$
 4:     for $d = 1$ to $t$ do
 5:         Select the set $\lambda_{c_d}$ of all hypernyms from each chain $c_d$    ▷ where $c_d \in Chunks\_list$
 6:         Select the set $\beta$ of all synsets that appear in at least half of $c_d$
 7:         if $\beta = \emptyset$ then
 8:             $\delta$ = $\lambda_{c_d}$
 9:         else
10:             $\delta$ = $\beta$
11:         Perform cut-off items in $\delta$ based on a chosen limit, producing $\varepsilon$
12:         if $\varepsilon = \emptyset$ then
13:             $\omega$ = $\delta$
14:         else
15:             $\omega$ = $\varepsilon$
16:         Extract the set $\alpha$ of all maximally occurring synsets in $\omega$
17:         Select the set $\gamma$ of maximally deepest synsets in $alpha$
18:         Add a random synset from $\gamma$ to $FXLC\_list$
19:     return $FXLC\_list$
```

Then, from Ω we extract all maximally occurring synsets, α. We then construct γ, a subset of synsets that occur at the deepest level of α. If α has more than one synset, the synset to represent the initial chain c_d is then a random synset from α. Since we already limit the search to the hypernyms, to guarantee a certain level of generality, now we want to maintain the semantic value within each fixed chain. Figure 6.4 provides a pictorial example of FXLC result.

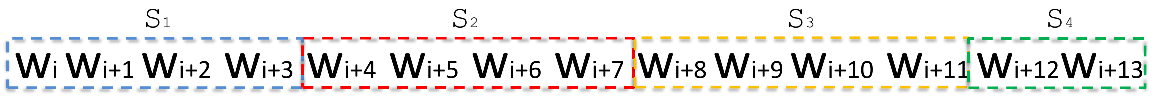


Figure 6.4: Fixed Lexical Chains construction diagram.

6.2.5 Distributed Semantic Extraction Mapping

Once all the words in the documents are properly transformed into semantic vector representations, either using BSD, FLC, FXLC or F2F, we can explore how these vectors can be used. We propose to work with distances between the produced synsets to extract the semantic signature embedded in each document. The choice for the relative distance, instead of the actual position, is justified because of its invariance independently of a document length. The distances between these obtained synsets is used to define in which bins, in a histogram representing each document, it is located.

Considering a document d. For each $1 \leq i \leq NSynsets$, we define $h(d,i)$ to be the histogram of relative distances between consecutive occurrences of syn_i in document d. Note that the number of bins of $h(d,i)$ and $h(e,j)$ are the same for any 2 documents d, e, and synsets i, j. Also, for $h(d,i)$, if syn_i does not occur in document d, then the histogram consists of all 0's. Document d is then represented as a normalized concatenation of $h(d,syn_1), h(d,syn_2), \ldots, h(d,syn_{NSynsets})$.

We note that synsets occurring once present a problem regarding their distance, so we treat them in two ways: we either ignore them or not. To make sure these issues are covered, we explore three variations considering the distances of synsets for each kind of chain: (I) ignoring single occurrences of synsets, (II) considering single occurrences of synsets as having a 0 relative distance from themselves and (III) considering single occurrences of synsets and the first synset of a chain as having a 0 relative distance from themselves. An example for each approach is shown in Tables 6.1, 6.2, and 6.3 respectively. In these examples, we use five documents divided into 4 segments (bins) for the same vector of synsets in each document. Initially, all bins are initialized with 0 and based on the distance of identical synsets (considering the given approach) we find in which bin this synset-distance belongs.

Each bin is represented as a half-closed, half open set of relative distance ranges. Bin 1 corresponds to the set [0,0.25), bin 2 to the set [0.25,0.5), bin 3 to the set [0.5,0.75), and bin 4 to the set [0.75,1). Since each distance occurring in a synset string of length n is at most $n - 1$, the largest relative distance possible is $(n - 1)/n$, which approaches to 1, as $n \to \infty$. Synsets that do not occur in a string will have 0's in all bins. In Table 6.1, synsets that only occur once are ignored, while Table 6.2 and Table 6.3 consider synsets occurring once and the first synset in a chain respectively. In a nutshell, what our approach does is to characterize the spatial distribution of synsets in a document using a histogram to keep track of those synsets using their relative distances. Needless to say, this process is normalized so longer/shorter documents are not biased.

Table 6.1: Toy Example I - Ignoring single occurrences of synsets.

Doc	Sequence of Synsets	Raw Distances	4-Bin Histogram Representation
1	$S_1S_2S_2S_4S_2S_3S_1$	$S_1 < 6 >, S_2 < 1,2 >, S_3 <>, S_4 <>$	$< 0,0,0,1 > \mid < 1,1,0,0 > \mid < 0,0,0,0 > \mid < 0,0,0,0 >$
2	$S_3S_4S_3S_2S_3$	$S_1 <>, S_2 <>, S_3 < 2,2 >, S_4 <>$	$< 0,0,0,0 > \mid < 0,0,0,0 > \mid < 0,2,0,0 > \mid < 0,0,0,0 >$
3	$S_1S_2S_1S_4S_4S_1S_3S_3S_4S_3S_2S_1$	$S_1 < 2,3,6 >, S_2 < 9 >, S_3 < 1,2 >, S_4 < 1,4 >$	$< 1,1,1,0 > \mid < 0,0,0,1 > \mid < 2,0,0,0 > \mid < 1,1,0,0 >$
4	$S_4S_3S_3$	$S_1 <>, S_2 <>, S_3 < 1 >, S_4 <>$	$< 0,0,0,0 > \mid < 0,0,0,0 > \mid < 0,1,0,0 > \mid < 0,0,0,0 >$
5	$S_4S_1S_1S_3S_1S_2S_4S_3S_2$	$S_1 < 1,2 >, S_2 < 3 >, S_3 < 4 >, S_4 < 6 >$	$< 2,0,0,0 > \mid < 0,1,0,0 > \mid < 0,1,0,0 > \mid < 0,0,1,0 >$

Table 6.2: Toy Example II - Considering single occurrences of synsets having a 0 relative distance from themselves.

Doc	Sequence of Synsets	Raw Distances	4-Bin Histogram Representation
1	$S_1S_2S_2S_4S_2S_3S_1$	$S_1 < 6 >, S_2 < 1,2 >, S_3 < 0 >, S_4 < 0 >$	$< 0,0,0,1 > \mid < 1,1,0,0 > \mid < 1,0,0,0 > \mid < 1,0,0,0 >$
2	$S_3S_4S_3S_2S_3$	$S_1 <>, S_2 < 0 >, S_3 < 1,1 >, S_4 < 0 >$	$< 0,0,0,0 > \mid < 1,0,0,0 > \mid < 0,2,0,0 > \mid < 1,0,0,0 >$
3	$S_1S_2S_1S_4S_4S_1S_3S_3S_4S_3S_2S_1$	$S_1 < 2,3,6 >, S_2 < 9 >, S_3 < 1,2 >, S_4 < 1,4 >$	$< 1,1,1,0 > \mid < 0,0,0,1 > \mid < 2,0,0,0 > \mid < 1,1,0,0 >$
4	$S_4S_3S_3$	$S_1 <>, S_2 <>, S_3 < 1 >, S_4 < 0 >$	$< 0,0,0,0 > \mid < 0,0,0,0 > \mid < 0,1,0,0 > \mid < 1,0,0,0 >$
5	$S_4S_1S_1S_3S_1S_2S_4S_3S_2$	$S_1 < 1,2 >, S_2 < 3 >, S_3 < 4 >, S_4 < 6 >$	$< 2,0,0,0 > \mid < 0,1,0,0 > \mid < 0,1,0,0 > \mid < 0,0,1,0 >$

6.2.6 Semantic Term Frequency-Inverse Document Frequency (TF-IDF)

With all chains properly represented, one could simply apply a normal count of words (e.g. BOW) to the obtained synsets and use the most frequent semantic representation.

Table 6.3: Toy Example III - Considering single occurrences of synsets and first synset of a chain having a 0 relative distance of themselves.

Doc	Sequence of Synsets	Raw Distances	4-Bin Histogram Representation
1	$S_1S_2S_2S_4S_2S_3S_1$	$S_1 < 0,6 >, S_2 < 0,1,2 >, S_3 < 0 >, S_4 < 0 >$	$< 1,0,0,1 > \mid < 2,1,0,0 > \mid < 1,0,0,0 > \mid < 1,0,0,0 >$
2	$S_3S_4S_3S_2S_3$	$S_1 <>, S_2 < 0 >, S_3 < 0,1,1 >, S_4 < 0 >$	$< 0,0,0,0 > \mid < 1,0,0,0 > \mid < 1,2,0,0 > \mid < 1,0,0,0 >$
3	$S_1S_2S_1S_4S_4S_1S_3S_3S_4S_3S_2S_1$	$S_1 < 0,2,3,6 >, S_2 < 0,9 >, S_3 < 0,1,2 >, S_4 < 0,1,4 >$	$< 2,1,1,0 > \mid < 1,0,0,1 > \mid < 3,0,0,0 > \mid < 2,1,0,0 >$
4	$S_4S_3S_3$	$S_1 <>, S_2 <>, S_3 < 0,1 >, S_4 < 0 >$	$< 0,0,0,0 > \mid < 0,0,0,0 > \mid < 1,1,0,0 > \mid < 1,0,0,0 >$
5	$S_4S_1S_1S_3S_1S_2S_4S_3S_2$	$S_1 < 0,1,2 >, S_2 < 0,3 >, S_3 < 0,4 >, S_4 < 0,6 >$	$< 3,0,0,0 > \mid < 1,1,0,0 > \mid < 1,1,0,0 > \mid < 1,0,1,0 >$

However, as for terms, a synset that appears very often in the corpus can bias the final result in a information retrieval system. To mitigate such a problem, we extend the concept of the traditional tf-idf to the synsets obtained from BSD, FLC, FXLC, and F2F algorithms. Equation (6.5) shows how the relevance of each synset is calculated [156].

$$x_{ij} = tf_{ij} * log(\frac{n}{n_j}) \tag{6.5}$$

where X is our corpus, x_{ij} is the weight of feature j (synset) in document i, tf_{ij} is the synset frequency of j in i; n is the number of documents in X, and n_j is the number of documents in X that contain the synset j. The $log(\frac{n}{n_j})$ term represents the inverse document frequency part in tf-idf. Whissell and Clarke [180] discuss several alternatives (e.g. Okapi BM25) that can be applied to Equation (6.5) for normalizing tf-idf.

6.3 Proof-Of-Concept Experiments

To assess the proposed techniques in Section 6.2, we create a corpus of 30 distinct documents from Wikipedia[2] as a Proof-of-Concept (POC). These are distributed equally in three major categories: dogs, computers and sports. The html files of these pages are saved and parsed, so common English stopwords (e.g. a, an, the) are removed. Although this seems, ostensibly, a modest dataset, the amount of information extracted from these documents is quite large. The combined documents present a total of 216,514 words in

[2]https://www.wikipedia.org/

their html files, of which 68,836 are mapped into WordNet. For this dataset, 32.62% of the words have at least one synset match in WordNet, which contains multiple senses. Table 6.4 shows the details of our dataset.

Table 6.4: Wikipedia Dataset Details.

Wikipedia Category	**Number of Documents**	**Number of Words**	**Nouns Matched in WN**	**Avg. of Nouns in WN (%)**
Dogs	10	48,650	16,239	34.37
Computers	10	79,332	24,331	31.11
Sports	10	88,532	28,266	32.38
Total	**30**	**216,514**	**68,836**	**32.62**

Two experimental categories are developed in order test our proposed algorithms (Section 6.2). The first concerns how semantic extraction via lexical chains can be used in Information Retrieval (IR). The second explores how semantic features, such as key-concepts, can be suggested as keywords for documents based on their meaning. Both experiments do not rely on human supervision, but instead, make use of the semantic content of each document in the corpus.

6.3.1 Semantic Extraction Based on Lexical Chains

As explained in Algorithm 5, during the pre-processing step, we only keep the nouns for each document that a synset match in WordNet. After pre-processing the data (e.g. lowercase, stopwords removal), our corpus has a total of approximately 216K words, of which 68K (nouns) have a match in WN, as Table 6.4 shows.

For our synset experiments, the number of synsets in our term/document matrix ranges between 1284 and 7490 elements. In addition, the documents considered in this experiments have, on average, 7,200 words each, which can produce a considerable large dataset to process.

After all datasets are properly cleaned, we extract the BSID representation (Section 6.2.1), which is used as a base for all our lexical chains scenarios (FLC, FXLC, and F2F). Once all flexible lexical chains are extracted from the documents, they are used to map fixed lexical chain structures, and to create the corresponding semantic vector representations. We also derive FXLC directly from the BSID vectors, using a fixed chain size, as shown in Section 6.2.4.

In our experiments, we validate our various approaches in a clustering task, using 256 bins for our synset-based techniques. As mentioned previously, we have documents from 3 major categories, so we perform a variant of K-Means clustering for $K = 3$ and evaluate the resulting clusters using both the *Adjusted Rand Index* and the *Mean Individual Silhouette* values. The former metric is a measure of similarity between two clusters. We compare the derived clusters to the ground truth clusters, consisting of all the *dog* documents, all the *computer* documents, and all the *sport* documents. The latter metric sees how well the clusters are designed, determining whether documents in the same cluster are close together, while documents in different clusters are far apart.

We use spherical K-Means clustering [70], as this technique considers *cosine distance* [63, 154] rather than Euclidean distance. To validate the proposed algorithms, we also design, implement, and extend traditional approaches for document similarity, such as: BOW with all words (except common stopwords) in the documents (Bag-of-Words-Raw (BOWR)), BOW with only matched nouns in WordNet (Bag-of-Words-WordNet (BOWN)), BOW with the first synset match (commonly used) in WordNet (Bag-of-Words-Synsets (BOWS)), and BOW with the BSID (Bag-of-Words-Best (BOWB)) extracted from the BSD. Since the traditional approaches are variation of counts, only one bin is considered for these histograms.

Table 6.5 provides a summary of all experiments performed, while Figure 6.5 shows a scatter plot of these results. These results show that various permutations of our general approach worked better than traditional ones, of which four of our approaches stand out. Considering the results presented in Figure 6.5, some observations can be made:

- Three out of the four results with perfect clustering are from our techniques. Two

Table 6.5: Experiments using lexical chains algorithms and traditional approaches.

Label	**Algorithm**	**Adjusted Rand Index**	**Mean Individual Silhouette**
A	Pure Flex–Method III	1	0.1908
B	Pure Flex–Method II	1	0.1775
C	BOW-N–Nouns in Wordnet	1	0.1757
D	BOW-B–Best Synsets	1	0.1686
E	Flex-2-Fixed–Method I	0.8981704	0.3964
F	Flex-2-Fixed–Method III	0.8981704	0.3878
G	BOW-R–Raw Words	0.8981704	0.1591
H	Flex-2-Fixed–Method II	0.8066667	0.3578
I	BOW-S–WordNet First Synset	0.6671449	0.1542
J	Pure Flex–Method I	0.6590742	0.1826
K	Pure Fixed–Method I	0.6044735	0.2137
L	Pure Fixed–Method III	0.5165853	0.2734
M	Pure Fixed–Method III	0.40252	0.2743

of these perfect clusterings use FLC (considering their variations) while the third perfect clustering results from the proposed methodology of finding the best synset representation for a document (Section 6.2.1);

- The only perfect clustering result which is on the Pareto front (not dominated by another result), is the one using the third approach in Section 6.2.5 (III) for extracting flexible chains;

- The clustering with the maximum silhouette value results from our first approach in Section 6.2.5 (I) to our technique for extracting F2F chains. This clustering is also on the Pareto front; and

- The only clusterings on the Pareto front result from our techniques.

In this experiment, we explore how extracted semantic features can aid in document retrieval tasks. Furthermore, we present several contributions on how these features can be extracted to form more robust lexical chains. First, we explore and extend the notion of WSD and how to represent words, considering the effect of their immediate neighbors in their meaning BSID. Second, we propose three new algorithms: (a) a new methodology to create variable length size semantic chains (FLC), (b) an algorithm to derive fixed lexical structures

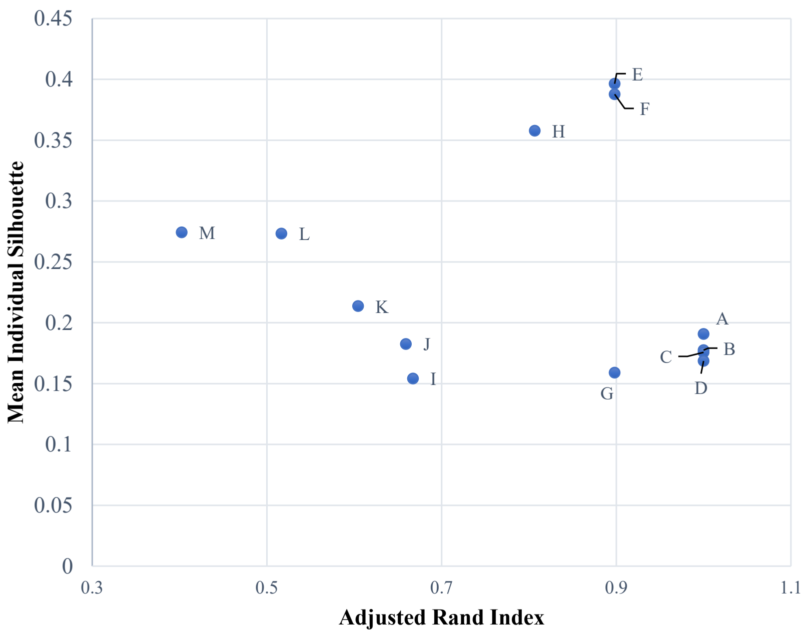


Figure 6.5: Scatter plot between mean individual silhouette and Adjusted Rand Index for proposed techniques (Table 6.5 data).

(FXLC) directly from semantic representations and (c) a new approach to transform variable length size semantic chains into fixed parameterized structures (F2F). Third, we provide different alternatives to construct the semantic dispersion over a document (Section 6.2.5). To establish a comparison with the proposed approaches we compare them with traditional ones, such as BOW and a few of its variations. Our findings show that several of our approaches achieve superior results.

6.3.2 Semantic Extraction - Reuters-21578 Benchmarking

In this set of experiments we want to verify if the use of lexical chains, based on our semantic representation, provides good results in cluster evaluation for a well-known dataset, named Reuters-21578 [90]. This dataset, has been used as a baseline for text evaluation for many years in the text classification community. Reuters Corpus Volume I (RCV1) is composed of 21578 records (train and test) distributed in 17 categories, as shown in Table 6.6

As Table 6.6 shows, the documents in each topic are not distributed uniformly, which

Table 6.6: Reuters-21578 Dataset topic distribution.

Topics	Train	Test	Other	Total
0	1828	280	8103	10211
1	6552	2581	361	9494
2	890	309	135	1334
3	191	64	55	310
4	62	32	10	104
5	39	14	8	61
6	21	6	3	30
7	7	4	0	11
8	4	2	0	6
9	4	2	0	6
10	3	1	0	4
11	0	1	1	2
12	1	1	0	2
13	0	0	0	0
14	0	2	0	2
15	0	0	0	0
16	1	0	0	1
Total	**9603**	**3299**	**8676**	**21578**

does not promote a good environment for our techniques. So, a sample corpus, called Reuters-R8[3] (R8), is used instead. This corpus has a total of 7,674 records, where 5,485 are for training and 2,189 for testing. These documents are reorganized into 8 (eight) different classes, as Table 6.7 shows. Since our approach does not rely on machine learning techniques, we only use the documents in the test division group.

Table 6.7: Sample Reuters-21578 (R8) Dataset topic distribution.

Class	Train	Test	Total
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	**5485**	**2189**	**7674**

To evaluate how well our lexical chains illustrate the given clusters, we use the Silhouette

[3]https://www.cs.umb.edu/ smimarog/textmining/datasets/

evaluation [147] through a MatLab [101] implementation. First, we apply the F2F algorithm for the 3 different scenarios discussed in Section 6.2.5, and also obtain the BOWR, BOWN, BOWS, and BOWB for R8. Second, after extracting the semantic vector representation, based on the relative distances of synsets, we calculate the similarity of all pair of documents d_i and d_j based on the cosine distance illustrated in Equation 6.6.

$$\cos(d_i, d_j) = \frac{\overrightarrow{d_i} \bullet \overrightarrow{d_j}}{\|d_i\| * \|d_j\|} \tag{6.6}$$

where d_k is a vector representation of a document k; $\bullet$ represents the dot product between documents d_i and d_j; and $\|d_k\|$ denotes the norm of a vector d_k.

The silhouette is calculated for K number of clusters, where $K \in [2, 10]$ with optimal value being the number of classes available in Table 6.7 for $Klist$ equals to 8. In addition, the clusters are calculated using three different criteria [167]:

- Complete-Linkage: The maximum distance between elements of each cluster;
- Single-Linkage: The minimum distance between elements of each cluster; and
- Average-Linkage: The mean distance between elements of each clusters.

The silhouette value ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters and a lower value otherwise. Tables 6.8, 6.9, and 6.3.2, show the results for the Silhouette values considering complete, single, and average linkage respectively. We also apply tf-idf to all F2F types (I, II, and III) for 8 and 1 bins. All tables follow the same label structure in which they are described as $F2F[T][NB]$, where $F2F$ is the algorithm applied, T represents the type of semantic mapping (Section 6.2.5) and NB the number of bins. For the BOWR, BOWN, BOWS and BOWB only tf-idf is applied. In all results, with the exception of BOWR for $K = 2$ using a Single-Linkage criteria (Table 6.9), our techniques present the best results.

Considering $K = 8$, which is the optimal number of clusters, our approaches outperform the traditional ones.

Table 6.8: Silhouette values using Complete-Linkage for R8.

Klist	BOWR	BOWN	BOWS	BOWB	F2F18B	TFIDF F2F28B	F2F38B	F2F18B	N_TFIDF F2F28B	F2F38B	F2F11B	TFIDF F2F21B	F2F31B
2	0.676151	0.686307	0.692830	0.709832	0.748315	0.68264	0.688285	0.751098	0.621004	0.663514	0.725597	0.723823	0.716519
3	0.647674	0.644507	0.638558	0.636677	0.707231	0.619227	0.653755	0.687085	0.607775	0.636916	0.675335	0.679655	0.651138
4	0.648212	0.622400	0.648059	0.565860	0.682106	0.626394	0.651444	0.640624	0.657077	0.663896	0.611431	0.585153	0.587824
5	0.581421	0.592337	0.629298	0.546278	0.620196	0.565927	0.635662	0.603084	0.616786	0.645666	0.605293	0.616073	0.604827
6	0.505711	0.555967	0.542279	0.557926	0.585092	0.532517	0.569278	0.579201	0.623257	0.627896	0.577291	0.581066	0.575648
7	0.514248	0.541449	0.529408	0.545692	0.563665	0.522126	0.579578	0.577095	0.573614	0.575313	0.590608	0.595462	0.577846
8	0.518504	0.527968	0.507091	0.564807	0.501073	0.513813	0.566298	0.520971	0.561005	0.565438	0.572945	0.584377	0.549457
9	0.510619	0.547094	0.537580	0.561090	0.483368	0.548587	0.537850	0.513657	0.536961	0.560868	0.561792	0.516203	0.536522
10	0.530594	0.538165	0.537276	0.542777	0.497052	0.544803	0.544536	0.524776	0.506360	0.582179	0.494155	0.513052	0.553762

Table 6.9: Silhouette values using Single-Linkage for R8.

Klist	BOWR	BOWN	BOWS	BOWB	F2F18B	TFIDF F2F28B	F2F38B	F2F18B	N_TFIDF F2F28B	F2F38B	F2F11B	TFIDF F2F21B	F2F31B
2	0.650568	0.602427	0.589385	0.469210	0.469210	0.518770	0.509838	0.432166	0.525504	0.433753	0.502739	0.534758	0.576733
3	0.588101	0.570857	0.527785	0.488958	0.488958	0.588379	0.621737	0.506146	0.618863	0.609221	0.580751	0.590981	0.583023
4	0.567827	0.544725	0.532246	0.537467	0.537467	0.634043	0.584709	0.496142	0.656657	0.631710	0.620261	0.610911	0.625833
5	0.515514	0.455132	0.515499	0.531870	0.531870	0.621357	0.561739	0.507846	0.644333	0.634186	0.617114	0.609967	0.589761
6	0.505577	0.504164	0.529438	0.549661	0.549661	0.552643	0.572941	0.483117	0.565215	0.508525	0.571690	0.547034	0.537042
7	0.506185	0.512772	0.509743	0.543066	0.543066	0.582054	0.531042	0.514974	0.517026	0.460308	0.582275	0.515976	0.529703
8	0.489722	0.531081	0.523103	0.511094	0.511094	0.528449	0.523812	0.502321	0.529891	0.496979	0.535289	0.520520	0.528266
9	0.468483	0.539251	0.516185	0.524173	0.524173	0.547048	0.503592	0.503695	0.513297	0.519579	0.477582	0.522089	0.490709
10	0.487594	0.544297	0.482526	0.531379	0.531379	0.558379	0.490613	0.508039	0.508070	0.531477	0.490551	0.514298	0.493155

Table 6.10: Silhouette values using Average-Linkage for R8.

Klist	BOWR	BOWN	BOWS	BOWB	F2F18B	TFIDF F2F28B	F2F38B	F2F18B	N_TFIDF F2F28B	F2F38B	F2F11B	TFIDF F2F21B	F2F31B
2	0.598647	0.661578	0.661315	0.647147	0.715684	0.627734	0.649585	0.706731	0.629435	0.636651	0.735324	0.732316	0.723169
3	0.611935	0.653283	0.605236	0.586738	0.609604	0.589331	0.601001	0.640710	0.599209	0.627902	0.683099	0.662566	0.674607
4	0.560659	0.640616	0.563835	0.536996	0.542805	0.621032	0.471991	0.584110	0.631419	0.647137	0.646876	0.624824	0.550412
5	0.476527	0.603320	0.555955	0.556171	0.566415	0.590858	0.463545	0.562018	0.578493	0.652388	0.551679	0.573486	0.526791
6	0.498018	0.569913	0.517900	0.533697	0.557727	0.540020	0.471812	0.524059	0.558378	0.629871	0.519074	0.575352	0.540394
7	0.512231	0.541034	0.500372	0.533219	0.580348	0.539035	0.499073	0.547631	0.597666	0.549291	0.518087	0.542776	0.526717
8	0.564515	0.549948	0.512154	0.519097	0.555763	0.556021	0.518117	0.531699	0.569601	0.545941	0.495236	0.502994	0.520076
9	0.526056	0.525604	0.481923	0.519352	0.565741	0.551700	0.547723	0.523991	0.599621	0.527052	0.469230	0.496032	0.506958
10	0.509048	0.523073	0.464020	0.498003	0.520587	0.557523	0.565371	0.547010	0.586796	0.517534	0.470974	0.515166	0.516708

Still considering the same dataset (R8), we also produce the hierarchical cluster (i.e. dendogram) based on the cosine similarity after applying our techniques. Two measures are extracted from these experiments [171]: consistency and dissimilarity. The former compares the height of each link in a cluster with the heights of neighboring links below it in the tree. This value is expressed as the *inconsistency* coefficient, so the lower the value, the better the consistency. The latter, represents the *cophenetic distance* (i.e. distance of two clusters or height) between two objects. This coefficient measures how faithfully a dendrogram preserves the pairwise distances between the original unmodeled data points. For this metric, a value close to 1 represents a good result. Figures 6.6, 6.7, and 6.8 show our results.

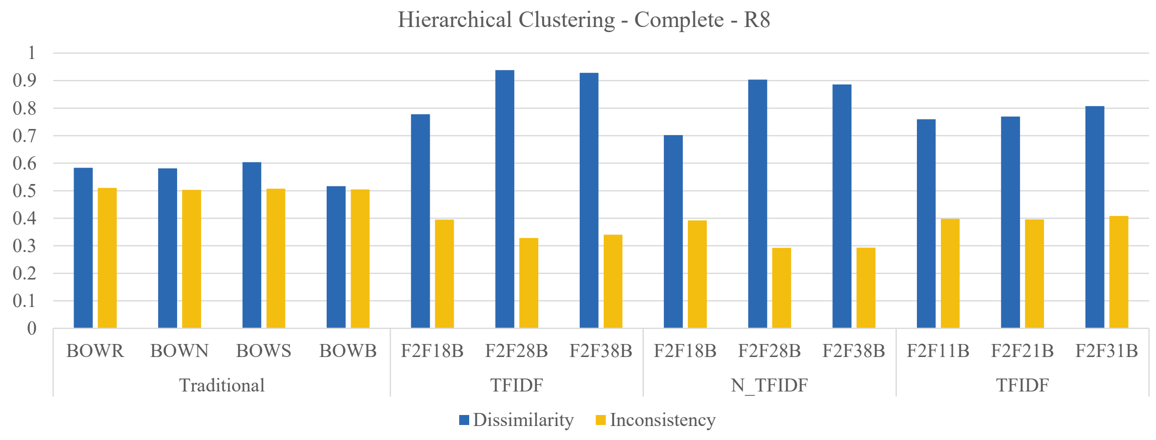


Figure 6.6: Consistency and Dissimilarity using Complete-Linkage for R8.

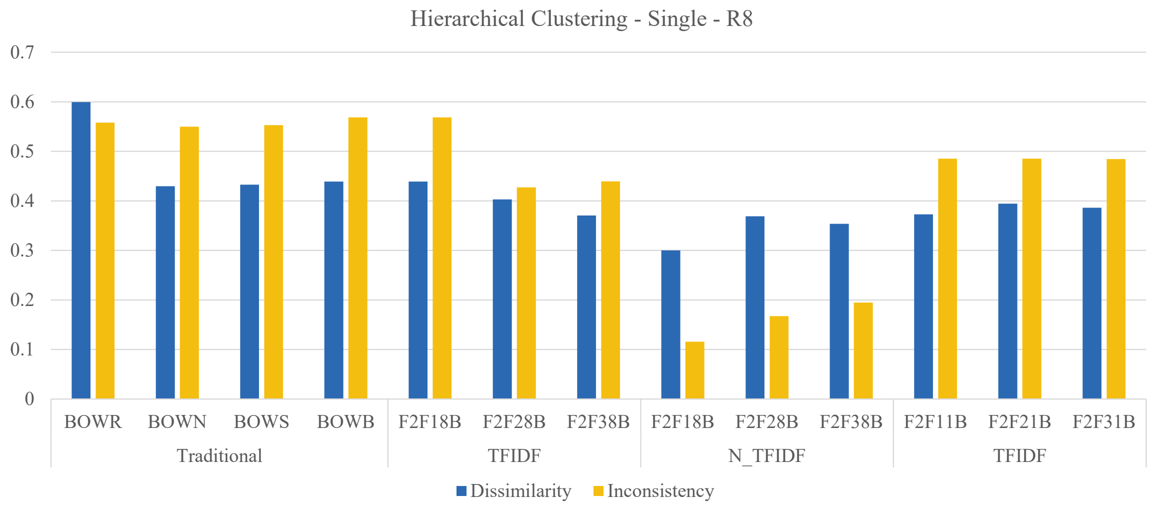


Figure 6.7: Consistency and Dissimilarity using Single-Linkage for R8.

The labels for these experiments follow the same structure as the previous one for the Silhouette value. Considering complete (Figure 6.6) and average (Figure 6.8) linkage, all our results, for both dissimilarity and consistency, outperform traditional approaches. More specifically, for tf-idf - $F2F28B$, $F2F38B$ in complete linkage; and for tf-idf - $F2F28B$, $F2F38B$ and N_tf-idf - $F2F28B$, $F2F38B$ we obtain values above 0.90 for dissimilarity. For single linkage (Figure 6.7) however, our approaches only perform better for consistency.

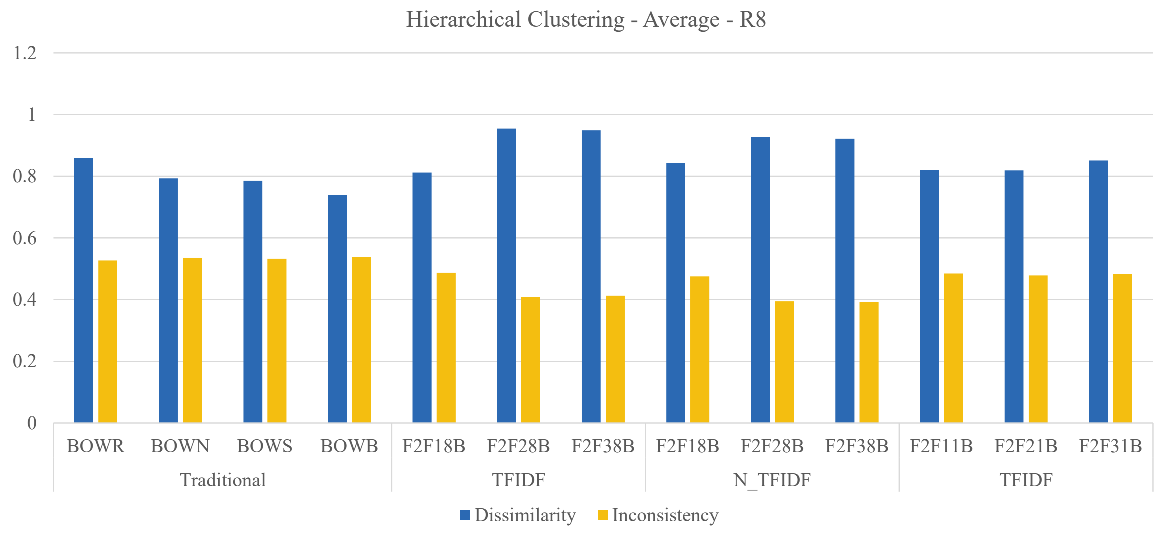


Figure 6.8: Consistency and Dissimilarity using Average-Linkage for R8.

The partial results, in these early experiments, support the use of our proposed approaches to extract and use semantic features in document retrieval, clustering, and classification tasks.

6.3.3 Keyconcept Extraction through Lexical Chains

Another contribution of our research is in the keyword extraction problem. We explore the use of the BSD (Section 6.2.1) and FLC (Section 6.2.2) algorithms to suggest keywords for documents based on their semantic features. In these experiments, we use the POC dataset discussed in Section 6.3. We reduce our corpus to the synsets produced using BSD and FLC. Next, we rank them multiplying their quantity to the weight value obtained through tf-idf (Section 6.2.6). Table 6.11 shows the number of synset obtained per document in each category. It also shows the average of unique synsets per document category.

It is evident that the number of synsets obtained through BSD are far more numerous, if compared to those produced using FLC. The reason is because in the former, we are analyzing separate words, considering only the concepts within its immediate surroundings (predecessor and successor). In the latter, the chains are tracking the continuity of more broad ideas, so the groups of synsets are clustered into a more common concept, which, in

Table 6.11: Distribution of synsets obtained through BSD and FLC.

	BSD			**FLC**		
Docs	**Dogs**	**Computers**	**Sports**	**Dogs**	**Computers**	**Sports**
Doc_01	627	1458	1546	152	364	368
Doc_02	461	526	1477	91	113	365
Doc_03	1229	382	924	281	95	214
Doc_04	687	793	1007	161	185	260
Doc_05	535	1276	1136	126	318	268
Doc_06	608	1336	346	143	379	82
Doc_07	693	578	911	156	119	226
Doc_08	978	501	161	229	96	396
Doc_09	1285	1205	1473	335	301	382
Doc_10	677	1453	1018	161	349	242
Average	**778**	**951**	**1000**	**184**	**232**	**280**

turn, is represented in a single synset. In other words, BSD produces fine grained results with one synset per existing word in WordNet, while FLC provides general abstractions reducing to total number of synsets in a document.

Once the keyword candidates are obtained (in decreasing order, based on their tf-idf values), we analyze the top-5 ones through a survey comparing them with the document-category defined in each Wikipedia article (ground truth). These categories are implemented considering MediaWiki[4], which adds an automated listing to represent and incorporate a given webpage to a subject area, which can be found at the bottom of every Wikipedia article.

The main objective of this experiment is to evaluate whether keywords obtained through our proposed techniques can represent the essential concepts in an given article. Thus, we create a range with scores varying from 1 (strongly disagree) to 5 (strong agree) to assess the quality of the suggested keywords. This experiment is performed through a survey answered by members of our department which consist of 8 people (1 Full Professor, 4 Ph.D. candidates, 2 Ph.D. students and 1 M.S. student).

Tables 6.12 and 6.13 show a sample of how the participants correlate the categories

[4]https://en.wikipedia.org/wiki/MediaWiki

Table 6.12: MediaWiki categories sample.

Document	Key 01	Key 02	Key 03	Key 04	Key 05
Doc_X	Computer hardware	Electronics			
Doc_Y	Computer programming	Computers			
Doc_Z	Linux	1991 software	Computing platforms	Cross-platform software	Finnish inventions

in MediaWiki (Table 6.12) and the ones suggested through our algorithms (Table 6.13), respectively. They are asked to compare and score the values for matching documents in both groups, the ground truth and the suggested keywords. The exercise in our survey is done for each document (30 Wikpedia articles), considering the first 5 synsets for the BSD and FLC approaches. A total of 300 records are assessed per participant, which provides 2,400 final evaluations in the entire survey.

Table 6.13: Suggested synsets sample.

Synset ID	Type	Doc
SID-06321054-N	intensifier,intensive	Doc_X
SID-06566077-N	software,software_program,computer_software	Doc_X
SID-00928947-N	programming,programing,computer_programming	Doc_X
SID-05650820-N	language,speech	Doc_X
SID-05996646-N	discipline,subject,subject_area,subject_field	Doc_X
SID-05847438-N	algorithm,algorithmic_rule,algorithmic_program	Doc_X
SID-03642806-N	laptop,laptop_computer	Doc_Y
SID-03699975-N	machine	Doc_Y
SID-07739125-N	apple	Doc_Y
SID-09840217-N	baron,big_businessman,business_leader,king	Doc_Y
SID-13645010-N	horsepower,HP,H.P.	Doc_Y
SID-13699442-N	dram,	Doc_Z
SID-09752246-N	Aries,Ram	Doc_Z
SID-13912992-N	constriction,bottleneck,chokepoint	Doc_Z
SID-03744276-N	memory,computer_memory,storage	Doc_Z
SID-06198876-N	devices	Doc_Z

For each document in our corpus and a given keyword, we compute a mean rating over all survey participants. For this experiment, we found 5 keywords per document and perform a weighted average over the 5 mean ratings we have computed, where the weights

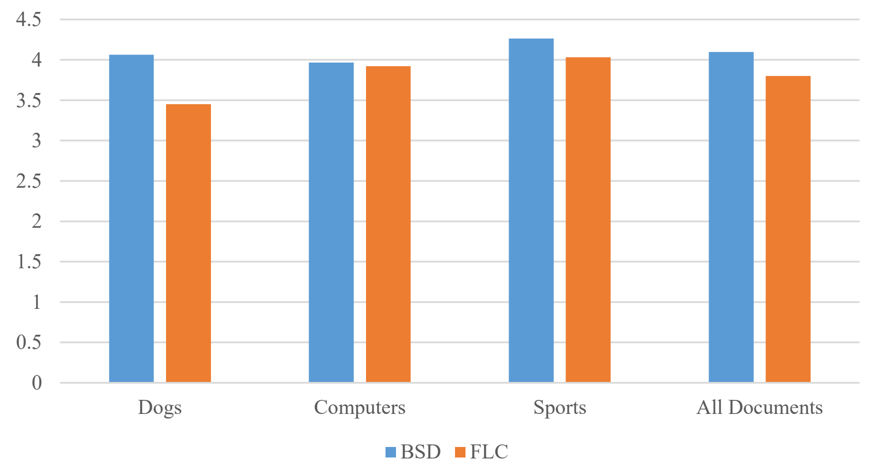


Figure 6.9: Keyword survey average rankings.

correspond to the tf-idf associated with each of the five keywords.

In Figure 6.9, we show the average of the results over each of the three categories of our corpus (i.e. documents of dogs, computers, sports), so we can capture a more robust outcome from our participants. We also show, in Figure 6.10, the correlation between the individual tf-idf values for each proposed keyword and the average score given by the human reviewers. We see that the correlation is higher for the BSD approach than for the FLC approach, and that the average correlation is high.

The BSD algorithm produces better results in capturing the main concepts for all document categories, when compared to the FLC algorithm. As mentioned previously, BSD provides synsets with more details, which results more precise keywords. FLC, on the other hand, provides more general abstractions, that do not perform as well as the previous technique. An example of this situation is document D6, the sixth document concerning *Dogs*. The BSD method has the words *kennel* and *doghouse*, while the FLC approach suggests the words *building* and *edifice*.

In this experiment we explore how semantic features can be extracted and used for keyword suggestion. Instead of relying solely on syntax analysis, statistical approaches or annotated corpora, we apply multiple techniques that use semantic representation to recommend possible keywords. This representation is obtained through two of our proposed

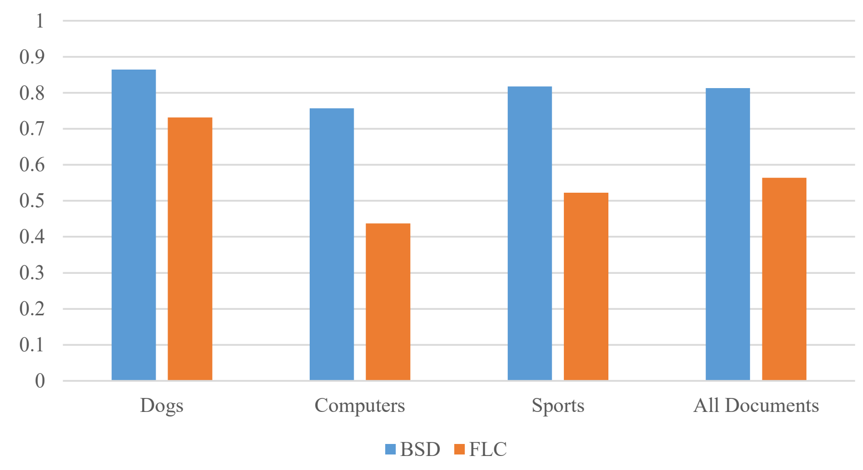


Figure 6.10: Keyword survey average correlations with keyword strengths.

techniques, BSD and FLC. Both consider the context surrounding each word to better represent the meaning underlying the text itself. While the former provides a more detailed extraction of synsets, the latter captures more general aspects discussed in the corpus.

BIBLIOGRAPHY

[1] (2010). Princeton University "About WordNet." WordNet. Princeton University.

[2] Achakulvisut, T., Acuna, D. E., Ruangrong, T., and Kording, K. (2016). Science Concierge: A fast content-based recommendation system for scientific publications. *PLoS ONE*, 11(7):1–11.

[3] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

[4] Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

[5] AlAgha, I. and Nafee, R. (2014). An Efficient Approach For Semantically-Enhanced Document Clustering By Using Wikipedia Link Structure. *International Journal of Artificial Intelligence & Applications*, 5(6):53–62.

[6] Bär, D., Zesch, T., and Gurevych, I. (2015). Composing measures for computing text similarity.

[7] Barzilay, R. and Elhadad, M. (1997). Using lexical chains for text summarization. *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 17(48):10–17.

[8] Beel, J. and Dinesh, S. (2017). Real-world recommender systems for academia: The pain and gain in building, operating, and researching them [long version]. *CoRR*, abs/1704.00156.

[9] Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2015). Research-paper recommender systems: A literature survey. *Int. J. Digit. Libr.*, 17(4):305–338.

[10] Beel, J., Langer, S., Genzmehr, M., Gipp, B., Breitinger, C., and Nürnberger, A. (2013). Research paper recommender system evaluation: A quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, pages 15–22, New York, NY, USA. ACM.

[11] Beierle, F., Aizawa, A., and Beel, J. (2017). Exploring choice overload in related-article recommendations in digital libraries. *CoRR*, abs/1704.00393.

[12] Beliga, S. (2014). Keyword extraction: a review of methods and approaches. *University of Rijeka, Department of Informatics, Rijeka*, pages 1–9.

[13] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

[14] Berg-Kirkpatrick, T., Burkett, D., and Klein, D. (2012). An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 995–1005, Stroudsburg, PA, USA. Association for Computational Linguistics.

[15] Berry, M. and Kogan, J. (2010). *Text Mining: Applications and Theory*. Wiley InterScience. Wiley.

[16] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

[17] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[18] Bollacker, K. D., Lawrence, S., and Giles, C. L. (1998). Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, pages 116–123, New York, NY, USA. ACM.

[19] Boratto, L., Carta, S., Fenu, G., and Saia, R. (2017). Semantics-aware content-based recommender systems: Design and architecture guidelines. *Neurocomputing*, 254(Supplement C):79 – 85. Recent Advances in Semantic Computing and Personalization.

[20] Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2012). Joint learning of words and meaning representations for open-text semantic parsing. In *In Proceedings of 15th International Conference on Artificial Intelligence and Statistics*.

[21] Brandtzæg, P. B. (2013). Big data, for better or worse: 90% of world's data generated over last two years. Online. SINTEF.

[22] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[23] Bruni, E., Boleda, G., Baroni, M., and Tran, N.-K. (2012). Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 136–145, Stroudsburg, PA, USA. Association for Computational Linguistics.

[24] Budanitsky, A. and Hirst, G. (2001). Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pages 29–24, Pittsburgh, PA.

[25] Budanitsky, A. and Hirst, G. (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1):13–47.

[26] Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2015). A unified multilingual semantic representation of concepts. In *ACL (1)*, pages 741–751. The Association for Computer Linguistics.

[27] Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2016). Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artif. Intell.*, 240:36–64.

[28] Carrell, P. L. (1982). Cohesion is not coherence. *TESOL Quarterly*, 16(4):479–488.

[29] Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder. *CoRR*, abs/1803.11175.

[30] Chen, P.-I. and Lin, S.-J. (2010). Automatic keyword prediction using google similarity distance. *Expert Systems with Applications*, 37(3):1928 – 1938.

[31] Chen, T., Xu, R., He, Y., and Wang, X. (2015). Improving distributed representation of word sense via wordnet gloss composition and context clustering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 15–20. The Association for Computational Linguistics.

[32] Chen, X., Liu, Z., and Sun, M. (2014). A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. ACL.

[33] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

[34] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[35] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[36] Dai, A. M., Olah, C., and Le, Q. V. (2015). Document embedding with paragraph vectors.

[37] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

[38] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

[39] Dhillon, P. S., Foster, D., and Ungar, L. (2011). Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24.

[40] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271.

[41] Dong, H., Hussain, F. K., and Chang, E. (2009). *A Hybrid Concept Similarity Measure Model for Ontology Environment*, pages 848–857. Springer Berlin Heidelberg, Berlin, Heidelberg.

[42] Drucker, H., Wu, D., and Vapnik, V. N. (1999). Support vector machines for spam categorization. *Trans. Neur. Netw.*, 10(5):1048–1054.

[43] Enríquez, F., Troyano, J. A., and López-Solaz, T. (2016). An approach to the use of word embeddings in an opinion classification task. *Expert Systems with Applications*, 66:1 – 6.

[44] Ercan, G. and Cicekli, I. (2007). Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705 – 1714. Text Summarization.

[45] Ercan, G. and Cicekli, I. (2008). Lexical cohesion based topic modeling for summarization. In *Computational Linguistics and Intelligent Text Processing, 9th International Conference, CICLing 2008, Haifa, Israel, February 17-23, 2008, Proceedings*, pages 582–592.

[46] Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E. H., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *HLT-NAACL*, pages 1606–1615. The Association for Computational Linguistics.

[47] Faruqui, M., Tsvetkov, Y., Rastogi, P., and Dyer, C. (2016). Problems with evaluation of word embeddings using word similarity tasks. *CoRR*, abs/1605.02276.

[48] Fellbaum, C., editor (1998). *WordNet: an electronic lexical database*. MIT Press.

[49] Ferreira, C. H. P., de França, F. O., and de Medeiros, D. M. R. (2018a). Combining multiple views from a distance based feature extraction for text classification. In *2018 IEEE Congress on Evolutionary Computation, CEC 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–8.

[50] Ferreira, C. H. P., de Medeiros, D. M. R., and de França, F. O. (2018b). Dcdistance: A supervised text document feature extraction based on class labels.

[51] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

[52] Firth, J. R. (1957). *A synopsis of linguistic theory 1930-55.*, volume 1952-59. The Philological Society, Oxford.

[53] Fu, M., Qu, H., Huang, L., and Lu, L. (2018). Bag of meta-words: A novel method to represent document for the sentiment classification. *Expert Systems with Applications*, 113:33 – 43.

[54] Gale, W. A., Church, K. W., and Yarowsky, D. (1992). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5/6):415–439.

[55] Garfield, E. (1955). Citation indexes for science: A new dimension in documentation through association of ideas. *Science*, 122(3159):108–111.

[56] Garfield, E. (2009). From the science of science to scientometrics visualizing the history of science with histcite software. *J. Informetrics*, 3(3):173–179.

[57] Gerz, D., Vulić, I., Hill, F., Reichart, R., and Korhonen, A. (2016). SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *EMNLP*.

[58] Gonzales, A. R., Mascarell, L., and Sennrich, R. (2017). Improving word sense disambiguation in neural machine translation with sense embeddings. In Bojar, O., Buck, C., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno-Yepes, A., Koehn, P., and Kreutzer, J., editors, *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 11–19. Association for Computational Linguistics.

[59] Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

[60] Grosky, W. I. and Ruas, T. L. (2017). The Continuing Reinvention of Content-Based Retrieval: Multimedia Is Not Dead. *IEEE MultiMedia*, 24(1):6–11.

[61] Guo, W. and Diab, M. (2011). Semantic topic models: Combining word distributional statistics and dictionary definitions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 552–561, Edinburgh, Scotland, UK. Association for Computational Linguistics.

[62] Halliday, M. A. K. and Hasan, R. (1976). *Cohesion in english.* Longman Group, London.

[63] Han, E.-H. S. and Karypis, G. (2000). *Centroid-Based Document Classification: Analysis and Experimental Results*, pages 424–431. Springer Berlin Heidelberg, Berlin, Heidelberg.

[64] Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

[65] Hasan, K. S. and Ng, V. (2010). Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 365–373, Stroudsburg, PA, USA. Association for Computational Linguistics.

[66] Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA. ACM.

[67] Hein, A. M. (2010). Identification and bridging of semantic gaps in the context of multi-domain engineering. Abstract.

[68] Heydari, A., Tavakoli, M. a., Salim, N., and Heydari, Z. (2015). Detection of review spam. *Expert Syst. Appl.*, 42(7):3634–3642.

[69] Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

[70] Hornik, K., Feinerer, I., Kober, M., and Buchta, C. (2012). Spherical k-means clustering. *Journal of Statistical Software, Articles*, 50(10):1–22.

[71] Hotho, A., Staab, S., and Stumme, G. (2003). Wordnet improves text document clustering. In *Proc. of the SIGIR 2003 Semantic Web Workshop*, Toronto, Canada.

[72] Houvardas, J. and Stamatatos, E. (2006). N-gram feature selection for authorship identification. In Euzenat, J. and Domingue, J., editors, *Artificial Intelligence: Methodology, Systems, and Applications*, pages 77–86, Berlin, Heidelberg. Springer Berlin Heidelberg.

[73] Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

[74] Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2015). Sensembed: Learning sense embeddings for word and relational similarity. In *ACL (1)*, pages 95–105. The Association for Computer Linguistics.

[75] Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2016). Embeddings for word sense disambiguation: An evaluation study. In *ACL (1)*. The Association for Computer Linguistics.

[76] Jain, J. (2016). Fasttext and gensim word embeddings. Online.

[77] Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008.

[78] Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[79] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142, Berlin, Heidelberg. Springer-Verlag.

[80] Johansson, R. and Piña, L. N. (2015). Embedding a semantic network in a word space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Denver, United States, May 31 âĂŞ June 5, 2015*, pages 1428–1433.

[81] Jonnalagadda, S., Cohen, T., Wu, S. T.-I., and Gonzalez, G. (2012). Enhancing clinical concept extraction with distributional semantics. *Journal of Biomedical Informatics*, 45(1):129–140.

[82] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[83] Kamkarhaghighi, M. and Makrehchi, M. (2017). Content tree word embedding for document representation. *Expert Systems with Applications*, 90:241 – 249.

[84] Kazemi, B. and Abhari, A. (2017). A comparative study on content-based paper-to-paper recommendation approaches in scientific literature. In *Proceedings of the 20th Communications & Networking Symposium*, CNS '17, pages 5:1–5:10, San Diego, CA, USA. Society for Computer Simulation International.

[85] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

[86] Lau, J. H. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *Proceedings of the 1st Workshop on Representation Learning for NLP*.

[87] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents.

[88] Leacock, C. and Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. In Fellbaum, C., editor, *WordNet: An Electronic Lexical Database*, pages 265–283. MIT Press.

[89] Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

[90] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

[91] Li, J. and Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics.

[92] Li, Y., Bandar, Z. A., and McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):871–882.

[93] Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[94] Liu, H. and Singh, P. (2004). Conceptnet: A practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

[95] Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J., and Li, H. (2017). Generative adversarial network for abstractive text summarization.

[96] Liu, X., He, P., Chen, W., and Gao, J. (2019). Multi-task deep neural networks for natural language understanding.

[97] Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015). Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2418–2424. AAAI Press.

[98] Mallery, J. C. (1988). Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers. In *Master's thesis, M.I.T. Political Science Department*.

[99] Mancini, M., Camacho-Collados, J., Iacobacci, I., and Navigli, R. (2017). Embedding words and senses together via joint knowledge-enhanced training. In *CoNLL*, pages 100–111. Association for Computational Linguistics.

[100] Mascarell, L. (2017). Lexical chains meet word embeddings in document-level statistical machine translation. In Webber, B. L., Popescu-Belis, A., and Tiedemann, J., editors, *Proceedings of the Third Workshop on Discourse in Machine Translation, DiscoMT@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 99–109. Association for Computational Linguistics.

[101] MATLAB (2017). *version 9.2.0 (R2017a)*. The MathWorks Inc., Natick, Massachusetts.

[102] Matsuo, Y. and Ishizuka, M. (2003). Keyword extraction from a single document using word co-occurrence statistical information. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 392–396. AAAI Press.

[103] McCarthy, D., Koeling, R., Weeds, J., and Carroll, J. (2004). Finding predominant word senses in untagged text. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

[104] McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall.

[105] Mcgee, I. (2009). Traversing the lexical cohesion minefield. *ELT J*, 63(3):212–220.

[106] Medeiros, D. M. R. and Carvalho, A. C. P. L. F. (2004). Gene clusters analysis using text mining. In *WOB - Third Workshop on Bioinformatics*, pages 141–144, Brasília - DF. SBC.

[107] Medeiros, D. M. R. and Carvalho, A. C. P. L. F. (2005). Applying text mining and machine learning techniques to gene clusters analysis. In *ICCIMA '05: Proceedings of the Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05)*, pages 23–28, Washington, DC, USA. IEEE Computer Society.

[108] Meng, L., Huang, R., and Gu, J. (2013). A Review of Semantic Similarity Measures in WordNet. *International Journal of Hybrid Information Technology*, 6(1):1–12.

[109] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[110] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[111] Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

[112] Miller, G. A. (1995). Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

[113] Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language & Cognitive Processes*, 6(1):1–28.

[114] Mitchell, T. M. (1997). *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill.

[115] Moldovan, D. and Novischi, A. (2002). Lexical chains for question answering. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

[116] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS*. Society for Artificial Intelligence and Statistics.

[117] Moro, A., Raganato, A., and Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244.

[118] Morris, J. and Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–48.

[119] Nalimov, V. and Mul'chenko, Z. (1969). *Measurement of Science, Study of the Development of Science as an Information Process*. Machine aided translation of Naukometriya.

[120] Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69.

[121] Navigli, R. and Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

[122] Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, pages 1059–1069. ACL.

[123] Ng, H. T. (1997). Getting serious about word sense disambiguation.

[124] Oele, D. and Noord, v. G. (2018). Simple Embedding-Based Word Sense Disambiguation. In *Proceedings of the 9th 9th Global Wordnet Conference*, Nanyang Technological University (NTU), Singapore. Association for Computational Linguistics.

[125] Panchenko, A. (2016). Best of both worlds: Making word sense embeddings interpretable. In Chair), N. C. C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

[126] Paraschiv, I. C., Dascalu, M., Trausan-Matu, S., Nistor, N., Oca, A. M. M. D., and McNamara, D. S. (2017). Semantic similarity versus co-authorship networks: A detailed comparison. In *21st International Conference on Control Systems and Computer Science (CSCS)*, pages 566–570. IEEE Xplorer.

[127] Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet::similarity: Measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

[128] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[129] Pelevina, M., Arefyev, N., Biemann, C., and Panchenko, A. (2016). Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183.

[130] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

[131] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

[132] Pilehvar, M. T., Camacho-Collados, J., Navigli, R., and Collier, N. (2017). Towards a seamless integration of word senses into downstream nlp applications. *CoRR*, abs/1710.06632.

[133] Pilehvar, M. T. and Collier, N. (2016). De-conflated semantic representations. In *EMNLP*, pages 1680–1690. The Association for Computational Linguistics.

[134] Pradhan, N., Gyanchandani, M., and Wadhvani, R. (2015). Article: A review on text similarity technique used in ir and its application. *International Journal of Computer Applications*, 120(9):29–34. Full text available.

[135] Price, D. J. d. S. (1961). *Science since Babylon / Derek de Solla Price*. Yale University Press New Haven, enl. ed. edition.

[136] Putnam, H. (1970). Is semantics possible? *Metaphilosophy*, 1(3):187–201.

[137] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

[138] Reinsel, D., Gantz, J., and Rydning, J. (2018). The digitaliztion of the world from edge to core.

[139] Reisinger, J. and Mooney, R. (2010a). A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1173–1182, Stroudsburg, PA, USA. Association for Computational Linguistics.

[140] Reisinger, J. and Mooney, R. J. (2010b). Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 109–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

[141] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[142] Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*, pages 1–35. Springer US, Boston, MA.

[143] Roget, P. M. (1979). *Roget's International Thesaurus (Harper Colophon Books)*. HarperCollins Publishers.

[144] Rollins, J., McCusker, M., Carlson, J., and Stroll, J. (2017). Manuscript matcher: A content and bibliometrics-based scholarly journal recommendation system. In *BIR@ECIR*, volume 1823 of *CEUR Workshop Proceedings*, pages 18–29. CEUR-WS.org.

[145] Rothe, S. and Schütze, H. (2015). Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL (1)*, pages 1793–1803. The Association for Computer Linguistics.

[146] Rothe, S. and Schütze, H. (2017). Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617.

[147] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(Supplement C):53 – 65.

[148] Ruas, T. and Grosky, W. (2017a). Keyword Extraction Through Contextual Semantic Analysis of Documents. In *Proceedings of the 9th International Conference on Management of Emergent Digital EcoSystems*, Bangkok. ACM Press.

[149] Ruas, T. and Grosky, W. (2018). Semantic Feature Structure Extraction from Documents Based on Extended Lexical Chains. In *Proceedings of the 9th 9th Global Wordnet Conference*, Nanyang Technological University (NTU), Singapore. Association for Computational Linguistics.

[150] Ruas, T., Grosky, W., and Aizawa, A. (2019). Multi-sense embeddings through a word sense disambiguation process. Pre-Print.

[151] Ruas, T. and Grosky, W. I. (2017b). Keyword extraction through contextual semantic analysis of documents. In Chbeir, R., Kawtrakul, A., Grosky, W. I., and Ouni, A., editors, *Proceedings of the 9th International Conference on Management of Digital EcoSystems, MEDES 2017, Bangkok, Thailand, November 07-10, 2017*, pages 150–156. ACM.

[152] Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

[153] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[154] Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[155] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[156] Salton, G. and Yang, C. S. (1973). On the specification of term values in automatic indexing. *Journal of Documentation.*, 29(4):351–372.

[157] Sanders, T. and Pander Maat, H. (2006). Cohesion and Coherence: Linguistic Approaches. In Quirk, R., editor, *Encyclopedia of Language & Linguistics*, pages 591–595. Elsevier, London.

[158] Saratlija, J., Šnajder, J., and Dalbelo Bašić, B. (2011). *Unsupervised Topic-Oriented Keyphrase Extraction and Its Application to Croatian*, pages 340–347. Springer Berlin Heidelberg, Berlin, Heidelberg.

[159] Sedding, J. and Kazakov, D. (2004). Wordnet-based text document clustering. In *Proceedings of the 3rd Workshop on RObust Methods in Analysis of Natural Language Data*, ROMAND '04, pages 104–113, Stroudsburg, PA, USA. Association for Computational Linguistics.

[160] Shaoul, C. and Westbury, C. (2010). The westbury lab wikipedia corpus.

[161] Shi, T. and Liu, Z. (2014). Linking glove with word2vec.

[162] Silber, H. G. and McCoy, K. F. (2000). Efficient Text Summarization Using Lexical Chains. *Proceedings of the ACM Conference on Intelligent User Interfaces*, pages 252–255.

[163] Silva, J. and Lopes, G. (2010). Towards automatic building of document keywords. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1149–1157, Stroudsburg, PA, USA. Association for Computational Linguistics.

[164] Simov, K. I., Boytcheva, S., and Osenova, P. (2017). Towards lexical chains for knowledge-graph-based word embeddings. In *RANLP*, pages 679–685. INCOMA Ltd.

[165] Sinoara, R. A., Camacho-Collados, J., Rossi, R. G., Navigli, R., and Rezende, S. O. (2019). Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, 163:955 – 971.

[166] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[167] Székely, G. J. and Rizzo, M. L. (2005). Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *J. Classification*, 22(2):151–183.

[168] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA. ACM.

[169] Tanimoto, T. (1957). An elementary mathematical theory of classification and prediction. Ibm internal report 17th, IBM.

[170] Tekli, J. (2016). An overview on xml semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges. *IEEE Trans. Knowl. Data Eng.*, 28(6):1383–1407.

[171] Thompson, C. and Shure, L. (2017). *Hierarchical Clustering;[User's Guide]*. The MathWorks Inc.

[172] Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., and Liu, T.-Y. (2014). A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 151–160.

[173] Trask, A., Michalak, P., and Liu, J. (2015). sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR*, abs/1511.06388.

[174] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

[175] Turing, A. M. (1950). I. Computing Machinery and Intelligence. *Mind*, LIX(236):433–460.

[176] Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141.

[177] Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

[178] Weaver, W. (1949). Translation. In Locke, W. N. and Boothe, D. A., editors, *Machine Translation of Languages*, pages 15–23. MIT Press.

[179] Wei, T., Lu, Y., Chang, H., Zhou, Q., and Bao, X. (2015). A semantic approach for text clustering using wordnet and lexical chains. *Expert Syst. Appl.*, 42(4):2264–2275.

[180] Whissell, J. S. and Clarke, C. L. A. (2011). Improving document clustering using okapi bm25 feature weighting. *Information Retrieval*, 14(5):466–487.

[181] Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics.

[182] Xia, F., Liu, H., Lee, I., and Cao, L. (2016). Scientific Article Recommendation: Exploiting Common Author Relations and Historical Preferences. *IEEE Transactions on Big Data*, 2(2):101–112.

[183] Yang, D. and Powers, D. M. W. (2006). Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

[184] Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90.

[185] Yao, Y., Li, X., Liu, X., Liu, P., Liang, Z., Zhang, J., and Mai, K. (2017). Sensing spatial distribution of urban land use by integrating points-of-interest and google word2vec model. *Int. J. Geogr. Inf. Sci.*, 31(4):825–848.

[186] Yu, M. and Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *ACL (2)*, pages 545–550.

[187] Zhang, M., Yang, H., Ji, D., Teng, C., and Wu, H. (2013). *Discourse Coherence: Lexical Chain, Complex Network and Semantic Field*, pages 756–765. Springer Berlin Heidelberg, Berlin, Heidelberg.

[188] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

[189] Zheng, R., Li, J., Chen, H., and Huang, Z. (2006). A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, 57(3):378–393.

[190] Zhou, Z., Wang, Y., and Gu, J. (2008). New model of semantic similarity measuring in wordnet.

[191] Zou, W. Y., Socher, R., Cer, D. M., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.