Distinguishing Antonymy, Synonymy and Hypernymy with Distributional and Distributed Vector Representations and Neural Networks

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung.

Vorgelegt von

Kim Anh Nguyen

aus Vietnam

Hauptberichter PD. Dr. Sabine Schulte im Walde
Mitberichter Prof. Dr. Ngoc Thang Vu

Tag der mündlichen Prüfung: July 31, 2018

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2018

Abstract

In the last decade, computational models that distinguish semantic relations have become crucial for many applications in Natural Language Processing (NLP), such as machine translation, question answering, sentiment analysis, and so on. These computational models typically distinguish semantic relations by either representing semantically related words as vector representations in the vector space, or using neural networks to classify semantic relations. In this thesis, we mainly focus on the improvement of such computational models. Specifically, the goal of this thesis is to address the tasks of distinguishing antonymy, synonymy, and hypernymy.

For the task of distinguishing antonymy and synonymy, we propose two approaches. In the first approach, we focus on improving both families of word vector representations, which are distributional and distributed vector representations. Regarding the improvement of distributional vector representation, we propose a novel weighted feature for constructing word vectors by relying on distributional lexical contrast, a feature capable of differentiating between antonymy and synonymy. In terms of the improvement of distributed vector representations, we propose a neural model to learn word vectors by integrating distributional lexical contrast into the objective function of the neural model. The resulting word vectors can distinguish antonymy from synonymy and predict degrees of word similarity. In the second approach, we aim to use lexico-syntactic patterns to classify antonymy and synonymy. To do so, we propose two pattern-based neural networks to distinguish antonymy from synonymy. The lexico-syntactic patterns are induced from the syntactic parse trees and then encoded as vector representations by neural networks. As a result, the two pattern-based neural networks improve performance over prior pattern-based methods.

For the tasks of distinguishing hypernymy, we propose a novel neural model to learn hierarchical embeddings for hypernymy detection and directionality. The hierarchical embeddings are learned according to two underlying aspects (i) that the similarity of hypernymy is higher than similarity of other relations, and (ii) that the distributional hierarchy is generated between hyponyms and hypernyms. The experimental results show that hierarchical embeddings significantly outperform state-of-the-art word embeddings.

In order to improve word embeddings for measuring semantic similarity and relatedness, we propose two neural models to learn word denoising embeddings by filtering noise from original word embeddings without using any external resources. Two proposed neural models receive original word embeddings as inputs and learn denoising matrices to filter noise from original word embeddings. Word denoising embeddings achieve the improvement against original word embeddings over tasks of semantic similarity and relatedness. Furthermore, rather than using English, we also shift the focus on evaluating the performance of computational models to Vietnamese. To that effect, we introduce two novel datasets of (dis-)similarity and relatedness for Vietnamese. We then make use of computational models to verify the two datasets and to observe their performance in being adapted to Vietnamese. The results show that computational models exhibit similar behaviour in the two Vietnamese datasets as in the corresponding English datasets.

Zusammenfassung

Komputationalle Modelle, die in der Lage sind, semantische Relationen zu unterscheiden, sind für eine Vielzahl von Anwendungen im Bereich der Computerlinguistik nützlich. So kann das Wissen über semantischen Relationen für maschinelle Übersetzung, Frage-Antwort Systeme oder auch Sentimentanalyse hilfreich sein. Ansätze dieser Art nutzen in der Regel Vektorraummodelle, um semantisch ähnliche Wörter darzustellen, oder neuronale Netze um semantische Relationen zu klassifizieren. Das zentrale Thema, der vorliegenden Dissertation, ist die Verbesserung von komputationellen Modellen zur korrekten Unterscheidung und Erkennung von den folgenden drei semantischen Relationen: Antonymie, Synonymie sowie Hyperonymie.

Wie präsentieren zunächst zwei Ansätze für die binäre Unterscheidung zwischen Antonymie und Synonymie. Der erste Ansatz verwendet lexikalischen Kontrast, ein Merkmal zur Unterscheidung der beiden Relationen, dass wir in unterschiedliche Arten von Wortrepräsentationen integrieren. Für distributionelle Vektoren präsentieren wir eine Merkmalsgewichtung, die lexikalischen Kontrast induziert. Für distributed Vektoren (Embeddings) präsentieren wir ein neues neuronales Netz, dass lexikalischen Kontrast beim Lernen der Repräsentationen berücksichtigt. Unsere Ergebnisse zeigen, dass unsere neuen Wortrepräsentationen Synonymie und Antonymie unterscheiden können und darüber hinaus Wortähnlichkeiten verbessert vorhersagen können.

Der zweite Ansatz verwendet lexikalisch-syntaktische Muster, basierend auf Korpus-Kookkurrenz sowie Parsebäumen. Wir präsentieren zwei muster-basierte neuronale Netze, die bisherige Ansätze verbessern.

Für die Erkennung von Hypernymie präsentieren wir ein neues neuronales Modell, dass hierarchische Repräsentation erlernt. Die hierarchischen Repräsentation beinhalten dabei zwei wichtige Aspekte i) Die Hyperonymy relation erhält eine höhere Ähnlichkeit als andere semantische Relationen und ii) Aufgrund der hierarchischen Struktur der Wortrepräsentationen kann zwischen Hyponym und Hyperonym unterschieden werden. Unsere Experimente zeigen, dass unsere hierarchischen Räpresentationen signifikant bessere Ergebnisse erzielen, als bisherige Ansätze mit Wortrepräsentationen.

Des Weiteren präsentieren wir zwei Denoising-Methoden, diese Ansätze bereinigen Wortrepräsentationen, sodass diese verbessert Wortähnlichkeiten vorhersagen. Beide Methoden verwenden hierbei keine externen Resourcen, sondern benötigen lediglich Wortrepräsentationen als Eingabe. Unsere Evaluation zeigt Verbesserungen durch unsere Methode für die Vorhersage von semantischer Ähnlichkeit.

Darüber hinaus präsentieren wir zwei neue Datensätze, die komputationelle Modelle nutzen können, um Ähnlichkeiten zwischen Wörtern vorherzusagen. Während die Literatur überwiegend Ressourcen für die englische Sprache beinhaltet, präsentieren wir Datensätze für die Vietnamesische Sprache. Des Weiteren erstellen wir komputationelle Modelle um beide Datensätze zu verifizieren. Unsere Ergebnisse zeigen, dass komputationelle Modelle auf unseren neuen Datensätzen ähnliches Verhalten zeigen wie auf entsprechenden englischen Datensätzen.

List of Publications

Parts of the research described in this thesis have been published in the papers listed below:

- Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2016a). Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany

- Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2016b). Neural-based noise filtering from word embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2699–2707, Osaka, Japan

- Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2017b). Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 76–85, Valencia, Spain

- Nguyen, K. A., Köper, M., Schulte im Walde, S., and Vu, N. T. (2017a). Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243, Copenhagen, Denmark

- Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2018). Introducing two vietnamese datasets for evaluating semantic models of (dis-)similarity and relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HTL)*, pages 199–205, New Orleans, Louisiana

Acknowledgments

I would like to thank my advisors, Sabine Schulte im Walde and Ngoc Thang Vu, for the patience guidance, encouragement, and advice they have provided throughout my time as their Ph.D student. Without Sabine's adopting me as her Ph.D student at first, I would not have gone that far in the doctoral journey, and I would not have such a chance to discover and maximise my potentials. I also want to extend my gratitude to small but in-time encouragement such as coffee and cookies in the meetings with all the group members.

I expect to lend this chance to thank all the colleagues in IMS, especially Maximillian, whose ideas and knowledge made a significant contribution to one of my research. He is also the best coffee mate ever whose sharing about family and Germany enriches my experience during Ph.D time.

Last but not least, this thesis would not have been made possible without the whole-hearted support from my family, especially my wife, Nuong, who always takes care of the kids and lends me a great source of motivation in every step I take.

This research received the grant from the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT), the DFG Collaborative Research Centre SFB 732, and the DFG Heisenberg Fellowship SCHU-2580/1 led by Sabine Schulte im Walde.

Contents

List of Abbreviations

AP Average Precision

CBOW Continuous Bag-of-Words Model

dLCE distributional Lexical-Contrast Embeddings

LMI Local Mutual Information

LSTM Long Short-Term Memory

NLP Natural Language Processing

NNs Neural Networks

PMI Pointwise Mutual Information

PPMI Positive Pointwise Mutual Information

RNN Recurrent Neural Network

SGNS Skip-gram with Negative Sampling

SVM Support Vector Machine

VSM Vector Space Model

1 Introduction

1.1 Motivation

In recent years, NLP has gained impressive achievements in modeling human languages for end-to-end systems. One of the biggest challenges of making full use of the power of end-to-end systems on computers is that they comprehend very little meaning of human language. This obstacle profoundly affects several aspects of computer performance such as the ability to give instructions to those systems, the way the systems respond intelligently to us, and the systems' ability to analyse and process raw input. Among those, understanding and modeling the meaning of words and their semantics is crucial for many applications in NLP such as machine translation, textual entailment, coreference resolution, and taxonomy creation.

To carry out the designed tasks, most NLP applications need to access large amounts of different types of data, which can be unlabeled raw data, labeled data, or parallel data. These kinds of data often provide both implicit and explicit contents for NLP applications through the interpretation of linguistic structures such as morphology, syntax, semantics, discourse, and pragmatics. Therefore, among others, modeling and representing semantics of input data plays a prominent role in most NLP architectures. In linguistic structures, one can exploit many aspects of semantics, such as semantic typing, lexical semantics, semantic relations, semantic similarity, and semantic relatedness. In this thesis, we mainly focus on studying semantic relations. Concretely, we aim to address the tasks of distinguishing antonymy (e.g. *good/bad*), synonymy (e.g. *formal/conventional*), and hypernymy (e.g. *bird/animal*). In addition, we also distinguish degrees of semantic similarity and semantic relatedness, where semantic similarity represents semantic relations of synonymy and co-hyponymy (e.g. *car/bicycle*); and semantic relatedness stands for semantic relations such as hypernymy, antonymy, or meronymy (e.g. *wheels/bicycle*).

Distinguishing between antonymy and synonymy in the computational perspective is typically addressed by modeling those semantic relations in vector space models. These models tend to use the co-occurrence of words and their contexts to represent the meaning of words in vector spaces. Two families of vector space models have entered common usage: distributional and distributed vector representations. However, since words that occur in the same contexts tend to have similar meanings (Harris, 1954), the two families of vector representations are not sufficient to discriminate between synonymy and

antonymy.

In addition to vector space models, pattern-based models are also applied to differentiate between antonymy and synonymy. These models make use of lexico-syntactic patterns, which are string-matching patterns based on lexicon and syntactic structure, to represent antonymous and synonymous word pairs as vector representations. However, the sparsity of such patterns is problematic for these kinds of pattern-based models because of the limited number of lexico-syntactic patterns and their inability to cover all word pairs of antonymy and synonymy.

In a similar way, the task of distinguishing hypernymy is often complicated by issues of hypernymy detection and directionality. Approaches-based vector space models typically detect hypernymy by encoding it as a standard first-order distributional co-occurrence. However, they also similarly encode other relations such as antonymy, synonymy, and meronymy. As a result, the ability of these approaches to distinguish hypernymy from other relations is problematic in terms of accuracy. Further complicating things, hypernymy is an asymmetric relation with a predetermined ordering of the two words in a hypernym pair.

Distributed vector representations often called *word embeddings* have achieved great success in NLP applications in recent years. Apart from being used to encode word meaning for distinguishing semantic relations, word embeddings can be used as an efficient means of representing the inputs of various tasks such as machine translation, question answering, and sentiment analysis. The power of word embeddings is that they are easily trained on a large corpora, which is freely available on the Internet. However, such kinds of corpora also contain a lot of unnecessary information that could negatively affect the performance of word embeddings. For example, the word *"jump"* can be treated as a context of the target word *"cloud"* in a particular sentence. Therefore, word embeddings could be improved by filtering out unnecessary information.

Computation models that distinguish between some aspects of semantic relations are often performed and evaluated on languages rich in morphology, such as English, German, Italian, or Spanish. In other words, these models benefit from the variety of a large corpora and gold standard resources. However, for low-resource languages that are less morphologically rich and lack gold standard resources, modeling and evaluating these computational models remains problematic.

1.2 Strategies

This thesis explores strategies to address the tasks of distinguishing antonymy, synonymy, and hypernymy by improving vector representations of these semantic relations. For the task of distinguishing antonymy and synonymy, we investigate two strategies. The first strategy aims to improve both distributional and distributed word vector representations

by integrating distributional lexical contrast of antonymy and synonymy into word vector representations. As a result, vector representations of antonymous word pairs are further away from each other, while vector representations of synonymous word pairs are closer to each other in the vector space. The second strategy exploits lexico-syntactic patterns between antonymous and synonymous word pairs to distinguish antonymy from synonymy. Instead of using only a small number of standard lexico-syntactic patterns to represent antonymy and synonymy, we induce new lexico-syntactic patterns from the syntactic parse tree of sentences that contain antonymous and synonymous word pairs. This strategy is motivated by empirical linguistic studies on antonymy and synonymy where antonymous word pairs co-occur with each other within a sentence more often than would be expected of synonymous word pairs. By inducing the new lexico-syntactic patterns, we can handle the sparsity of standard lexico-syntactic patterns. In order to model the new lexico-syntactic patterns, we make use of neural networks to encode these patterns as vector representations for distinguishing antonymy from synonymy.

Given the task of distinguishing hypernymy, the strategy is to learn new word embeddings for hypernymy. The new word embeddings of hypernymy are learned to encode two crucial aspects: (i) distributional similarity of hypernymy is strengthened to be higher than distributional similarity of other relations such as synonymy, antonymy, co-hyponymy, and so on; and (ii) distributional hierarchy is generated to differentiate between hyponyms and hypernyms. Regarding these two aspects, the cosine similarity of hypernymy is higher than the cosine similarity of other relations, and the distributional hierarchy between hyponyms and hypernyms is treated as an indicator to determine the directionality of hypernymy.

Word embeddings, as mentioned above, are trained on large raw corpora that contains a lot of unnecessary information, negatively affecting the ability of word embeddings to measure semantic similarity and semantic relatedness. The strategy to deal with this issue is to enhance word embeddings by reducing the negative effects of unnecessary information. To do so, we make use of neural models to filter out unnecessary information from word embeddings by strengthening salient features and weakening unnecessary features. The neural models are designed to receive state-of-the-art word embeddings as inputs and proceed to generate new word embeddings without using any external resources. The new word embeddings are expected to measure similarity and relatedness information better than the state-of-the-art word embeddings do.

At the end of this thesis, the focus shifts from English to Vietnamese, which is a low-resource language. In this perspective, we aim to verify computational models that distinguish between semantic similarity and semantic relatedness with Vietnamese which differs from morphologically rich languages and lacks gold standard resources for evaluating semantic similarity and relatedness. The strategy to deal with this verification is that we introduce newly annotated datasets of (dis-)similarity and relatedness for Viet-

namese and evaluate computational models on these datasets. This strategy also allows us to observe the behaviour of these computational models when they are applied to measure semantic similarity and relatedness in a low-resource language.

1.3 Contributions

The main objective of my thesis research is the improvement of computational models that distinguish antonymy, synonymy, and hypernymy; and that measure semantic similarity and relatedness. Moreover, the focus of my thesis is also shifted to evaluate such computational models with the low-resource Vietnamese rather than English. Therefore the contributions of my thesis are summarised as follows:

Vector representations for distinguishing antonymy and synonymy: I investigated two approaches to improve both distributional and distributed word vector representations for distinguishing antonymy and synonymy (Nguyen et al., 2016a). In the first approach, I proposed a novel weighted feature for distributional vector representations of words by using the distributional lexical contrast obtained from external resources. The distributional vector representations with novel weighted feature showed an improvement in differentiating antonymy from synonymy compared to standard weighted features. In the second approach, I proposed a novel model that integrates the distributional lexical contrast into the Skip-gram model (Mikolov et al., 2013a) to learn the new word embeddings. These word embeddings outperformed the state-of-the-art word embeddings on distinguishing antonymy from synonymy, and on measuring degrees of word similarity.

Pattern-based neural networks for distinguishing antonymy and synonymy: In this contribution, I investigated two pattern-based neural networks to distinguish antonymy from synonymy (Nguyen et al., 2017b). In both models, I induced new lexico-syntactic patterns of antonymous and synonymous word pairs obtained from the syntactic parse tree of sentences. The induced patterns differ from prior standard patterns of antonymy and synonymy in terms of the features of the lexico-syntactic pattern, in which I proposed to include the *distance* feature. As the lexico-syntactic patterns were induced from sentences containing antonymous and synonymous word pairs, they mitigated the sparsity of prior lexico-syntactic patterns. As a result, the two neural networks based on lexico-syntactic patterns showed an improvement in distinguishing antonymy from synonymy over the existing pattern-based models.

Hierarchical embeddings for hypernymy: While the previous contributions were investigated to solve the task of distinguishing antonymy and synonymy, this contribution focused on addressing tasks concerning hypernymy, which are hypernymy detection and

directionality. Specifically, I proposed a novel neural model to learn hierarchical embeddings for hypernymy (Nguyen et al., 2017a). The hierarchical embeddings are learned according to two crucial aspects (i) that the similarity score for hypernymy is higher than the similarity score for other relations; and (ii) that distributional hierarchy differentiates between hyponyms and hypernyms. As a result, while previous embedding models were not sufficient to address either hypernymy detection or directionality tasks, the hierarchical embeddings achieved a significant improvement compared to state-of-the-art measures and previous embedding models in both unsupervised and supervised tasks of hypernymy detection and directionality.

Word denoising embeddings: For this contribution, I introduced two neural models for unsupervised improvement of word embeddings (Nguyen et al., 2016b). The first model is referred to as complete word denoising embeddings, while the second model is described as overcomplete word denoising embeddings. Both neural models automatically filter out noise from word embeddings to generate word denoising embeddings. The underlying idea in the two neural models is to use a filter to learn the denoising matrix without using any external resources. To the best of my knowledge, these two neural models are the first models working on filtering out noise from word embeddings. Consequently, given state-of-the-art word embeddings as inputs of two neural models, a series of experiments showed that word denoising embeddings outperformed the given word embeddings on several benchmark tasks of semantic similarity and semantic relatedness.

Evaluating semantic models in Vietnamese: This contribution shifts to evaluate computational models on Vietnamese, a low-resource language. For this purpose, I introduced two novel datasets of (dis-)similarity and relatedness for Vietnamese (Nguyen et al., 2018). The first dataset consists of lexical contrast word pairs, aiming to distinguish between similarity and dissimilarity. The second dataset contains annotated pairs of semantic relations, reflecting the continuum between similarity and relatedness. I relied on computational models that measure semantic similarity and semantic relatedness to verify these two datasets. As a result, the main findings of this contribution are twofold: (i) two Vietnamese datasets are introduced to verify computational models for evaluating (dis-)similarity and relatedness; (ii) computational models show similar behaviours in the two Vietnamese datasets as in the corresponding English datasets.

1.4 Outline

The remainder of this thesis is structured as follows:

Chapter 2 provides some background on the concepts explored in this thesis. The first section introduces the phenomenon of semantic relations, which is the focus of this thesis. The following two sections describe in more detail the approaches to word vector representations, including distributional vector representations and distributed vector representations. The final section briefs on some details about architectures of neural networks.

Chapter 3 describes two approaches to improve both distributional and distributed vector representations of words. The first approach presents a novel weighted feature that is used to improve distributional vector representations for distinguishing antonymy and synonymy. The second approach focuses on presenting a novel neural model to learn word embeddings that significantly improve upon state-of-the-art word embeddings in terms of predicting degrees of similarity and distinguishing antonymy from synonymy.

Chapter 4 introduces two novel pattern-based neural networks to distinguish antonymy from synonymy. The lexico-syntactic patterns of antonymous and synonymous word pairs are induced from the syntactic parse trees. The first pattern-based neural network encodes lexico-syntactic patterns as vector representations, which are fed into a classifier to distinguish antonymy and synonymy. The second pattern-based neural network also represents lexico-syntactic patterns as vector representations. However, in addition to using the resulting vector representations of lexico-syntactic patterns, this neural model takes into account the concatenation of both vector representations of antonymous and synonymous words. The concatenation of the three vector representations is then used to classify antonymy and synonymy.

Chapter 5 presents a novel neural model to learn hierarchical embeddings for hypernymy. The hierarchical embeddings aim to solve the two tasks concerning hypernymy, which are hypernymy detection and directionality. Therefore, the hierarchical embeddings are trained to handle two aspects (i) that strengthens similarity of hypernymy to be higher than similarity of other relations; and (ii) that generates distributional hierarchy between hyponyms and hypernyms. These two aspects of hierarchical embeddings are then utilised to address the two tasks of hypernymy.

Chapter 6 describes two novel neural models to improve word embeddings. The two neural models learn word denoising embeddings by filtering out noise from original state-of-the-art word embeddings. The first neural model generates complete word denoising embeddings whose dimensions are equal to those of the original word embeddings. The second neural model generates overcomplete word denoising embeddings whose dimensions exceed those of the original word embeddings. In order to compare them to the

state-of-the-art word embeddings, the word denoising embeddings are evaluated on several benchmark tasks of similarity and relatedness.

Chapter 7 introduces two novel datasets of (dis-)similarity and relatedness for Vietnamese. The first dataset contains synonymous and antonymous word pairs across noun, verb, and adjective classes. It aims to distinguish between similarity and dissimilarity in Vietnamese. The second dataset comprises 400 rated word pairs across the three main word classes and five semantic relations. This dataset offers data to reflect the continuum between similarity and relatedness.

Chapter 8 summarises the main findings and results of this thesis, and outlines ideas for future work.

2 Background

From a computational point of view, distinguishing antonymy, synonymy, and hypernymy is addressed by modeling those semantic relations in vector spaces or neural networks. While vector spaces in both distributional and distributed vector representations offer a means of representing the meaning of words and determining the semantic relations between them, neural-based methods distinguish semantic relations by exploiting both semantic and syntactic information between word pairs to model those relations. In this chapter, we present an overview of semantic relations and the approaches used to model semantic relations. This chapter first introduces the phenomena of synonymy, antonymy, and hypernymy (Section 2.1). Section 2.2 and Section 2.3 then illustrate the background of distributional and distributed vector representations, respectively. Section 2.4 elucidates some basics of neural networks used in our approaches.

2.1 Semantic Relations

Semantic relations between words have been the focus of research in various fields such as philosophy, cognitive psychology, linguistics, literary theory, and computer science. In the literature, the term "semantic relations" is occasionally used to refer to phrasal or sentential relations such as paraphrase, entailment, and contradiction, but in the scope of this dissertation, it is used to denote semantic relations among words.

Among others, semantic relations such as antonymy, synonymy, and hypernymy are central to the organisation of the mental lexicon (Deese, 1965; Miller and Fellbaum, 1991; Murphy, 2003) where these semantic relations provide a structure for the lexical concepts that words express. Specifically, antonymy plays a central role in organising the adjective lexicon and provides a crucial structure in the mental lexicon for verbs. In contrast, hypernymy plays a role as a natural relation for organising the noun lexicon, and has a minor importance for verbs and unnatural for adjectives. In the following, we briefly describe these semantic relations.

2.1.1 Synonymy

The synonym relation is a relation between two words that map to the same meaning (Murphy, 2003). When a single word is replaced by its synonym in a sentence, the literal

meaning of the sentence is not changed. For example, the nouns *mother, mum* and *mom* are synonyms of each other, thus the literal meaning of the two following sentences is the same: "*His mother gave me an apple*" and "*His mom gave me an apple*". There are two kinds of synonymy: full synonymy and near-synonymy.

Full synonymy is a relation of two words identical in every sense. In natural language, candidates for full synonymy tend to be words with relatively limited number of senses, such as the full synonym of *groundhog* and *woodchuck*. Moreover, Clark (1992) conducted research utilising the principle of contrast, namely that "every two forms contrast in meaning," to show that language works to eliminate full synonymy. This means that full synonymy is limited mostly to dialectal variation and technical terms.

Near-synonymy are words that are close in meaning, or are very similar but not identical. In language, near-synonymy has no senses that are exactly the same, but each member of a near-synonym pair has a sense which is similar to the sense of its counterpart. Therefore, something described by a member of a near-synonym pair can be described by the other; i.e, near-synonyms like $foggy \approx misty$ or $mob \approx crowd$. Near-synonymy thus is what we often find in thesauri, and is usually what people mean when they use the term *synonymy* (Murphy, 2003). For instance, *lie, falsehood, untruth, fib* and *misrepresentation* are near-synonyms of each other. These words all refer to a statement that does not comfort to the truth but they differ from one another in the fine aspects of their denotation.

2.1.2 Antonymy

The term "antonymy" was coined in 1867 by C.J.Smith as the opposite of "synonymy". Since 1867, many attempts have been made to define antonymy, with the caveat that definitions of antonymy tend towards illustration rather than description. Specifically, it is more effective to explain what antonymy is by providing some examples such as *hot/cold, buy/sell, bad/good, young/old*, rather than give a definition of antonymy. Lyons (1977) defined *antonyms* as words opposite in meaning and *antonymy* as the oppositeness between words. In general, the term "antonymy" can be understood in two senses: a broad and a narrow sense. In the broad sense, antonymy includes a wide range of word pairs that are the opposite of each other and are expressed by different word classes. In the narrow sense, antonymy is a relation that holds between a small number of adjective pairs that are the opposite of each other. In this thesis, we refer to "antonymy" in its broad sense, in which any word pair that is opposite in meaning is called an antonymous pair.

Theoretical research has further categorised antonymy into two major classes, which are gradable and non-gradable antonymy. Non-gradable antonymy is a term that refers to antonymous word pairs such as *man/woman, alive/dead, active/passive*. Thus, any

antonymous word pair that does not admit a midpoint is categorised as non-gradable antonymy. For instance, in the antonymous pair *male/female*, there is no midpoint between the absolute *male* and *female* extremes. In contrast, the more common gradable antonymy, such as *hot/cold, good/bad*, exists on a scale that contains a midpoint. Considering the antonymous pair *hot/cold*, "hot" and "cold" are two points in the dimension of "temperature" with a midpoint like "tepid" that is neither "hot" nor "cold" but somewhere in between. As a result, gradable antonymy opens up possibilities for comparison in a particular dimension.

2.1.3 Hypernymy

Hypernymy also referred to as "Is-a" is an asymmetric relation that differs from symmetric relations such as synonymy and antonymy. Hypernymy holds the relationship between a generic term, a "hypernym" (or *superodinate*), and a specific instance of it, a "hyponym" (or *subordinate*). In other words, a hypernym is a word or phrase whose semantic field is more abstract than that of its hyponym. For example, *animal - cat* and *bird - eagle* are hypernymy where "animal" is a hypernym of "cat", and "eagle" is a hyponym of "bird".

Being part of an asymmetric relation, a hyponym necessarily implies its hypernym but not vice versa. For example, while "wheels" is a part of "bicycle," it does not make sense to say that "bicycle is part of wheels." Nevertheless, hypernymy is a transitive relation in which a particular word can be both a hypernym and a hyponym. In particular, if **X** is a hyponym of **Y**, and **Y** is a hyponym of **Z**, then **X** is a hyponym of **Z**. For instance, since "eagle" is a hyponym of "bird" and "bird" is a hyponym of "animal," "eagle" is a hyponym of "animal."

2.2 Distributional Word Vector Representations

Vectors are common in the areas of computer and cognitive science, and had been used before the Vector Space Model (VSM) introduced by Salton et al. (1975). The novelty of VSM was to make use of frequencies of text in a corpus as a clue to discover semantic information. Salton et al. (1975) focused on measuring document similarity by constructing a term-document matrix in which the row vectors of the matrix correspond to the terms (usually terms are the words in the documents) and the column vectors of the matrix stand for the documents. In a term-document matrix, each vector of the document and query is represented as a bag of words in which the weights of the vector indicate the frequency of the words in the bag. Figure 2.1 shows an example of the structure of the term-document matrix. Hence, the relevance of a document to a query can be estimated through the similarity of two corresponding vectors. Moreover, the similarity between two column vectors is also used to estimate the relevance between

two corresponding documents in the term-document matrix.

	Doc1	Doc2	Doc3	Doc4	Doc5
England	1	0	5	2	0
Germany	2	0	4	3	7
football	6	1	2	1	5
cricket	2	0	1	7	4
India	0	5	1	1	4
sport	4	2	1	0	3
play	3	1	6	1	0

Figure 2.1: An example of the term-document matrix where the rows indicate the terms, the columns stand for the documents, and the values are the frequencies of terms occurring in the documents.

Deerwester et al. (1990) proposed to measure word similarity instead of document similarity by considering the row vectors rather than the column vectors in the term-document matrix. Their reasoning was that words (or terms) are similar if they tend to occur in the same documents. Furthermore, the distributional hypothesis (Harris, 1954) claimed that words that occur in similar contexts tend to have similar meanings. In general, a word can be represented by a vector whose each element is derived from the co-occurrence of the word in various contexts such as words, phrases, sentences, documents, patterns, paragraphs, or sequences of characters. Not unlike document similarity, the similarity between two words can be estimated across the similarity of two row vectors in a term-context matrix whose given row indicates the vector of the corresponding word (or term) and whose given column represents the context. Therefore, each row vector of the term-context matrix represents the vector of a unique word in the lexicon. Figure 2.2 illustrates an example of the term-context matrix in which contexts (in the columns of the term-context matrix) are derived corresponding to the target words (in the rows of the term-context matrix) from the sentences.

2.2.1 Context

As discussed above, a word can be encoded as a vector in which the weights of the vector are derived from various contexts, such as documents (Salton et al., 1975), contextual windows of words (Lund and Burgess, 1996; Turney and Pantel, 2010), grammatical dependencies (Lin, 1998; Padó and Lapata, 2007), and patterns (Lin and Pantel, 2001). Once the contexts are considered as documents, the vectors of similar words tend to represent similar topics. For example, the vectors *cricket* and *football* are close to each other, because *cricket* and *football* often occur in documents related to the topic of *sport*. Although using documents as contexts can help generate words that belong to the same topic, the similar meaning of those words is problematic.

In order to represent vectors of words with similar meaning, we can narrow the context down to the contextual windows of words where the words in a fixed-size window co-occur on both sides of the target word[1]. The intuition behind this approach is that words with similar meaning, such as *car* and *bus*, tend not to occur in the same sentence (or even document). Therefore, in the term-context matrix, the row vectors of words with similar meaning are close to each other in the VSM. For example, in Figure 2.2, the vectors of *football* and *basketball* are similar to each other because both *football* and *basketball* co-occur in the contexts of *team*, *sport*, and *player*.

Football is a family of team sports that involve, to varying degrees, kicking a ball with a foot to score a goal. The various codes of football share certain common elements: Players in American football, Canadian football, rugby union and rugby league take up positions in a limited area of the field at the start of the game.
Basketball is a limited-contact sport played on a rectangular court. While most basketball players often played as a team sport with five players on each side, three-on-three, two-on-two, and one-on-one competitions are also common.

	team	sport	player	family	goal	court
football	1	1	2	1	1	0
basketball	1	2	1	0	0	1
American	0	0	1	0	0	0
Canadian	0	0	1	0	0	0
game	0	1	0	0	0	1
score	0	0	1	0	1	0
play	1	2	2	0	0	1

Figure 2.2: Simple example of the term-context matrix where the rows indicate the terms, the columns stand for the contexts, and the values are the frequencies of terms occurring in the contexts.

2.2.2 Window-size

In order to structure the term-context matrix, each target word is represented by contexts that co-occur with the target word within a certain contextual window-size. A window-size is a window of contextual words that co-occur with a certain target word in a given length (or *size*). The contextual window-size simply spans a number of contextual words occurring around the target word on both the left and right side of the word. Figure 2.3 illustrates an example of a window size of four contextual words where the target word *football* receives four words on each side as its contexts.

[1]The target word refers to the word where we are constructing the corresponding vector in the term-context matrix.

Recent studies in vector semantic models have found that window-size impacts a particular task in various ways. Among others, Hill et al. (2013) investigated the impact of window-size on measuring the similarity of concrete and abstract nouns. They found that smaller window-sizes work best for measuring concrete nouns, whereas larger window-sizes are better for measuring the similarity of abstract nouns. Schulte im Walde et al. (2013) claimed that a larger window-size worked well for measuring compositionality on the dataset of German noun-noun compounds. Kiela and Clark (2014) conducted a systematic study on training parameters for semantic space models. The authors experimented with window-sizes of 3, 5, 7, 9 and a full sentence. Their experiments in a variety of similarity tasks showed that the highest similarity score is typically achieved with a large corpora and a small window-size.

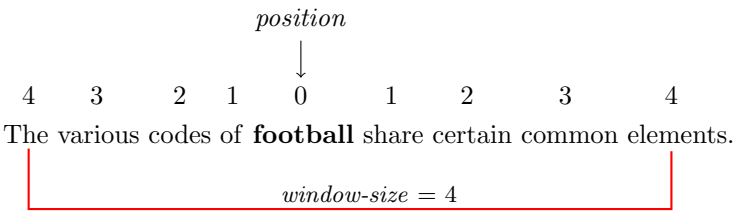


Figure 2.3: An example of a window-size with the target word at position 0 and four contextual words occurring on each side of the target word.

2.2.3 Weighting

The term-context matrix in Figure 2.2 represented each cell of the matrix by the frequency of the co-occurrence of terms (target words) and contexts. However, a problem arises with high frequency contexts, which tend to co-occur with many different target words. It turns out that such high frequency contexts can affect the discrimination of different kinds of target words. For example, contexts like *a, the, is* or *it* that occur frequently with all kinds of words like *tomato, computer* or *information*, are not able to discriminate between *tomato* and *computer*. Ideally, contexts that are particularly informative should indicate the frequency of two target words co-occuring above chance.

Pointwise mutual information (PMI): is a weighting or measure of association between words. PMI was proposed by Church and Hanks (1990), based on the motivation of mutual information. Given the target word w and the context c, the PMI association value between w and c is computed as follows:

$$PMI(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)} \tag{2.1}$$

where $P(w,c)$ denotes the joint probability between the target word w and context c; and $P(w)$ stands for the marginal probability of the target word w. The numerator

of Equation 2.1 observes how often target word w co-occurs with context c. The denominator of Equation 2.1 expresses the expectation of target word w and context c in co-occurring with each other. PMI values range from negative to positive infinity. Negative values, however, tend to express unreliable information between target words and contexts because they imply observations that co-occur less often than would be expected by chance. In order to mitigate this issue, it is reasonable to use positive PMI (PPMI), which replaces all negative values by zero. As a result, the PPMI value is defined as follows:

$$PPMI(w, c) = \max(0, \log_2 \frac{P(w, c)}{P(w)P(c)}) \tag{2.2}$$

Local Mutual Information (LMI): is an extension of the pointwise mutual information measure from the information theory (Evert, 2005). In comparison to PMI, LMI improves the problem of propagating low-frequent events by taking into account the multiplication of observed frequency and mutual information. The LMI association value between target word w and context c is formulated as follows:

$$LMI(w, c) = P(w, c) \times \log_2 \frac{P(w, c)}{P(w)P(c)} \tag{2.3}$$

2.2.4 Measuring Similarity

To measure similarity between two target words u and v in the term-context matrix, we need to take the two corresponding row vectors of u and v and then compute the similarity between two such vectors. There are various similarity measures such as cosine, Jaccard (Jaccard, 1912), and Kullback-Leibler divergence (Kullback and Leibler, 1951). By far the most common metric used to compute the similarity between two vectors is the cosine measure. Therefore, in this thesis we mainly focus on using the cosine similarity to measure the similarity between two target words. The cosine similarity metric between two vectors $\vec{u}$ and $\vec{v}$ is defined as follows:

$$cosine(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}||\vec{v}|} \tag{2.4}$$

where $\vec{u} \cdot \vec{v}$ denotes the dot product (or *inner product*) between two vectors $\vec{u}$ and $\vec{v}$; and $|\vec{u}|$ indicates the vector length (or *vector norm*). The dot product between two vectors tends to receive the high value when the two vectors have large values in the same dimensions. The dot product between two vectors $(\vec{u}, \vec{v})$ and the vector length of $\vec{u}$ are defined as in equations 2.5 and 2.6, respectively:

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{N} u_i v_i \tag{2.5}$$

$$|\vec{u}| = \sqrt{\sum_{i=1}^{N} u_i^2} \tag{2.6}$$

By combining all three equations 2.4, 2.5, and 2.6 together, the cosine similarity measure can be computed as in equation 2.7:

$$cosine(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}||\vec{v}|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\sum_{i=1}^{N} u_i^2}\sqrt{\sum_{i=1}^{N} v_i^2}} \tag{2.7}$$

2.3 Distributed Word Vector Representations

Distributed word vector representations (often called *word embeddings*) are a means to represent words in vector space so that they are embedded as low-dimensional dense vectors. The term *word embeddings* was originally coined by Bengio et al. (2003), who used word embeddings trained in a neural language model together with the model's parameters. The power of word embeddings was first demonstrated by Collobert and Weston (2008), whose models treated word embeddings as a highly effective tool when used in a downstream task. Subsequently, word embeddings gained increased popularity with Mikolov et al. (2013a) who proposed two models to learn word embeddings effectively. The following year, Pennington et al. (2014) introduced GloVe, an approach to learn competitive word embeddings.

Word embeddings can be considered to be among a small number of successful applications of unsupervised learning so far. The advantage of word embeddings is that they are trained on unannotated corpora, helping us to obtain word embeddings in many different languages easily. Moreover, although word embeddings are encoded in low-dimensional vector representations, the information regarding word similarity is still retained. The pre-trained embeddings can be used in downstream tasks that use a small amount of labeled data.

In general, every feed-forward neural network that takes words from the vocabulary as inputs embeds those words as vectors into a low-dimensional space. The feed-forward neural model then fine-tunes their parameters through back-propagation to yield word embeddings that are weights of the first layer (usually referred to as *Embedding layer*). However, learning word embeddings through such neural models is simply too computationally expensive for a large vocabulary. Mathematically speaking, consider a training corpus containing a sequence of $\mathcal{T}$ training words $w_1, w_2, ..., w_T$ that belong to a vocabulary $\mathcal{V}$ whose size is $|\mathcal{V}|$. Embedding models generally consider a context of n words to train a target word. Each word in the vocabulary $\mathcal{V}$ is associated with an input embed-

ding v_w with d dimensions, and with an output embedding v'_w. Embedding models then optimize an objective function $\mathcal{J}_\theta$ with respect to parameters θ, and the models output the score $f_\theta(w)$ for every target word w.

2.3.1 Classic Language Model

Word embedding models are closely related to language models in the fact that the quality of language models is measured based on their ability to learn a probability distribution over the vocabulary $\mathcal{V}$. Similarly, embedding models also try to predict the next word in a sequence. Among others, the classic neural language model (Bengio et al., 2003) consists of input, projection, hidden and output layers. The architecture of the classic language model is illustrated in Figure 2.4.

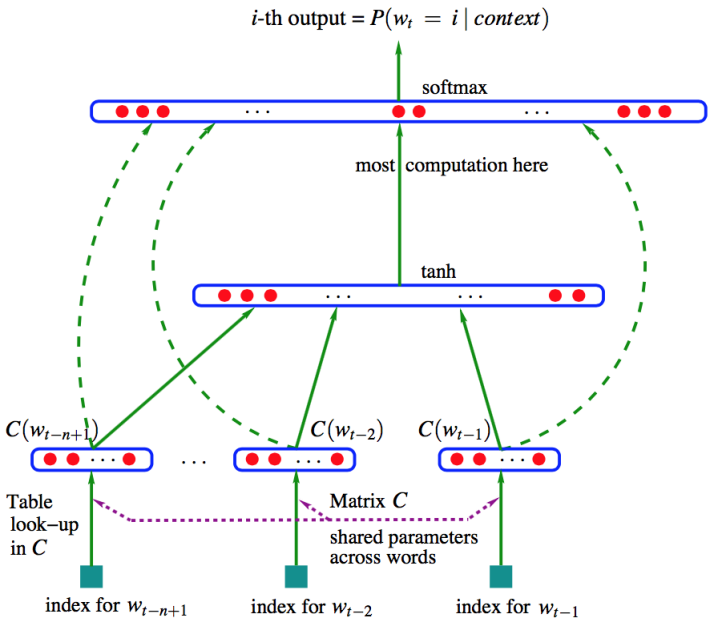


Figure 2.4: The architecture of the classic language model (Bengio et al., 2003)

The model is then trained to maximize the average of log probabilities of all words in the vocabulary $\mathcal{V}$ given their previous n words as follows:

$$J_\theta = \frac{1}{T}\sum_{1}^{T} \log p(w_t|w_{t-1}, ..., w_{t-n+1}) \tag{2.8}$$

the probability $\log p(w_t|w_{t-1}, ..., w_{t-n+1})$ can be computed by using the softmax layer and the output vector of a hidden layer h as in equation 2.9:

$$\log p(w_t|w_{t-1}, ..., w_{t-n+1}) = \frac{\exp(h^T v'_{w_t})}{\sum_{w_i \in V} \exp(h^T v'_{w_t})} \tag{2.9}$$

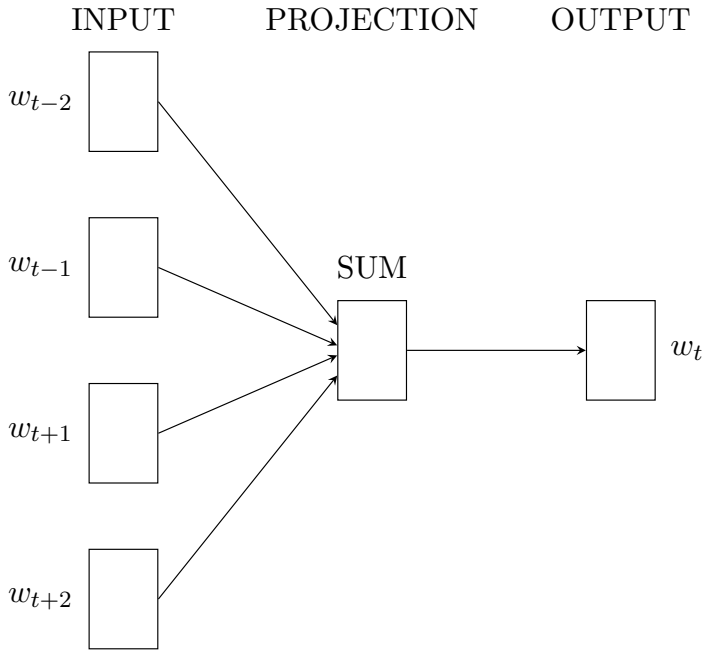


Figure 2.5: The CBOW architecture that predicts the target word based on the contexts.

The architecture of such language model becomes complex for computation between projection and hidden layers over the vocabulary because values in the projection are dense. Moreover, finding ways to mitigate the computational cost associated with the softmax layer is one of the key challenges in learning word embeddings effectively.

2.3.2 Word2vec Embedding Models

This section introduces word2vec models that were proposed by Mikolov et al. (2013a). Word2vec models are arguably the most popular word embedding models and offer two main benefits over previous embedding models: they do away with the expensive hidden layer and enable the language model to take additional context into account. Word2vec models consist of two neural architectures: continuous bag-of-word and continuous skip-gram models.

Continuous Bag-of-Word Model

The continuous bag-of-word model (CBOW) is similar to the classic neural model (Bengio et al., 2003); however, the non-linear hidden layer is removed and the projection layer is shared for all words in the vocabulary. While the classic language model is only able to look at the previous words for the prediction of the target word, CBOW model uses n words both before and after the target word w_t to predict it as depicted in Figure 2.5.

The objective function of CBOW model is slightly different compared to the objective

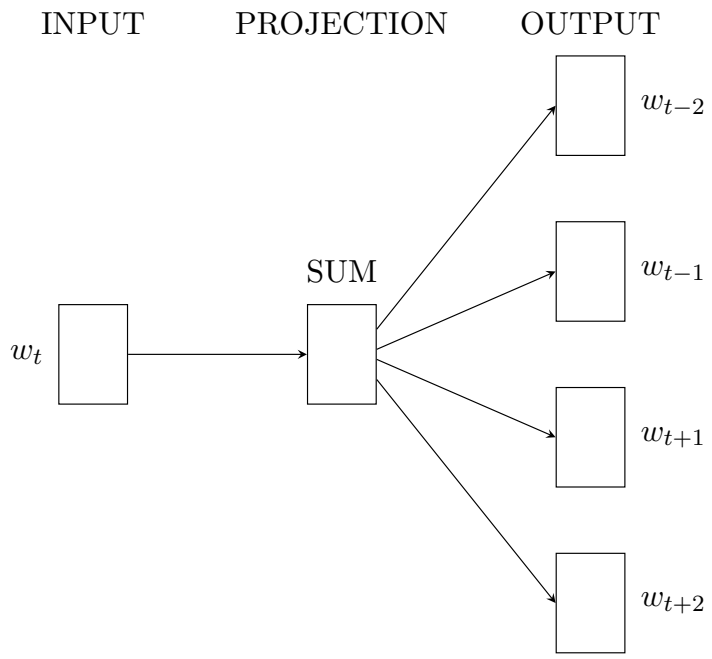


Figure 2.6: The skip-gram architecture that treats the target word as an input to predict words around the target word.

function of the classic language model, defined as follows:

$$J_\theta = \frac{1}{T}\sum_{1}^{T} \log p(w_t|w_{t-n}, ..., w_{t-1}, w_{t+1}, ..., w_{t+n}) \tag{2.10}$$

Instead of feeding n previous words into the model to train the target word, CBOW model considers a window of n words around the target word w_t at each time step t.

Continuous Skip-gram Model

The continuous skip-gram model is similar to the CBOW model; however, while the CBOW model predicts the target word based on the contexts around it, the skip-gram model tries to maximize classification of a target word based on another word in the same sentence. Specifically, each target word is treated as an input to a log-linear classifier with a continuous projection layer. The target word is then used to predict words within a certain window-size before and after the target word. Figure 2.6 depicts the architecture of the skip-gram model.

The skip-gram objective function thus sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t as the following objective function:

$$J_\theta = \frac{1}{T}\sum_{1}^{T} \sum_{-n \leq j \leq n} \log p(w_{t+j}|w_t) \tag{2.11}$$

Since the skip-gram architecture removes the hidden layer that produces an output

vector h as in the architecture of the classic language model, the output vector h of the skip-gram model is simply the word embedding $\vec{v}_{w_t}$ of the input word w_t. Thus the probabilities $\log p(w_{t+j}|w_t)$ can be computed as follows:

$$\log p(w_{t+j}|w_t) = \frac{\exp(v_{w_t}^T v'_{w_{t+j}})}{\sum_{w_i \in V} \exp(v_{w_t}^T v'_{w_i})} \tag{2.12}$$

Negative Sampling

The output of embedding models can be computed by using a softmax layer, but the complexity of computing the final softmax layer is too expensive. Thus mitigating such complexity has been one of the main challenges in training word embeddings. In the last decade, different strategies have been proposed to approximate the softmax such as hierarchical softmax (Morin and Bengio, 2005), CNN-softmax (Kim et al., 2016), differentiated softmax (Chen et al., 2016), and noise contrastive estimation (Gutmann and Hyvärinen, 2010).

Mikolov et al. (2013c) proposed negative sampling to train the skip-gram model so that it can be seen as an approximation of noise contrastive estimation. While noise contrastive estimation can be shown to approximate the loss of the softmax layer as the number of samples k increases, the negative sampling approach is used simply to learn high quality word embeddings. Negative sampling can be defined by the following objective function:

$$\log \theta({v'_{w_i}}^T v_{w_i}) + \sum_{j=1}^{k} \mathbb{E}_{w_i} \sim P_n(w)[\log \theta(-{v'_{w_j}}^T v_{w_i})] \tag{2.13}$$

where k is the number of negative samples; $P_n(w)$ is noise distribution with respect to the target word w.

2.4 Neural Networks

Neural networks (NNs) are computational models that are inspired by biology of the human brain. NNs contain a large number of neurons (i.e., computational nodes) that are trained to map the inputs to the outputs through mapping functions. NNs usually consist of several layers including an input layer, one or several hidden layers, and an output layer. Each layer of NNs contains multiple computational neurons. In the architecture of NNs, the input layer is in charge of receiving the input signals from the training data. The hidden layers are responsible for computing and transforming input signals into representations of training data. The output layer then transforms representations of hidden layers to the particular output format.

In recent years, approaches based on NNs have gained impressive achievements in solving tasks in NLP. In the following, we discuss how neural networks are used to deal with NLP tasks. Specifically, we introduce the input layer, certain kinds of hidden layers, and the output layer.

2.4.1 Input Layer

The input layer in NNs is used to represent basic units of training data under suitable formats providing for hidden layers of NNs. In the NLP task, there are various kinds of basic units that the input layer presents, such as words (Kim, 2014), characters (Kim et al., 2016), morphemes (Botha and Blunsom, 2014), patterns (Shwartz et al., 2016), and even sentences (Kiros et al., 2015). More specifically, each unit will be assigned to a unique index, which is then mapped to a corresponding distributed vector representation, as discussed in Section 2.3. This procedure is carried out for all units in the vocabulary of training data. The resulting vector representations of units are stored in a matrix called the *lookup table*. For example, if the dimension of a vector representation that encodes units of training data is d and the vocabulary size of units is V, the dimension of a look table will be $V \times d$.

In NLP tasks, input data are often fed into NNs through sequences of basic units that reserve information such as semantics, syntax, and ordering of units. Given a sequential input $x_1 x_2 ... x_k$ where x_i is a unit of the input sequence and k is the length of the input sequence, we encode each unit x_i with a d-dimensional vector representation $x_i \in \mathbb{R}^d$. Thus, the representation of a sequential input will be a matrix $\mathbf{X} \in \mathbb{R}^{k \times d}$ as follows:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1d} \\ x_{21} & x_{22} & x_{23} & \dots & x_{1d} \\ x_{31} & x_{32} & x_{33} & \dots & x_{1d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & x_{k3} & \dots & x_{kd} \end{bmatrix} \tag{2.14}$$

where the row i of matrix $\mathbf{X}$ represents the vector representation of unit ith in the input sequence.

2.4.2 Hidden Layer

The hidden layer is the most important layer in the architecture of NNs where representations of the input layer will be computed and transformed to the output layer. The architecture of the hidden layer can be typical, such as feed-forward, recurrent, and convolution; or a combination of different architectures. The names of some typical NNs are followed by integrating the corresponding typical hidden architectures. In this section,

we briefly introduce feed-forward and recurrent neural networks in which feed-forward and recurrent architectures are considered hidden layers.

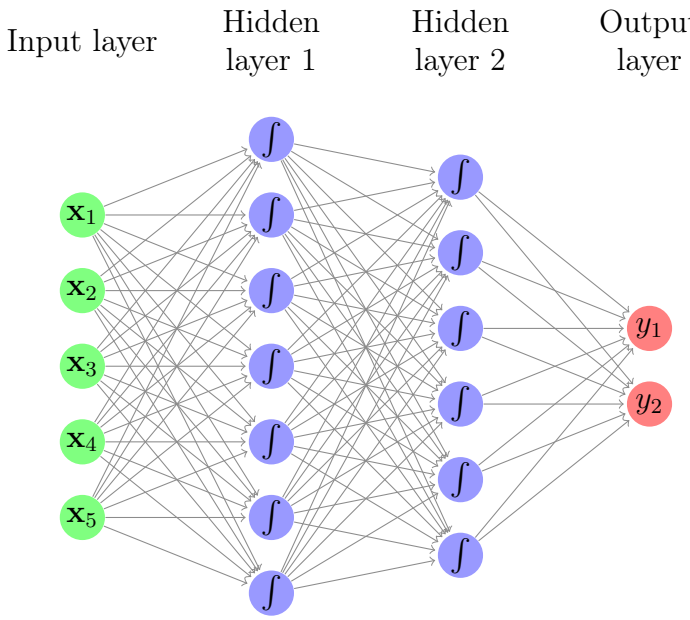


Figure 2.7: A fully connected feed-forward neural network with two fully connected hidden layers followed by an output layer. The input units are represented by vectors $\mathbf{x}_i$.

Feed-forward Neural Network

The feed-forward neural network consists of at least three layers including an input layer, a hidden layer, and an output layer. The total number of layers determines the depth of the NN. In the feed-forward neural network, there is no "feedback" of the output layer toward the input layer throughout the hidden layers. That is why we call it the *feed-forward neural network*. Figure 2.7 depicts a full connected feed-forward neural network with an input layer, two hidden layers, and an output layer. In this neural network, the input layer has five input units, the first and second hidden layers contain seven and six neurons respectively, and there are two output neurons in the output layer. Each neuron of the two hidden layers is computed by using a linear transformation as follows:

$$\mathbf{h}_i = f(\mathbf{W}_i \mathbf{h}_{i-1}) \tag{2.15}$$

where $f(\cdot)$ is a non-linear activation function applied to the linear transformation between the previous hidden unit $\mathbf{h}_{i-1}$ and the matrix $\mathbf{W}_i$. The non-linear activation function can be tanh, sigmoid, or ReLu (rectified linear unit) function.

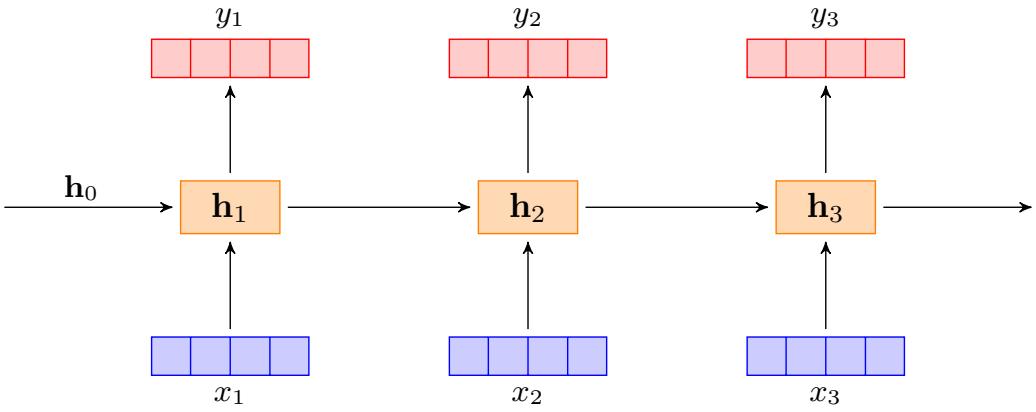


Figure 2.8: An example of a recurrent neural network where $\mathbf{h}_t$ is updated by considering both input x_t and previous hidden state $\mathbf{h}_{t-1}$.

Recurrent Neural Network

The recurrent neural network (RNN) is also called the Elman network (Elman, 1990). It receives the previous hidden state and the current input as inputs of the current hidden state. The RNN can thus store information about the past inputs for a time. Specifically, given a sequence of k words $\mathbf{X} = [x_1, x_2, ..., x_k]$ as input data, the RNN processes each word x_t at a time and returns a vector of state h_k for the complete input sequence. For each time step t, the RNN updates an internal memory state h_t that depends on the current input x_t and the previous state h_{t-1} as follows:

$$\mathbf{h}_t = \sigma(\mathbf{V}x_t + \mathbf{U}h_{t-1} + b) \tag{2.16}$$

where $\mathbf{V}$ and $\mathbf{U}$ are learnable matrices; b is a bias; $\mathbf{h}_0$ is assigned to zero at the time step $t = 0$. Figure 2.8 illustrates an example of the RNN.

Yet, if the sequential input is a long-term dependency, an RNN faces the problem of gradient vanishing or exploding, leading to difficulties in training the model. Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) units were proposed to address these problems. The underlying idea of the LSTM is to use an adaptive gating mechanism to decide on the extent to which LSTM units keep the previous state and memorize the extracted features of the current input. More specifically, the LSTM is comprised of four components: an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t. The state of the LSTM at each time step t is formalized as follows:

$$\begin{aligned}
i_t &= \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \\
f_t &= \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \\
o_t &= \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \\
g_t &= \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \\
c_t &= i_t \otimes g_t + f_t \otimes c_{t-1}
\end{aligned}$$

W refers to a matrix of weights that projects information between two layers; b is a layer-specific vector of bias terms; σ denotes the sigmoid function. The output of an LSTM at a time step t is computed as follows:

$$h_t = o_t \otimes \tanh(c_t)$$

where $\otimes$ denotes element-wise multiplication.

2.4.3 Output Layer

In a neural network, the output layer is responsible for generating a prediction for output variables based on the input variables and hidden states of the neural model. Figure 2.9 shows an example of an output layer. For a particular task, the number of output variables depends on the number of classes, such as binary or multiple classes. Each unit of an output layer represents a score or probability prediction of a neural network for each corresponding class. Since the number of units in the last hidden layer is often larger than the number of classes, there is usually a full connected layer to transform the size of the last hidden layer to the size of classes. Thus, the output layer can be defined as follows:

$$\hat{\mathbf{y}} = f(\mathbf{W}_o \mathbf{h}) \tag{2.17}$$

in which $\mathbf{h}$ is the state of the last hidden layer. $\mathbf{W}_o$ stands for the weight matrix between the last hidden layer and the output layer. $f(\cdot)$ is the activation function where f is often the sigmoid function for binary classification or the softmax function for multiple classification. $\hat{y}$ accounts for the prediction of the corresponding class.

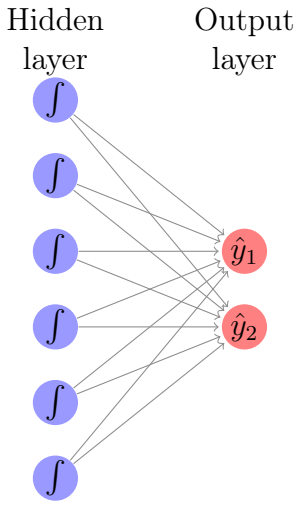


Figure 2.9: An example of an output layer with two units.

2.4.4 Training

In order to train the neural network, we need to minimize the predictions of the model with the gold labels over all classes. To do so, we rely on loss functions to minimize the difference between the predictions and the gold labels across classes of the output. There are several common loss functions such as cross entropy loss, logistic loss, and mean squared error loss. The cross entropy loss function is often chosen to deal with the classification task. The cross entropy loss function can be defined as follows:

$$-\sum_{i}^{|y|}\Big(\mathbf{y}_i \log \hat{\mathbf{y}}_i + (1-\mathbf{y}_i)\log(1-\hat{\mathbf{y}}_i)\Big) \tag{2.18}$$

where $|y|$ is the number of classes. $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ are the gold label and the prediction of class ith, respectively.

To minimize the loss function, stochastic gradient descent algorithms are applied to optimize the loss function by updating the parameters of the neural network. There are several common stochastic gradient descent algorithms such as AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), and ADAM (Kingma and Ba, 2014).

3 Distinguishing Antonymy and Synonymy with Vector Representations

3.1 Introduction

Antonymy and synonymy represent lexical semantic relations that are central to the organization of the mental lexicon (Miller and Fellbaum, 1991). While antonymy is defined as the oppositeness between words, synonymy refers to words that are similar in meaning (Deese, 1965; Lyons, 1977). From a computational point of view, distinguishing between antonymy and synonymy is important for NLP applications such as machine translation, coreference resolution, and recognizing textual entailment, which go beyond the general notion of semantic relatedness and require identification of the specific semantic relations. However, due to interchangeable substitution, antonyms and synonyms often occur in similar contexts. For example, considering the sentences "*the girl loves the cat*" and "*the girl hates the cat*," the antonymous pair *love/hate* co-occurs with the same contexts in the two sentences. As a result, the vector of "love" is close in word vector representations to the vector of "hate." This issue has been challenging automatic systems in distinguishing between antonymy and synonymy.

Numerous approaches have been proposed to distinguish antonymy from synonymy. Among others, approaches based on word vector representations have received many impressive results in dealing with this task. Two common approaches that represent words as vectors are distributional and distributed vector representations. They both rely on the distributional hypothesis (Harris, 1954; Firth, 1957), which states that words with similar distributions are related in meaning. In the distributional vector representation, each word is represented by a weighted feature vector, whose features typically correspond to words that co-occur in a particular context. Moreover, features are often weighted by applying several weighting methods, such as PMI, PPMI or LMI. However, since all these weighting methods are based on normalizing the co-occurrence of words and their contexts, using distributional vector representations tends to retrieve both synonyms (such as *formal–conventional*) and antonyms (such as *formal–informal*) as related words, and therefore cannot sufficiently distinguish between the two relations. Unlike

distributional vector representations, approaches based on distributed vector representations represent words as low-dimensional dense vector representations. However, the motivation behind distributed vector representations is still based on the distributional hypothesis, in which vector representation of a word is constructed from its contexts. Consequently, this approach is still not sufficient to handle the issue of distinguishing antonymy from synonymy.

In this chapter, we present two approaches that improve both distributional and distributed vector representations for antonym–synonym discrimination. The first approach is motivated by the fact that feature overlap in synonyms is supposedly stronger than feature overlap in antonyms. Therefore, in this approach, we incorporate lexical contrast into distributional vectors and strengthen those word features that are most salient to determine word similarities or dissimilarities. To establish the most salient features of antonyms and synonyms, we exploit the difference between the distribution of antonyms and synonyms.

Regarding the second approach, lexical contrast information is integrated into the objective function of the skip-gram model with negative sampling (SGNS) described by Mikolov et al. (2013a). The proposed model optimizes semantic vectors to predict degrees of word similarity and to distinguish antonyms from synonyms. The resulting vectors make vector representations of synonymy closer to each other, while forcing vector representations of antonymy to be further away from each other. The improved word embeddings outperform state-of-the-art models on antonym–synonym distinction and the word similarity task. The approaches and experiments described in this chapter are published in Nguyen et al. (2016a).

3.2 Related Work

In recent years, a number of distributional approaches have accepted the challenge to distinguish antonyms from synonyms, often in combination with lexical resources such as thesauruses or taxonomies. For example, Lin et al. (2003) used dependency triples to extract distributionally similar words, and then in a post-processing step filtered out words that appeared with the patterns 'from X to Y' or 'either X or Y' significantly often. Mohammad et al. (2013) assumed that word pairs that occur in the same thesaurus category are close in meaning and marked as synonyms, while word pairs occurring in contrasting thesaurus categories or paragraphs are marked as opposites. Scheible et al. (2013) showed that the distributional difference between antonyms and synonyms can be identified via a simple word space model by using appropriate features. Santus et al. (2014a) and Santus et al. (2014b) aimed to identify the most salient dimensions of meaning in vector representations and reported a new average-precision-based distributional measure and an entropy-based measure to discriminate antonyms from synonyms (and

further paradigmatic semantic relations).

Lately, antonym–synonym distinction has also been a focus of word embedding models. For example, Adel and Schütze (2014) integrated coreference chains extracted from large corpora into a skip-gram model to create word embeddings that identified antonyms. Ono et al. (2015) proposed thesaurus-based word embeddings to capture antonyms. They proposed two models: the WE-T model that trains word embeddings on thesaurus information; and the WE-TD model that incorporated distributional information into the WE-T model. Pham et al. (2015) introduced the multitask lexical contrast model (mLCM) by incorporating WordNet into a skip-gram model to optimize semantic vectors to predict contexts. Their model outperformed standard skip-gram models with negative sampling on both general semantic tasks and distinguishing antonyms from synonyms.

3.3 Approach

In this section, we present the two contributions: a new distributional vector representation that improves the quality of weighted features to distinguish between antonyms and synonyms (Section 3.3.1), and a novel extension of skip-gram models that integrates the distributional lexical contrast into the objective function of skip-gram model, in order to predict similarities between words and to identify antonyms (Section 3.3.2).

3.3.1 Improving the weights of feature vectors

We aim to improve the quality of weighted feature vectors by strengthening those features that are most salient in the vectors and by putting less emphasis on those that are of minor importance, when distinguishing degrees of similarity between words. We start out with standard corpus co-occurrence frequencies and apply LMI measure to determine the original strengths of the word features. Our score $weight^{SA}(w, f)$ subsequently defines the weights of a target word w and a feature f as follows:

$$\begin{aligned} weight^{SA}(w, f) = & \frac{1}{\#(w, u)} \sum\nolimits_{u \in W(f) \cap S(w)} sim(w, u) \\ & - \frac{1}{\#(w', v)} \sum_{w' \in A(w)} \sum\nolimits_{v \in W(f) \cap S(w')} sim(w', v) \end{aligned} \tag{3.1}$$

The new $weight^{SA}$ scores of a target word w and a feature f exploit the differences between the average similarities of synonyms to the target word ($sim(w, u)$, with $u \in S(w)$), and the average similarities between antonyms of the target word ($sim(w', v)$, with $w' \in A(w)$ and $v \in S(w')$). Only those words u and v are included in the calculation that have a positive original LMI score for the feature f: $W(f)$. To calculate the similarity sim between two word vectors, we rely on cosine distances. If a word w is not associated

with any synonyms or antonyms in our resources (cf. Section 3.4.1), or if a feature does not co-occur with a word w, we define $weight^{SA}(w, f) = 0$.

The intuition behind the *lexical contrast information* in our new $weight^{SA}$ is as follows. The strongest features of a word also tend to represent strong features of its synonyms, but weaker features of its antonyms. For example, the feature *conception* only occurs with synonyms of the adjective *formal* but not with the antonym *informal*, or with synonyms of the antonym *informal*. $weight^{SA}(formal, conception)$, which is calculated as the average similarity between *formal* and its synonyms minus the average similarity between *informal* and its synonyms, should thus return a high positive value. In contrast, a feature such as *issue* that occurs with many different adjectives, would enforce a feature score near zero for $weight^{SA}(formal, issue)$, because the similarity scores between *formal* and its synonyms and *informal* and its synonyms should not differ strongly. Last but not least, a feature such as *rumor* that only occurs with *informal* and its synonyms, but not with the original target adjective *formal* and its synonyms, should invoke a very low value for $weight^{SA}(formal, rumor)$. Figure 3.1 provides a schematic visualization for computing the new $weight^{SA}$ scores for the target *formal*.

Since the number of antonyms is usually much smaller than the number of synonyms, we enrich the number of antonyms: Instead of using the direct antonym links, we consider all synonyms of an antonym $w' \in A(w)$ as antonyms of w. For example, the target word *good* has only two antonyms in WordNet (*bad* and *evil*), in comparison to 31 synonyms. Thus, we also exploit the synonyms of *bad* and *evil* as antonyms for *good*.

3.3.2 Integrating the distributional lexical contrast into word embeddings

Our model relies on Levy and Goldberg (Levy and Goldberg, 2014) who showed that the objective function for a SGNS can be defined as follows:

$$\sum_{w \in V} \sum_{c \in V} \{\#(w, c) \log \sigma(sim(w, c)) + k\#(w) P_0(c) \log \sigma(-sim(w, c))\} \tag{3.2}$$

The first term in Equation (3.2) represents the co-occurrence between a target word w and a context c within a context window. The number of appearances of the target word and that context is defined as $\#(w, c)$. The second term refers to the negative sampling where k is the number of negatively sampled words, and $\#(w)$ is the number of appearances of w as a target word in the unigram distribution P_0 of its negative context c.

To incorporate our lexical contrast information into the SGNS model, we propose the objective function in Equation (3.3) to add distributional contrast followed by all contexts of the target word. V is the vocabulary; $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid func-

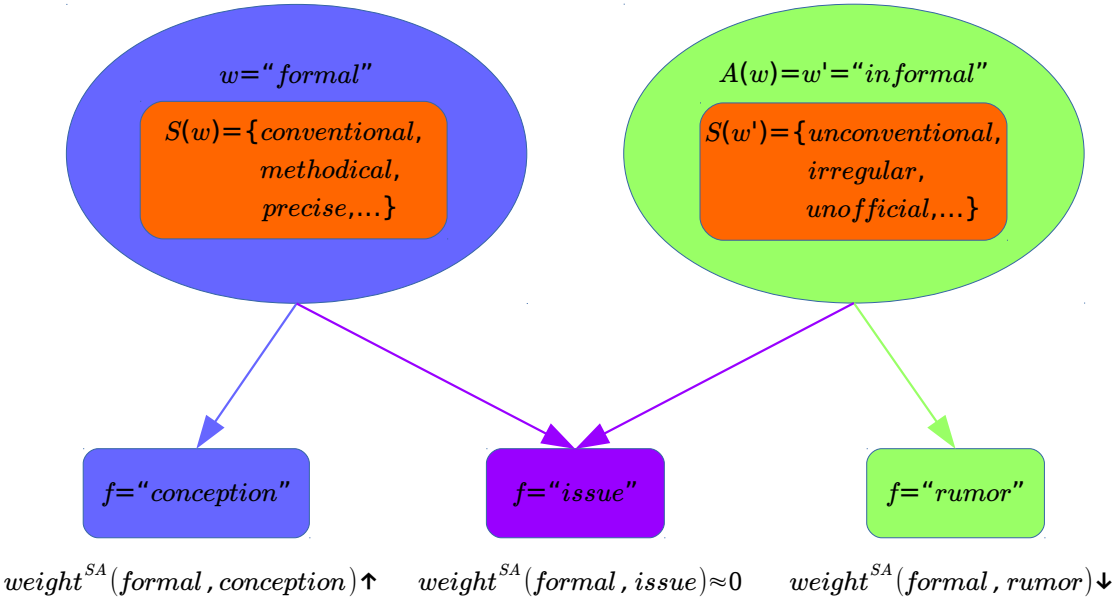


Figure 3.1: Illustration of the $weight^{SA}$ scores for the adjective target *formal*. The feature *conception* only occurs with *formal* and synonyms of *formal*, so $weight^{SA}(formal, conception)$ should return a positive value; the feature *rumor* only occurs with the antonym *informal* and with synonyms of *informal*, so $weight^{SA}(formal, rumor)$ should return a negative value; the feature *issue* occurs with both *formal* and *informal* and also with synonyms of these two adjectives, so $weight^{SA}(formal, issue)$ should return a feature score near zero.

tion; and $sim(w_1, w_2)$ is the cosine similarity between the two embedded vectors of the corresponding two words w_1 and w_2. We refer to our distributional lexical-contrast embeddings model as *dLCE*.

$$\sum_{w \in V} \sum_{c \in V} \{(\#(w,c) \log \sigma(sim(w,c)) + k\#(w)P_0(c) \log \sigma(-sim(w,c))) \\ + (\frac{1}{\#(w,u)} \sum\nolimits_{u \in W(c) \cap S(w)} sim(w,u) - \frac{1}{\#(w,v)} \sum\nolimits_{v \in W(c) \cap A(w)} sim(w,v))\} \tag{3.3}$$

Equation (3.3) integrates the lexical contrast information in a slightly different way compared to Equation (3.1): For each of the target words w, we only rely on its antonyms $A(w)$ instead of using the synonyms of its antonyms $S(w')$. This makes the word embeddings training more efficient in running time, especially since we are using a large amount of training data.

The dLCE model is similar to the WE-TD model (Ono et al., 2015) and the mLCM model (Pham et al., 2015); however, while the WE-TD and mLCM models only apply the lexical contrast information from WordNet to each of the target words, dLCE applies lexical contrast to every single context of a target word in order to better capture and classify semantic contrast.

	Adjectives		Nouns		Verbs	
	ANT	SYN	ANT	SYN	ANT	SYN
LMI	0.46	0.56	0.42	0.60	0.42	0.62
$weight^{SA}$	**0.36****	**0.75****	**0.40**	**0.66**	**0.38***	**0.71***
LMI + SVD	0.46	0.55	0.46	0.55	0.44	0.58
$weight^{SA}$ + SVD	**0.36*****	**0.76*****	**0.40***	**0.66***	**0.38*****	**0.70*****

Table 3.1: AP evaluation on DSMs.

3.4 Experiments

3.4.1 Experimental Settings

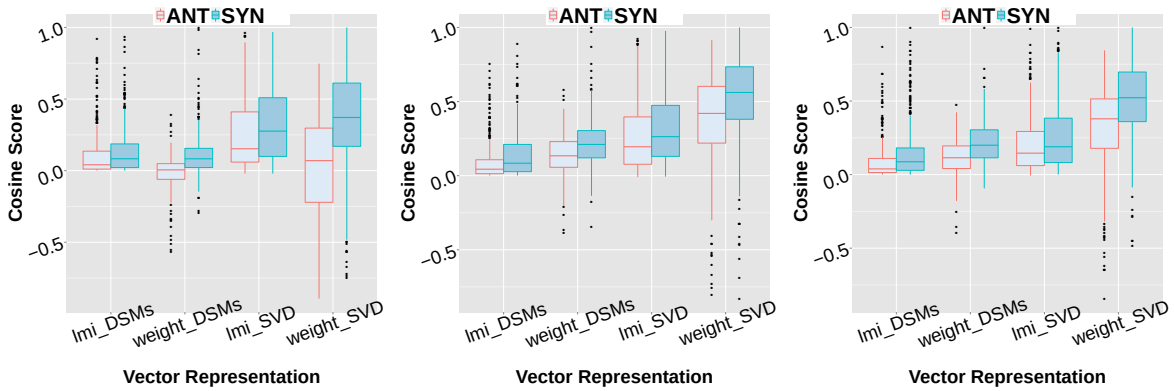
The corpus resource for our vector representations is one of the currently largest web corpora[1]: *ENCOW14A* (Schäfer and Bildhauer, 2012; Schäfer, 2015), containing approximately 14.5 billion tokens and 561K distinct word types. As distributional information, we used a window size of 5 tokens for both the original vector representation and the word embeddings models. For word embeddings models, we trained word vectors with 500 dimensions; k negative sampling was set to 15; the threshold for sub-sampling was set to 10^{-5}; and we ignored all words that occurred < 100 times in the corpus. The parameters of the models were estimated by backpropagation of error via stochastic gradient descent. The learning rate strategy was similar to Mikolov et al. (2013a) in which the initial learning rate was set to 0.025. For the lexical contrast information, we used WordNet (Miller, 1995) and Wordnik[2] to collect antonyms and synonyms, obtaining a total of 363,309 synonym and 38,423 antonym pairs.

3.4.2 Distinguishing antonyms from synonyms

The first experiment evaluates our lexical contrast vectors by applying the vector representations with the improved $weight^{SA}$ scores to the task of distinguishing antonyms from synonyms. As gold standard resource, we used the English dataset described in Roth and Schulte im Walde (2014), containing 600 adjective pairs (300 antonymous pairs and 300 synonymous pairs), 700 noun pairs (350 antonymous pairs and 350 synonymous pairs) and 800 verb pairs (400 antonymous pairs and 400 synonymous pairs). For evaluation, we applied Average Precision (AP) (Voorhees and Harman, 1999), a common metric in information retrieval previously used by Kotlerman et al. (2010) and Santus et al. (2014a), among others.

Table 3.1 presents the results of the first experiment, comparing our improved vector representations with the original LMI representations across word classes, without/with applying singular-value decomposition (SVD), respectively. In order to evaluate the

[1] `http://corporafromtheweb.org/`
[2] `http://www.wordnik.com`



(a) Adjective pairs (b) Noun pairs (c) Verb pairs

Figure 3.2: Differences between cosine scores for antonymous vs. synonymous word pairs.

distribution of word pairs with AP, we sorted the synonymous and antonymous pairs by their cosine scores. A synonymous pair was considered correct if it belonged to the first half; and an antonymous pairs was considered correct if it was in the second half. The optimal results would thus achieve an AP score of 1 for SYN and 0 for ANT. The results in the tables demonstrate that $weight^{SA}$ significantly[3] outperforms the original vector representations across word classes.

In addition, Figure 3.2 compares the medians of cosine similarities between antonymous pairs (red) vs. synonymous pairs (green) across word classes, and for the four conditions (1) LMI, (2) $weight^{SA}$, (3) SVD on LMI, and (4) SVD on $weight^{SA}$. The plots show that the cosine similarities of the two relations differ more strongly with our improved vector representations in comparison to the original LMI representations, and even more so after applying SVD.

3.4.3 Effects of distributional lexical contrast on word embeddings

The second experiment evaluates the performance of our dLCE model on both antonym–synonym distinction and a word similarity task. The idea is to examine the effect of the lexical contrast information also on other important word embeddings tasks such as word similarity. The similarity task requires to predict the degree of similarity for word pairs, and the ranked list of predictions is evaluated against a gold standard of human ratings, relying on the Spearman rank-order correlation coefficient ρ (Siegel and Castellan, 1988).

In this experiment, we use the *SimLex-999* dataset (Hill et al., 2015) to evaluate word embedding models on predicting similarities. The resource contains 999 word pairs (666 noun, 222 verb and 111 adjective pairs) and was explicitly built to test models on capturing similarity rather than relatedness or association. Table 3.2 shows that

[3] χ^2, $^{***}p < .001$, $^{**}p < .005$, $^{*}p < .05$

SGNS	mLCM	dLCE
0.38	0.51	**0.59**

Table 3.2: Spearman's ρ on SimLex-999.

	Adjectives	Nouns	Verbs
SGNS	0.64	0.66	0.65
mLCM	0.85	0.69	0.71
dLCE	**0.90**	**0.72**	**0.81**

Table 3.3: AUC scores for identifying antonyms.

our dLCE model outperforms both SGNS and mLCM, proving that the lexical contrast information has a positive effect on predicting similarity.

Therefore, the improved distinction between synonyms (strongly similar words) and antonyms (often strongly related but highly dissimilar words) in the dLCE model also supports the distinction between degrees of similarity.

For distinguishing between antonyms and synonyms, we computed the cosine similarities between word pairs on the dataset described in Section 3.4.2, and then used the area under curve (AUC) to evaluate the performance of dLCE compared to SGNS and mLCM. The results in Table 3.3 report that dLCE outperforms SGNS and mLCM also on this task.

3.5 Summary

This chapter introduced two approaches to represent words as vector representations, which are distributional vector representation and word embeddings. Both approaches are able to enhance predictions of word similarity. Firstly, we significantly improved the quality of weighted features in distributional vector representations to distinguish antonyms from synonyms by using lexical contrast information. The proposed weighted feature is able to distinguish between antonyms and synonyms by exploiting the distinctive contexts of antonyms and synonyms. Relying on the lexical contrast information, there are three kinds of contexts among antonyms and synonyms: (i) the distinctive contexts of synonyms, which only occur with synonymous word pairs; (ii) the distinctive contexts of antonyms, which only occur with antonymous word pairs; and (iii) the normal contexts, which co-occur with both synonymous and antonymous word pairs. The experimental results showed that the proposed weighted feature significantly outperforms other weighted features for the antonymy-synonymy distinction task.

Secondly, we incorporated the lexical contrast information into a skip-gram model to to learn word embeddings. The new word embeddings successfully predicted degrees of similarity and identified antonyms. In comparison to other word embeddings, the

new word embeddings are able to decrease the distance between word embeddings of synonymy and to increase the distance between word embeddings of antonymy. Two experiments on similarity and antonymy-synonymy distinction tasks proved that the proposed embeddings outperform state-of-the-art word embeddings.

4 Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network

4.1 Introduction

Distinguishing between antonyms and synonyms is a key task to achieve high performance in NLP systems. While this task is notoriously difficult to achieve with the use of distributional co-occurrence models, pattern-based methods have proven effective to differentiate between the relations. In this chapter, we focus on distinguishing antonyms and synonym by using a neural network to exploit lexico-syntactic patterns of antonymous and synonymous pairs from syntactic parse trees. From a computational point of view, two families of approaches to differentiate between antonyms and synonyms are predominant in NLP: co-occurrence models and pattern-based models. Both of these make use of vector representations, relying on the *distributional hypothesis* (Harris, 1954; Firth, 1957), which states that words with similar distributions have related meanings. These models offer a means to represent meaning vectors of words or word pairs, and to determine their semantic relatedness (Turney and Pantel, 2010).

In *co-occurrence models*, each word is represented by a weighted feature vector, where features typically correspond to words that co-occur in particular contexts. When using word embeddings, these models rely on neural methods to represent words as low-dimensional vectors. To create the word embeddings, the models either make use of neural-based techniques, such as the skip-gram model (Mikolov et al., 2013a), or use matrix factorization (Pennington et al., 2014) that builds word embeddings by factorizing word-context co-occurrence matrices. In comparison to standard co-occurrence vector representations, word embeddings address the problematic sparsity of word vectors and have achieved impressive results in many NLP tasks such as word similarity (e.g., Pennington et al. (2014)), relation classification (e.g., Vu et al. (2016)), and antonym-synonym distinction (e.g., Nguyen et al. (2016a)).

In *pattern-based models*, vector representations make use of lexico-syntactic surface patterns to distinguish between the relations of word pairs. For example, Justeson and Katz (1991) suggested that adjectival opposites co-occur with each other in specific lin-

ear sequences, such as `between X and Y`. Hearst (1992) determined surface patterns, e.g., `X such as Y`, to identify nominal hypernyms. Lin et al. (2003) proposed two textual patterns indicating semantic incompatibility, `from X to Y` and `either X or Y`, to distinguish opposites from semantically similar words. Roth and Schulte im Walde (2014) proposed a method that combined patterns with discourse markers for classifying paradigmatic relations including antonymy, synonymy, and hypernymy. Recently, Schwartz et al. (2015) used two prominent patterns from Lin et al. (2003) to learn word embeddings that distinguished antonyms from similar words in determining degrees of similarity and word analogy.

Despite the pattern-based models' ability to mitigate the interchangeable substitution of antonymy and synonymy, these models still face the issue of sparsity of patterns since typical lexico-syntactic patterns cannot cover all antonymous and synonymous pairs. For example, in the sentence *"My old village has been provided with the new services,"* the antonymous pair *old–new* cannot be derived from any typical patterns. In this chapter, in order to deal with the sparsity of patterns, we present a novel pattern-based neural method *AntSynNET* to distinguish antonyms from synonyms. We hypothesize that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than would be expected of synonymous pairs. This hypothesis is inspired by corpus-based studies on antonymy and synonymy. Among others, Charles and Miller (1989) suggested that adjectival opposites co-occur in patterns; Fellbaum (1995) stated that nominal and verbal opposites co-occur in the same sentence significantly more often than chance; Lin et al. (2003) argued that if two words appear in the clear antonym patterns, they are unlikely to represent a synonymous pair.

We start out by inducing patterns between `X` and `Y` from a large-scale web corpus, where `X` and `Y` represent two words of an antonym or synonym word pair, and the pattern is derived from the simple paths between `X` and `Y` in a syntactic parse tree. Each node in the simple path combines lexical and syntactic information; in addition, we suggest a novel feature for the patterns, i.e., the distance between the two words along the syntactic path. All pattern features are fed into a recurrent neural network with LSTM units (Hochreiter and Schmidhuber, 1997), which encodes the patterns as vector representations. Afterwards, the vector representations of the patterns are used in a classifier to distinguish between antonyms and synonyms. The experimental results show that AntSynNET improves performance over prior pattern-based methods. Furthermore, the implementation of our models is publicly available[1]. The approaches and experiments described in this chapter are published in Nguyen et al. (2017b).

[1] `https://github.com/nguyenkh/AntSynNET`

4.2 Related Work

Pattern-based methods: Regarding the task of antonym-synonym distinction, there exist a variety of approaches which rely on patterns. Lin et al. (2003) used bilingual dependency triples and patterns to extract distributionally similar words. They relied on clear antonym patterns such as `from X to Y` and `either X or Y` in a post-processing step to distinguish antonyms from synonyms. The main idea is that if two words `X` and `Y` appear in one of these patterns, they are unlikely to represent synonymous pair. Schulte im Walde and Köper (2013) proposed a method to distinguish between the paradigmatic relations antonymy, synonymy and hypernymy in German, based on automatically acquired word patterns. Roth and Schulte im Walde (2014) combined general lexico-syntactic patterns with discourse markers as indicators for the same relations, both for German and for English. They assumed that if two phrases frequently co-occur with a specific discourse marker, then the discourse relation expressed by the corresponding marker should also indicate the relation between the words in the affected phrases. By using the raw corpus and a fixed list of discourse markers, the model can easily be extended to other languages. More recently, Schwartz et al. (2015) presented a symmetric pattern-based model for word vector representation in which antonyms are assigned to dissimilar vector representations. Differently to the previous pattern-based methods which used the standard distribution of patterns, Schwartz et al. used patterns to learn word embeddings.

Vector representation methods: Yih et al. (2012) introduced a new vector representation where antonyms lie on opposite sides of a sphere. They derived this representation with the incorporation of a thesaurus and latent semantic analysis, by assigning signs to the entries in the co-occurrence matrix on which latent semantic analysis operates, such that synonyms would tend to have positive cosine similarities, and antonyms would tend to have negative cosine similarities. Scheible et al. (2013) showed that the distributional difference between antonyms and synonyms can be identified via a simple word space model by using appropriate features. Instead of taking into account all words in a window of a certain size for feature extraction, the authors experimented with only words of a certain part-of-speech, and restricted distributions. Santus et al. (2014a) proposed a different method to distinguish antonyms from synonyms by identifying the most salient dimensions of meaning in vector representations and reporting a new average-precision-based distributional measure and an entropy-based measure. Ono et al. (2015) trained supervised word embeddings for the task of identifying antonymy. They proposed two models to learn word embeddings: the first model relied on thesaurus information; the second model made use of distributional information and thesaurus information. More recently, Nguyen et al. (2016a) proposed two methods to distinguish antonyms from syn-

onyms: in the first method, the authors improved the quality of weighted feature vectors by strengthening those features that are most salient in the vectors, and by putting less emphasis on those that are of minor importance when distinguishing degrees of similarity between words. In the second method, the lexical contrast information was integrated into the skip-gram model (Mikolov et al., 2013a) to learn word embeddings. This model successfully predicted degrees of similarity and identified antonyms and synonyms.

4.3 Approach

In this section, we describe the AntSynNET model, using a pattern-based LSTM for distinguishing antonyms from synonyms. We first present the induction of patterns from a parsed corpus (Section 4.3.1). Section 4.3.2 then describes how we utilize the recurrent neural network with long short-term memory units to encode the patterns as vector representation. Finally, we present the AntSynNET model and two approaches to classify antonyms and synonyms (Section 4.3.3).

4.3.1 Induction of Patterns

Corpus-based studies on antonymy have suggested that opposites co-occur with each other within a sentence significantly more often than would be expected by chance. Our method thus makes use of patterns as the main indicators of word pair co-occurrence, to enforce a distinction between antonyms and synonyms. Figure 5.1 shows a syntactic parse tree of the sentence *"My old village has been provided with the new services"*. Following the characterizations of a tree in graph theory, any two nodes (vertices) of a tree are connected by a simple path (or one unique path). The simple path is the shortest path between any two nodes in a tree and does not contain repeated nodes. In the example, the lexico-syntactic tree pattern of the antonymous pair *old–new* is determined by finding the simple path (in red) from the lemma `old` to the lemma `new`. It focuses on the most relevant information and ignores irrelevant information which does not appear in the simple path (i.e., *has, been*). The example pattern between `X = old` and `Y = new` in Figure 5.1 is represented as follows: `X/JJ/amod/2 - village/NN/nsubj/1 - provide/VBN/ROOT/0 - with/IN/prep/1 - service/NNS/pobj/2 - Y/JJ/amod/3`.

Node Representation: The path patterns make use of four features to represent each node in the syntax tree: lemma, part-of-speech (POS) tag, dependency label and distance label. The lemma feature captures the lexical information of words in the sentence, while the POS and dependency features capture the morpho-syntactic information of the sentence. The distance label measures the path distance between the target word nodes in the syntactic tree. Each step between a parent and a child node represents a

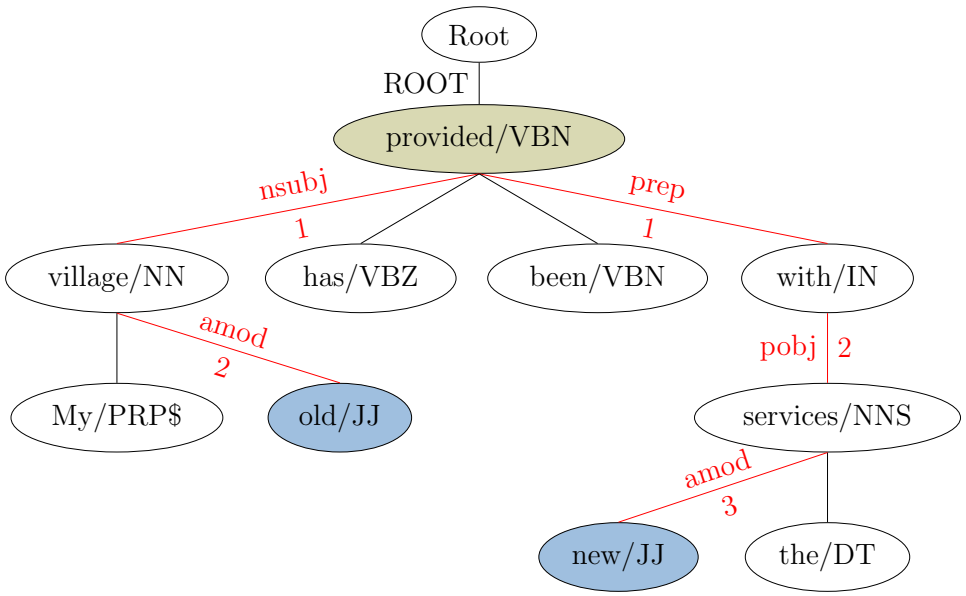


Figure 4.1: Illustration of the syntactic tree for the sentence *"My old village has been provided with the new services"*. Red lines indicate the path from the word `old` to the word `new`.

distance of 1; and the ancestor nodes of the remaining nodes in the path are represented by a distance of 0. For example, the node `provided` is an ancestor node of the simple path from `old` to `new`. The distances from the node `provided` to the nodes `village` and `old` are 1 and 2, respectively. The vector representation of each node concatenates the four-feature vectors as follows:

$$\vec{v}_{node} = [\vec{v}_{lemma} \oplus \vec{v}_{pos} \oplus \vec{v}_{dep} \oplus \vec{v}_{dist}]$$

where $\vec{v}_{lemma}, \vec{v}_{pos}, \vec{v}_{dep}, \vec{v}_{dist}$ represent the embeddings of the lemma, POS tag, dependency label and distance label, respectively; and the $\oplus$ denotes the concatenation operation.

Pattern Representation: For a pattern p which is constructed by the sequence of nodes $n_1, n_2, ..., n_k$, the pattern representation of p is a sequence of vectors:

$$p = [\vec{n}_1, \vec{n}_2, ..., \vec{n}_k]$$

The pattern vector $\vec{v}_p$ is then encoded by applying a recurrent neural network.

4.3.2 Recurrent Neural Network with Long Short-Term Memory Units

A recurrent neural network is suitable for modeling sequential data by a vector representation. In our methods, we use a long short-term memory network, a variant of a

recurrent neural network to encode patterns. Given a pattern of nodes $p = [n_1, n_2, ..., n_k]$ as input data, the long short-term memory network represents each node via output and hidden state vector representations.

In our methods, we rely on the last state h_k to represent the vector $\vec{v}_p$ of a pattern $p = [\vec{n}_1, \vec{n}_2, ..., \vec{n}_k]$. The $\vec{v}_p$ is then used for the classifier.

4.3.3 The Proposed AntSynNET Model

In this section, we present two models to distinguish antonyms from synonyms. The first model makes use of patterns to classify antonyms and synonyms, by using an LSTM to encode patterns as vector representations and then feeding those vectors to a logistic regression layer (Section 4.3.3). The second model creates combined vector representations of word pairs, which concatenate the vectors of the words and the patterns (Section 4.3.3).

Pattern-based AntSynNET

In this model, we make use of a recurrent neural network with LSTM units to encode patterns containing a sequence of nodes. Figure 4.2 illustrates the AntSynNET model. Given a word pair (x, y), we induce patterns for (x, y) from a corpus, where each pattern represents a path from x to y (cf. Section 4.3.1). We then feed each pattern p of the word pair (x, y) into an LSTM to obtain $\vec{v}_p$, the vector representation of the pattern p (cf. Section 4.3.2). For each word pair (x, y), the vector representation of (x, y) is computed as follows:

$$\vec{v}_{xy} = \frac{\sum_{p \in P(x,y)} \vec{v}_p \cdot c_p}{\sum_{p \in P(x,y)} c_p} \tag{4.1}$$

$\vec{v}_{xy}$ refers to the vector of the word pair (x, y); $P(x, y)$ is the set of patterns corresponding to the pair (x, y); c_p is the frequency of the pattern p. The vector $\vec{v}_{xy}$ is then fed into a logistic regression layer whose target is the class label associated with the pair (x, y). Finally, the pair (x, y) is predicted as positive (i.e., antonymous) word pair if the probability of the prediction for $\vec{v}_{xy}$ is larger than 0.5.

Combined AntSynNET

Inspired by the supervised distributional concatenation method in Baroni et al. (2012) and the integrated path-based and distributional method for hypernymy detection in Shwartz et al. (2016), we take into account the patterns and distribution of target pairs to create their combined vector representations. Figure 4.3 illustrates the combined AntSynNET model. Given a word pair (x, y), the combined vector representation of the pair (x, y) is determined by using both the co-occurrence distribution of the words and the syntactic

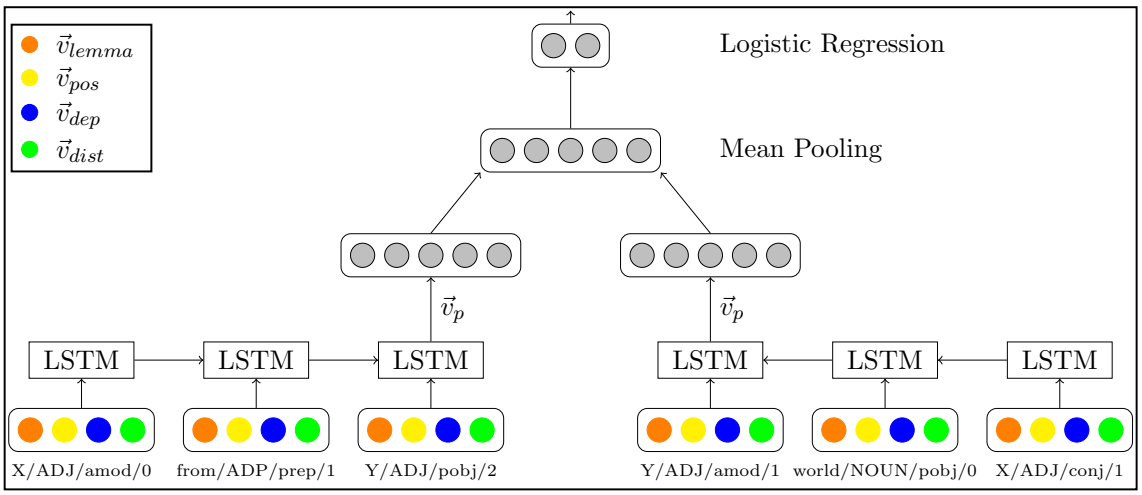


Figure 4.2: Illustration of the *AntSynNET* model. Each word pair is represented by several patterns, and each pattern represents a path in the graph of the syntactic tree. Patterns consist of several nodes where each node is represented by a vector with four features: lemma, POS, dependency label, and distance label. The mean pooling of the pattern vectors is the vector representation of each word pair, which is then fed to the logistic regression layer to classify antonyms and synonyms.

path patterns:

$$\vec{v}_{comb(x,y)} = [\vec{v}_x \oplus \vec{v}_{xy} \oplus \vec{v}_y] \tag{4.2}$$

$\vec{v}_{comb(x,y)}$ refers to the combined vector of the word pair (x, y); $\vec{v}_x$ and $\vec{v}_y$ are the vectors of word x and word y, respectively; $\vec{v}_{xy}$ is the vector of the pattern that corresponds to the pair (x, y), cf. Section 4.3.3. Similar to the pattern-based model, the combined vector $\vec{v}_{comb(x,y)}$ is fed into the logistic regression layer to classify antonyms and synonyms.

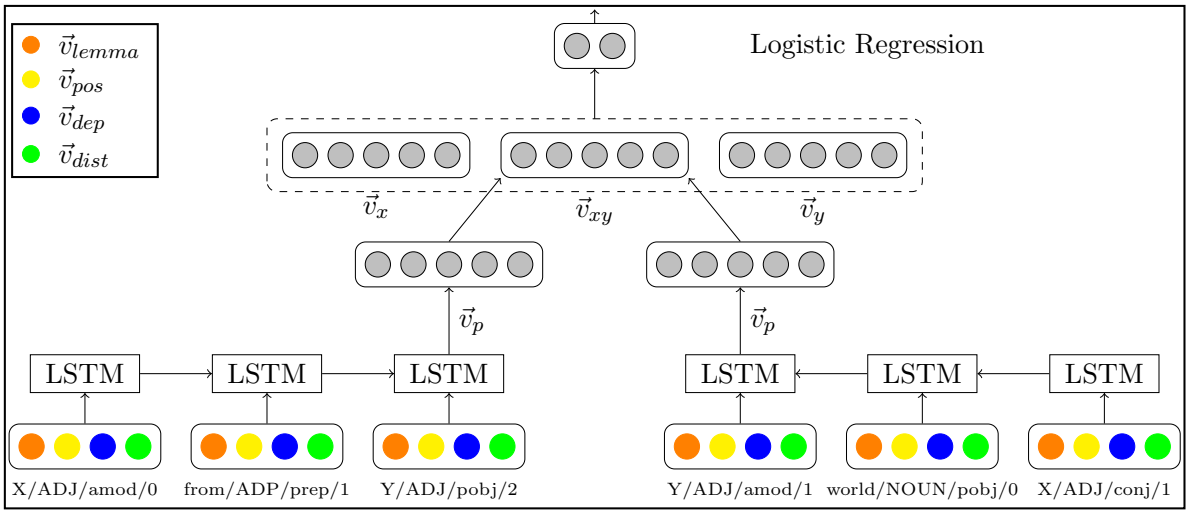


Figure 4.3: Illustration of the *combined AntSynNET* model. The concatenation of vectors $\vec{v}_x$, $\vec{v}_y$, and $\vec{v}_{xy}$ is considered as the vector representation of each word pair, and is then fed to the logistic regression layer to classify antonyms and synonyms.

4.4 Baseline Models

To compare AntSynNET with baseline models for pattern-based classification of antonyms and synonyms, we introduce two pattern-based baseline methods: the distributional method (Section 4.4.1), and the distributed method (Section 4.4.2).

4.4.1 Distributional Baseline

As a first baseline, we apply the approach by Roth and Schulte im Walde (2014), henceforth R&SiW. They used a VSM to represent pairs of words by a combination of standard lexico-syntactic patterns and discourse markers. In addition to the patterns, the discourse markers added information to express discourse relations, which in turn may indicate the specific semantic relation between the two words in a word pair. For example, contrast relations might indicate antonymy, whereas elaborations may indicate synonymy or hyponymy.

Michael Roth, the first author of R&SiW, kindly computed the relation classification results of the pattern–discourse model for our test sets. The weights between marker-based and pattern-based models were tuned on the validation sets; other hyperparameters were set exactly as described by the R&SiW method.

4.4.2 Distributed Baseline

The *SP* method proposed by Schwartz et al. (2015) uses symmetric patterns for generating word embeddings. In this work, the authors applied an unsupervised algorithm for the automatic extraction of symmetric patterns from plain text. The symmetric patterns were defined as a sequence of 3-5 tokens consisting of exactly two wildcards and 1-3 words. The patterns were filtered based on their frequencies, such that the resulting pattern set contained 11 patterns. For generating word embeddings, a matrix of co-occurrence counts between patterns and words in the vocabulary was computed, using positive point-wise mutual information. The sparsity problem of vector representations was addressed by smoothing. For antonym representation, the authors relied on two patterns suggested by Lin et al. (2003) to construct word embeddings containing an antonym parameter that can be turned on in order to represent antonyms as dissimilar, and that can be turned off to represent antonyms as similar.

To apply the SP method to our data, we make use of the pre-trained SP embeddings[2] with 500 dimensions[3]. We calculate the cosine similarity of word pairs and then use a Support Vector Machine with Radial Basis Function kernel to classify antonyms and synonyms.

[2] `http://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html`
[3] The 500-dimensional embeddings outperformed the 300-dimensional embeddings for our data.

4.5 Experiment

4.5.1 Dataset

For training the models, neural networks require a large amount of training data. We use the existing large-scale antonym and synonym pairs previously used by Nguyen et al. (2016a). Originally, the data pairs were collected from WordNet (Miller, 1995) and Wordnik[4].

Word Class	Train	Test	Validation	Total
Adjective	5562	1986	398	7946
Verb	2534	908	182	3624
Noun	2836	1020	206	4062

Table 4.1: The size of antonymous and synonymous pairs across three word classes.

In order to induce patterns for the word pairs in the dataset, we identify the sentences in the corpus that contain the word pair. Thereafter, we extract all patterns for the word pair. We filter out all patterns which occur less than five times; and we only take into account word pairs that contain at least five patterns for training, validating and testing. For the proportion of positive and negative pairs, we keep a ratio of 1:1 positive (antonym) to negative (synonym) pairs in the dataset. In order to create the sets of training, testing and validation data, we perform random splitting with 70% train, 25% test, and 5% validation sets. The final dataset contains the number of word pairs according to word classes described in Table 4.1. Moreover, Table 4.2 shows the average number of patterns for each word pair in our dataset.

Word Class	Train	Test	Validation
Adjective	135	131	141
Verb	364	332	396
Noun	110	132	105

Table 4.2: Average number of patterns per word pair across word classes.

4.5.2 Experimental Settings

We use the English Wikipedia dump[5] from June 2016 as the corpus resource for our methods and baselines. For parsing the corpus, we rely on spaCy[6]. For the lemma embeddings, we rely on the word embeddings of the dLCE model[7] (Nguyen et al., 2016a)

[4] http://www.wordnik.com
[5] https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
[6] https://spacy.io
[7] https://github.com/nguyenkh/AntSynDistinction

which is the state-of-the-art vector representation for distinguishing antonyms from synonyms. We re-implemented this cutting-edge model on Wikipedia with 100 dimensions, and then make use of the dLCE word embeddings for initialization the lemma embeddings. The embeddings of POS tags, dependency labels, distance labels, and out-of-vocabulary lemmas are initialized randomly. The number of dimensions is set to 10 for the embeddings of POS tags, dependency labels and distance labels. We use the validation sets to tune the number of dimensions for these labels. For optimization, we rely on the cross-entropy loss function and Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012). For training, we use the `Theano` framework (Theano Development Team, 2016). Regularization is applied by a dropout of 0.5 on each of component's embeddings (dropout rate is tuned on the validation set). We train the models with 40 epochs and update all embeddings during training.

4.5.3 Overall Results

Table 4.3 shows the significant[8] performance of our models in comparison to the baselines. Concerning adjectives, the two proposed models significantly outperform the two baselines: The performance of the baselines is around .72 for F_1, and the corresponding results for the combined AntSynNET model achieve an improvement of >.06. Regarding nouns, the improvement of the new methods is just .02 F_1 in comparison to the R&SiW baseline, but we achieve a much better performance in comparison to the SP baseline, an increase of .37 F_1. Regarding verbs, we do not outperform the more advanced R&SiW baseline in terms of the F_1 score, but we obtain higher recall scores. In comparison to the SP baseline, our models still show a clear F_1 improvement.

Model	Adjective			Verb			Noun		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
SP baseline	0.730	0.706	0.718	0.560	0.609	0.584	0.625	0.393	0.482
R&SiW baseline	0.717	0.717	0.717	**0.789**	0.787	**0.788**	**0.833**	0.831	0.832
Pattern-based AntSynNET	**0.764**	0.788	0.776*	0.741	**0.833**	0.784	0.804	0.851	0.827
Combined AntSynNET	0.763	**0.807**	**0.784***	0.743	0.815	0.777	0.816	**0.898**	**0.855****

Table 4.3: Performance of the AntSynNET models in comparison to the baseline models.

Overall, our proposed models achieve comparatively high recall scores compared to the two baselines. This strengthens our hypothesis that there is a higher possibility for the co-occurrence of antonymous pairs in patterns over synonymous pairs within a sentence. Because, when the proposed models obtain high recall scores, the models are able to retrieve most relevant information (antonymous pairs) corresponding to the patterns.

Regarding the low precision in the two proposed models, we sampled randomly 5 pairs in each population: true positive, true negative, false positive, false negative. We then compared the overlap of patterns for the true predictions (true positive pairs and true

[8] t-test, *$p < 0.05$, **$p < 0.1$

negative pairs) and the false predictions (false positive pairs and false negative pairs). We found out that there is no overlap between patterns of true predictions; and the number overlap between patterns of false predictions is 2, 2, and 4 patterns for noun, adjective, and verb classes, respectively. This shows that the low precision of our models stems from the patterns which represent both antonymous and synonymous pairs.

4.5.4 Effect of the Distance Feature

In our models, the novel distance feature is successfully integrated along the syntactic path to represent lexico-syntactic patterns. The intuition behind the distance feature exploits properties of trees in graph theory, which show that there exist differences in the degree of relationship between the parent node and the child nodes ($distance = 1$) and in the degree of relationship between the ancestor node and the descendant nodes ($distance > 1$). Hence, we use the distance feature to effectively capture these relationships.

Feature	Model	Adjective			Verb			Noun		
		P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Direction	Pattern-based	0.752	0.755	0.753	0.734	0.819	0.774	0.800	0.825	0.813
	Combined	0.754	0.784	0.769	0.739	0.793	0.765	**0.829**	0.810	0.819
Distance	Pattern-based	**0.764**	0.788	0.776	0.741	**0.833**	**0.784**	0.804	0.851	0.827
	Combined	0.763	**0.807**	**0.784****	**0.743**	0.815	0.777	0.816	**0.898**	**0.855****

Table 4.4: Comparing the novel distance feature with Schwarz et al.'s direction feature, across word classes.

In order to evaluate the effect of our novel distance feature, we compare the distance feature to the direction feature proposed by Shwartz et al. (2016). In their approach, the authors combined lemma, POS, dependency, and direction features for the task of hypernym detection. The direction feature represented the direction of the dependency label between two nodes in a path from `X` to `Y`.

For evaluation, we make use of the same information regarding dataset and patterns as in Section 4.5.3, and then replace the distance feature by the direction feature. The results are shown in Table 4.4. The distance feature enhances the performance of our proposed models more effectively than the direction feature does, across all word classes.

4.5.5 Effect of Word Embeddings

Our methods rely on the word embeddings of the dLCE model, state-of-the-art word embeddings for antonym-synonym distinction. Yet, the word embeddings of the dLCE model, i.e., supervised word embeddings, represent information collected from lexical resources. In order to evaluate the effect of these word embeddings on the performance

of our models, we replace them by the pre-trained GloVe word embeddings[9] with 100 dimensions, and compare the effects of the GloVe word embeddings and the dLCE word embeddings on the performance of the two proposed models.

Model	Word Embeddings	Adjective			Verb			Noun		
		P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Pattern-based Model	GloVe	0.763	0.770	0.767	0.705	0.852	0.772	0.789	0.849	0.818
	dLCE	0.764	0.788	**0.776**	0.741	0.833	**0.784**	0.804	0.851	**0.827**
Combined Model	Glove	0.750	0.798	0.773	0.717	0.826	0.768	0.807	0.827	0.817
	dLCE	0.763	0.807	**0.784**	0.743	0.815	**0.777**	0.816	0.898	**0.855**

Table 4.5: Comparing pre-trained GloVe and dLCE word embeddings.

Table 4.5 illustrates the performance of our two models on all word classes. The table shows that the dLCE word embeddings are better than the pre-trained GloVe word embeddings, by around .01 F_1 for the pattern-based AntSynNET model and the combined AntSynNET model regarding adjective and verb pairs. Regarding noun pairs, the improvements of the dLCE word embeddings over pre-trained GloVe word embeddings achieve around .01 and .04 F_1 for the pattern-based model and the combined model, respectively.

4.6 Summary

In this chapter, we presented a novel pattern-based neural model *AntSynNET* to distinguish antonyms from synonyms. Despite the success of the co-occurrence approaches introduced in chapter 3, there still exist some limitations, such as the issue of out-of-vocabulary, which prevents the lexical resources from covering all necessary words in the corpora; and the problem of low-resource languages for which lexical resources are no longer developed. Unlike these co-occurrence approaches, the *AntSynNET*, which relies on lexico-syntactic patterns, is able to mitigate these limitations by hypothesizing that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than synonymous word pairs.

Specifically, the lexico-syntactic patterns were derived from the simple paths between semantically related words in a syntactic parse tree. In addition to lexical and syntactic information, we suggested a novel path distance feature. The AntSynNET model consists of two approaches to classify antonyms and synonyms. In the first approach, we used a recurrent neural network with long short-term memory units to encode the patterns as vector representations; in the second approach, we made use of the distribution and encoded patterns of the target pairs to generate combined vector representations. The resulting vectors of patterns in both approaches were fed into the logistic regression layer for classification.

[9] http://www-nlp.stanford.edu/projects/glove/

Our proposed models significantly outperformed two baselines relying on previous work, mainly in terms of recall. Moreover, we demonstrated that the distance feature outperformed a previously suggested direction feature and that our embeddings outperformed the state-of-the-art GloVe embeddings. Finally, since our two proposed models rely only on corpus data, they are easily applicable to other languages and relations.

5 Hierarchical Embeddings for Hypernymy Detection and Directionality

5.1 Introduction

In chapters 3 and 4, we presented two approaches to distinguish antonymy and synonymy. In this chapter, we mainly focus on detecting hypernymy. In the literature on semantic relations, hypernymy is considered as a major semantic relation and a key organization principle of semantic memory (Miller and Fellbaum, 1991; Murphy, 2002). It is an asymmetric relation between two terms, a hypernym (superordinate) and a hyponym (subordinate), as in *animal–bird* and *flower–rose*, where the hyponym necessarily implies the hypernym, but not vice versa. The tasks of hypernymy detection and directionality have been a challenge for automatic systems. Specifically, the hypernymy detection task is to distinguish hypernymy from other relations such as synonymy, antonymy, meronymy, and so on. The hypernymy directionality task aims to determine which of the two words is the hypernym and which is the hyponym. From a computational point of view, automatic hypernymy detection is useful for NLP tasks such as taxonomy creation (Snow et al., 2006; Navigli et al., 2011), recognizing textual entailment (Dagan et al., 2013), and text generation (Biran and McKeown, 2013), among many others.

Two families of approaches to identify and discriminate hypernymy are predominant in NLP, both of them relying on word vector representations. ***Distributional count approaches*** make use of either directionally unsupervised measures or of supervised classification methods. Unsupervised measures exploit the *distributional inclusion hypothesis* (Geffet and Dagan, 2005; Zhitomirsky-Geffet and Dagan, 2009), or the *distributional informativeness hypothesis* (Santus et al., 2014a; Rimell, 2014). These measures assign scores to semantic relation pairs, with hypernymy scores expected to be higher than those of other relation pairs. Typically, AP measure (Kotlerman et al., 2010) is applied to rank and distinguish between the predicted relations. Supervised classification methods represent each pair of words as a single vector by using the concatenation or the element-wise difference of their vectors (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). The resulting vector is fed into a Support Vector Machine (SVM) or

into Logistic Regression (LR), to predict hypernymy. Across approaches, Shwartz et al. (2017) demonstrated that there is no single unsupervised measure that consistently deals well with discriminating hypernymy from other semantic relations. Furthermore, Levy et al. (2015) showed that supervised methods memorize *prototypical hypernyms* instead of *learning* a relation between two words.

Approaches of hypernymy-specific embeddings utilize neural models to learn vector representations for hypernymy. Yu et al. (2015) proposed a supervised method to learn term embeddings for hypernymy identification, based on pre-extracted hypernymy pairs. Recently, Tuan et al. (2016) proposed a dynamic weighting neural model to learn term embeddings, in which the model encodes not only the information of hypernyms vs. hyponyms but also their contextual information. The performance of this family of models is typically evaluated by using an SVM to discriminate hypernymy from other relations.

In this chapter, we present a novel neural model *HyperVec* to learn hierarchical embeddings that **(i)** discriminate hypernymy from other relations (**detection task**), and **(ii)** distinguish between the hypernym and the hyponym in a given hypernymy relation pair (**directionality task**). Our model learns to strengthen the distributional similarity of hypernym pairs, relative to other relation pairs, by moving hyponym and hypernym vectors close to each other. In addition, we generate a distributional hierarchy between hyponyms and hypernyms. Relying on these two new aspects of hypernymy distributions, the similarity of hypernym pairs receives higher scores than the similarity of other relation pairs; and the distributional hierarchy of hyponyms and hypernyms indicates the directionality of hypernymy.

Our model is inspired by the *distributional inclusion hypothesis*, which expects prominent context words of hyponyms to appear in a subset of the hypernym contexts. We assume that each context word that appears with both a hyponym and its hypernym can be used as an indicator to determine which of the two words is semantically more general. Common context word vectors that represent distinctive characteristics of a hyponym are expected to be closer to the hyponym vector than to its hypernym vector. For example, the context word *flap* is more characteristic of a *bird* than of its hypernym *animal*; hence, the vector of *flap* should be closer to the vector of *bird* than to the vector of *animal*.

We evaluate our *HyperVec* model on both unsupervised and supervised hypernymy detection and directionality tasks. In addition, we apply the model to the task of graded lexical entailment (Vulić et al. (2017)), and we assess the capability of *HyperVec* on generalizing hypernymy by mapping to German and Italian. Results on benchmark datasets of hypernymy show that the hierarchical embeddings outperform state-of-the-art measures and previous embedding models. Furthermore, the implementation of our

models is publicly available.[1] The approaches and experiments described in this chapter are published in Nguyen et al. (2017a).

5.2 Related Work

Unsupervised hypernymy measures: A variety of directional measures for unsupervised hypernymy detection (Weeds and Weir, 2003; Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012) all rely on some variation of the *distributional inclusion hypothesis*: If u is a semantically narrower term than v, then a significant number of salient distributional features of u is expected to be included in the feature vector of v as well. In addition, Santus et al. (2014a) proposed the *distributional informativeness hypothesis*, that hypernyms tend to be less informative than hyponyms, and that they occur in more general contexts than their hyponyms. Rimell (2014) introduced a measure which is the ratio of change in topic coherence, for hypernymy detection. The measure detects hypernyms with reasonable accuracy, and a family of topic coherence measures is used to perform a multi-way classification of tuples by relation class. All of these approaches represent words as vectors in distributional semantic models (Turney and Pantel, 2010), relying on the *distributional hypothesis* (Harris, 1954; Firth, 1957). For evaluation, these directional models use the AP measure to assess the proportion of hypernyms at the top of a score-sorted list. In a different vein, Kiela et al. (2015) introduced three unsupervised methods drawn from visual properties of images to determine a concept's generality in hypernymy tasks.

Supervised hypernymy methods: The studies in this area are based on word embeddings which represent words as low-dimensional and real-valued vectors (Mikolov et al., 2013a; Pennington et al., 2014). Each hypernymy pair is encoded by some combination of the two word vectors, such as concatenation (Baroni et al., 2012) or difference (Roller et al., 2014; Weeds et al., 2014). Hypernymy is distinguished from other relations by using a classification approach, such as SVM or LR. Because word embeddings are trained for similar and symmetric vectors, it is however unclear whether the supervised methods do actually learn the asymmetry in hypernymy (Levy et al., 2015).

Hypernymy-specific embeddings: These approaches are closest to our work. Yu et al. (2015) proposed a dynamic distance-margin model to learn term embeddings that capture properties of hypernymy. The neural model is trained on the taxonomic relation data which is pre-extracted. The resulting term embeddings are fed to an SVM classifier to predict hypernymy. However, this model only learns term pairs without considering their contexts, leading to a lack of generalization for term embeddings. Tuan et al. (2016)

[1] `www.ims.uni-stuttgart.de/data/hypervec`

introduced a dynamic weighting neural network to learn term embeddings that encode information about hypernymy and also about their contexts, considering all words between a hypernym and its hyponym in a sentence. The proposed model is trained on a set of hypernym relations extracted from WordNet (Miller, 1995). The embeddings are applied as features to detect hypernymy, using an SVM classifier. Tuan et al. (2016) handles the drawback of the approach by Yu et al. (2015), considering the contextual information between two terms; however the method still is not able to determine the directionality of a hypernym pair. Vendrov et al. (2016) proposed a method to encode order into learned distributed representations, to explicitly model partial order structure of the visual-semantic hierarchy or the hierarchy of hypernymy in WordNet. The resulting vectors are used to predict the transitive hypernym relations in WordNet.

5.3 Approach

In this section, we present our model of hierarchical embeddings *HyperVec*. We first describe the extraction of hypernymy (Section 5.3.1). Section 5.3.2 describes how we learn the embeddings for hypernymy, and Section 5.3.3 introduces the unsupervised measure *HyperScore* that is applied to the hypernymy tasks.

5.3.1 Extracting Hypernymy

The proposed approach makes use of the set of hypernymy to learn the hierarchical embeddings. For this purpose, we rely on WordNet which is a large lexical database of English for collecting the set of hypernymy. We extract all of hypernym relations for noun terms and verb terms, including both direct and indirect hypernym relations. Before training our model, we exclude all hypernym pairs which appear in any datasets used for evaluation.

Figure 5.1 shows the direct and indirect hypernymy in WordNet. For example, the direct hypernym of "frog" is "amphibian", while the indirect hypernyms of "frog" are "verterbrate", "chordate", and "animal". For training our model, we exclude all hypernym relations (the true label) which appear in all datasets used for the evaluation, guaranteeing the hypernym relations in the datasets are unseen during training the model. For example, if the hyponym–hypernym pair of "bird–animal" appears in any dataset, it will be excluded from the training set.

5.3.2 Learning Hierarchical Embeddings

In the following, Section 5.3.2 first describes the Skip-gram model which is integrated into our model for optimization. Section 5.3.2 then describes the objective functions to train the hierarchical embeddings for hypernymy.

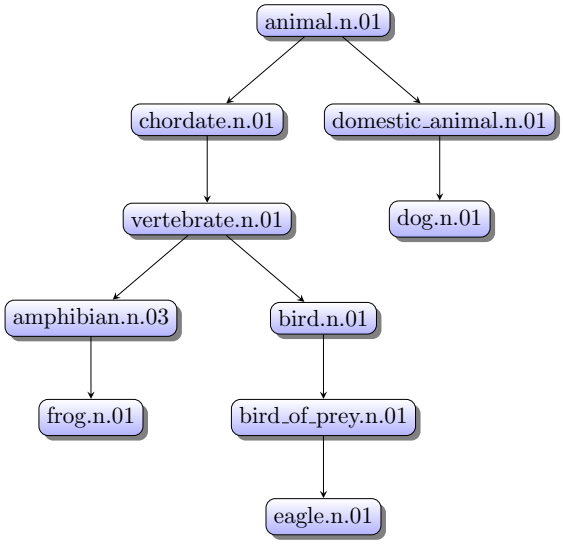


Figure 5.1: Illustration of the direct and indirect hypernymy in WordNet.

Skip-gram Model

The Skip-gram model is a word embeddings method suggested by Mikolov et al. (2013a). Levy and Goldberg (2014) introduced a variant of the Skip-gram model with negative sampling, in which the objective function is defined as follows:

$$\mathrm{J}_{SGNS} = \sum_{w \in V_W} \sum_{c \in V_C} \mathrm{J}_{(w,c)} \tag{5.1}$$

$$\mathrm{J}_{(w,c)} = \#(w,c) \log \sigma(\vec{w}, \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-\vec{w}, \vec{c}_N)] \tag{5.2}$$

where the skip-gram with negative sampling is trained on a corpus of words $w \in V_W$ and their contexts $c \in V_C$, with V_W and V_C the word and context vocabularies, respectively. The collection of observed words and context pairs is denoted as D; the term $\#(w,c)$ refers to the number of times the pair (w,c) appeared in D; the term $\sigma(x)$ is the sigmoid function; the term k is the number of negative samples and the term c_N is the sampled context, drawn according to the empirical unigram distribution P.

Hierarchical Hypernymy Model

Vector representations for detecting hypernymy are usually encoded by standard first-order distributional co-occurrences. In this way, they are insufficient to differentiate hypernymy from other paradigmatic relations such as synonymy, meronymy, antonymy, etc. Incorporating directional measures of hypernymy to detect hypernymy by exploiting the common contexts of hypernym and hyponym improves this relation distinction, but still suffers from distinguishing between hypernymy and meronymy.

Our novel approach presents two solutions to deal with these challenges. First of all, the embeddings are learned in a specific order, such that the similarity score for hyper-

nymy is higher than the similarity score for other relations. For example, the hypernym pair *animal–frog* will be assigned a higher cosine score than the co-hyponymy pair *eagle–frog*. Secondly, the embeddings are learned to capture the distributional hierarchy between hyponym and hypernym, as an indicator to differentiate between hypernym and hyponym. For example, given a hyponym–hypernym pair (p, q), we can exploit the Euclidean norms of $\vec{q}$ and $\vec{p}$ to differentiate between the two words, such that the Euclidean norm of the hypernym $\vec{q}$ is larger than the Euclidean norm of the hyponym $\vec{p}$.

Inspired by the distributional lexical contrast model in Nguyen et al. (2016a) for distinguishing antonymy from synonymy, this chapter proposes two objective functions to learn hierarchical embeddings for hypernymy. Before moving to the details of the two objective functions, we first define the terms as follows: $\mathbb{W}(c)$ refers to the set of words co-occurring with the context c in a certain window-size; $\mathbb{H}(w)$ denotes the set of hypernyms for the word w; the two terms $\mathbb{H}^+(w, c)$ and $\mathbb{H}^-(w, c)$ are drawn from $\mathbb{H}(w)$, and are defined as follows:

$$\mathbb{H}^+(w, c) = \{u \in \mathbb{W}(c) \cap \mathbb{H}(w) : cos(\vec{w}, \vec{c}) - cos(\vec{u}, \vec{c}) \geq \theta\}$$
$$\mathbb{H}^-(w, c) = \{v \in \mathbb{W}(c) \cap \mathbb{H}(w) : cos(\vec{w}, \vec{c}) - cos(\vec{v}, \vec{c}) < \theta\}$$

where $cos(\vec{x}, \vec{y})$ stands for the cosine similarity of the two vectors $\vec{x}$ and $\vec{y}$; θ is the margin. The set $\mathbb{H}^+(w, c)$ contains all hypernyms of the word w that share the context c and satisfy the constraint that the cosine similarity of pair (w, c) is higher than the cosine similarity of pair (u, c) within a max-margin framework θ. Similarly, the set $\mathbb{H}^-(w, c)$ represents all hypernyms of the word w with respect to the common context c in which the cosine similarity difference between the pair (w, c) and the pair (v, c) is within a min-margin framework θ. The two objective functions are defined as follows:

$$\mathrm{L}_{(w,c)} = \frac{1}{\#(w, u)} \sum\nolimits_{u \in \mathbb{H}^+(w,c)} \partial(\vec{w}, \vec{u}) \tag{5.3}$$
$$\mathrm{L}_{(v,w,c)} = \sum\nolimits_{v \in \mathbb{H}^-(w,c)} \partial(\vec{v}, \vec{w}) \tag{5.4}$$

where the term $\partial(\vec{x}, \vec{y})$ stands for the cosine derivative of $(\vec{x}, \vec{y})$; and ∂ then is optimized by the negative sampling procedure.

The objective function in Equation 5.3 minimizes the distributional difference between the hyponym w and the hypernym u by exploiting the common context c. More specifically, if the common context c is the distinctive characteristic of the hyponym w (i.e. the common context c is closer to the hyponym w than to the hypernym u), the objective function $\mathrm{L}_{(w,c)}$ tries to decrease the distributional generality of hypernym u by moving w closer to u. For example, given a hypernym-hyponym pair *animal–bird*, the context *flap* is a distinctive characteristic of *bird*, because almost every *bird* can flap, but not every *animal* can flap. Therefore, the context *flap* is closer to the hyponym *bird* than

to the hypernym *animal*. The model then tries to move *bird* closer to *animal* in order to enforce the similarity between *bird* and *animal*, and to decrease the distributional generality of *animal*.

In contrast to Equation 5.3, the objective function in Equation 5.4 minimizes the distributional difference between the hyponym w and the hypernym v by exploiting the common context c, which is a distinctive characteristic of the hypernym v. In this case, the objective function $\mathrm{L}_{(v,w,c)}$ tries to reduce the distributional generality of hyponym w by moving v closer to w. For example, the context word *rights*, a distinctive characteristic of the hypernym *animal*, should be closer to *animal* than to *bird*. Hence, the model tries to move the hypernym *animal* closer to the hyponym *bird*. Given that hypernymy is an asymmetric and also a hierarchical relation, where each hypernym may contain several hyponyms, our objective functions updates simultaneously both the hypernym and all of its hyponyms; therefore, our objective functions are able to capture the hierarchical relations between the hypernym and its hyponyms. Moreover, in our model, the margin framework θ plays a role in learning the hierarchy of hypernymy, and in preventing the model from minimizing the distance of synonymy or antonymy, because synonymy and antonymy share many contexts.

In the final step, the objective function which is used to learn the hierarchical embeddings for hypernymy combines Equations 5.1, 5.2, 5.3, and 5.4 by the objective function in Equations 5.5 and 5.6:

$$\mathrm{J}_{(w,v,c)} = \mathrm{J}_{(w,c)} + \mathrm{L}_{(w,c)} + \mathrm{L}_{(v,w,c)} \tag{5.5}$$

$$\mathrm{J} = \sum_{w \in V_W} \sum_{c \in V_C} \mathrm{J}_{(w,v,c)} \tag{5.6}$$

5.3.3 Unsupervised Hypernymy Measure

HyperVec is expected to show the two following properties: (i) the hyponym and the hypernym are close to each other, and (ii) there exists a distributional hierarchy between hypernyms and their hyponyms. Given a hypernymy pair (u, v) in which u is the hyponym and v is the hypernym, we propose a measure to detect hypernymy and to determine the directionality of hypernymy by using the hierarchical embeddings as follows:

$$\mathit{HyperScore}(u, v) = cos(\vec{u}, \vec{v}) * \frac{\|\vec{v}\|}{\|\vec{u}\|} \tag{5.7}$$

where $cos(\vec{u}, \vec{v})$ is the cosine similarity between $\vec{u}$ and $\vec{v}$, and $\|\cdot\|$ is the magnitude of the vector (or the Euclidean norm). The cosine similarity is applied to distinguish hypernymy from other relations, due to the first property of the hierarchical embeddings, while the second property is used to decide about the directionality of hypernymy, assuming that the magnitude of the hypernym is larger than the magnitude of the hyponym. Note

that the proposed hypernymy measure is unsupervised when the resource is only used to learn hierarchical embeddings.

5.4 Experiment

In this section, we first describe the experimental settings in our experiments (Section 5.4.1). We then evaluate the performance of *HyperVec* on three different tasks: i) unsupervised hypernymy detection and directionality (Section 5.4.2), where we assess *HyperVec* on ranking and classifying hypernymy; ii) supervised hypernymy detection (Section 5.4.3), where we apply supervised classification to detect hypernymy; iii) graded lexical entailment (Section 5.4.4), where we predict the strength of hypernymy pairs.

5.4.1 Experimental Settings

We use the ENCOW14A corpus (Schäfer and Bildhauer, 2012; Schäfer, 2015) with approx. 14.5 billion tokens for training the hierarchical embeddings and the default SGNS model. We train our model with 100 dimensions, a window size of 5, 15 negative samples, and 0.025 as the learning rate. The threshold θ is set to 0.05. The hypernymy resource for nouns comprises $105,020$ hyponyms, $24,925$ hypernyms, and $1,878,484$ hyponym–hypernym pairs. The hypernymy resource for verbs consists of $11,328$ hyponyms, $4,848$ hypernyms, and $130,350$ hyponym–hypernym pairs.

5.4.2 Unsupervised Hypernymy Detection and Directionality

In this section, we assess our model on two experimental setups: i) a ranking retrieval setup that expects hypernymy pairs to have a higher similarity score than instances from other semantic relations; ii) a classification setup that requires both hypernymy detection and directionality.

Ranking Retrieval

Shwartz et al. (2017) conducted an extensive evaluation of a large number of unsupervised distributional measures for hypernymy ranking retrieval proposed in previous work (Weeds and Weir, 2003; Santus et al., 2014a; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012; Santus et al., 2016). The evaluation was performed on four semantic relation datasets: **BLESS** (Baroni and Lenci, 2011), **Weeds** (Weeds et al., 2004), **EVALution** (Santus et al., 2015), and **Lenci&Benotto** (Benotto, 2015).

Table 5.1 describes the detail of these datasets in terms of the semantic relations and the number of instances. The AP ranking measure is used to evaluate the performance of the measures.

Dataset	Relation	#Instance	Total
BLESS	hypernymy	1,337	26,554
	meronymy	2,943	
	coordination	3,565	
	event	3,824	
	attribute	2,731	
	random-n	6,702	
	random-j	2,187	
	random-v	3,265	
EVALution	hypernymy	3,637	13,465
	meronymy	1,819	
	attribute	2,965	
	synonymy	1,888	
	antonymy	3,156	
Lenci&Benotto	hypernymy	1,933	5,010
	synonymy	1,311	
	antonymy	1,766	
Weeds	hypernymy	1,469	2,928
	coordination	1,459	

Table 5.1: Details of the semantic relations and the number of instances in each dataset.

In comparison to the state-of-the-art unsupervised measures compared by Shwartz et al. (2017) (henceforth, baseline models), we apply our unsupervised measure *HyperScore* (Equation 5.7) to rank hypernymy against other relations. Table 5.2 presents the results of using *HyperScore* vs. the best baseline models, across datasets. When detecting hypernymy among all other relations (which is the most challenging task), *HyperScore* significantly outperforms all baseline variants on all datasets. The strongest difference is reached on the BLESS dataset, where *HyperScore* achieves an improvement of 40% AP score over the best baseline model. When ranking hypernymy in comparison to a single other relation, *HyperScore* also improves over the baseline models, except for the *event* relation in the BLESS dataset. We assume that this is due to the different parts-of-speech (adjective and noun) involved in the relation, where *HyperVec* fails to establish a hierarchy.

Classification

In this setup, we rely on three datasets of semantic relations, which were all used in various state-of-the-art approaches before, and brought together for hypernymy evaluation by Kiela et al. (2015). **(i)** A subset of **BLESS** contains 1,337 hyponym-hypernym pairs. The task is to predict the directionality of hypernymy within a binary classification. Given a word pair such as *freezer–device*, we aim to identify the hyponym (*freezer*) and the hypernym (*device*). Our approach requires no threshold; we only need to compare the magnitudes of the two words and to assign the hypernym label to the word with

Dataset	Hypernymy vs.	Baseline	HyperScore
EVALution	other relations	0.353	**0.538**
	meronymy	0.675	**0.811**
	attribute	0.651	**0.800**
	antonymy	0.55	**0.743**
	synonymy	0.657	**0.793**
BLESS	other relations	0.051	**0.454**
	meronymy	0.76	**0.913**
	coordination	0.537	**0.888**
	attribute	0.74	**0.918**
	event	**0.779**	0.620
Lenci&Benotto	other relations	0.382	**0.574**
	antonymy	0.624	**0.696**
	synonymy	0.725	**0.751**
Weeds	coordination	0.441	**0.850**

Table 5.2: AP results of *HyperScore* in comparison to state-of-the-art measures.

the larger magnitude.

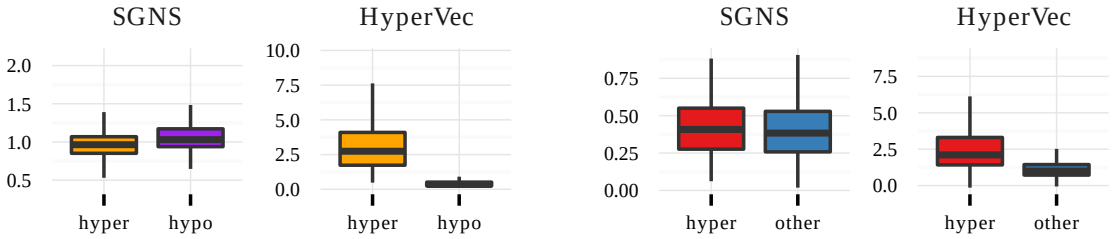
$$dir(w_1, w_2) = \begin{cases} w_1 = \text{hyper} & \text{if } \frac{\|\vec{w_1}\|}{\|\vec{w_2}\|} \geq 1 \\ w_1 = \text{hypo} & \text{otherwise} \end{cases} \tag{5.8}$$

Figure 5.2a indicates that the magnitude values of the *SGNS* model cannot distinguish between a hyponym and a hypernym, while the hierarchical embeddings provide a larger magnitude for the hypernym. **(ii)** Following Weeds et al. (2014), we conduct a binary classification with a subset of 1,168 BLESS word pairs. In this dataset (**WBLESS**), one class is represented by hyponym–hypernym pairs, and the other class is a combination of reversed hypernym–hyponym pairs, plus additional holonym-meronym pairs, co-hyponyms and randomly matched nouns. For this classification we make use of our *HyperScore* measure that ranks hypernymy pairs higher than other relation pairs. A threshold decides about the splitting point between the two classes: *hyper* vs. *other*. Instead of using a manually defined threshold as done by Kiela et al. (2015), we decided to run 1 000 iterations which randomly sampled only 2% of the available pairs for learning a threshold, using the remaining 98% for test purposes. We present average accuracy results across all iterations. Figure 5.2b compares the default cosine similarities between the relation pairs (as applied by *SGNS*) and *HyperScore* (as applied by *HyperVec*) on this task. Using *HyperScore*, the class "hyper" can clearly be distinguished from the class "other". **(iii) BIBLESS** represents the most challenging dataset; the relation pairs from WBLESS are split into three classes instead of two: hypernymy pairs, reversed hypernymy pairs, and other relation pairs. In this case, we perform a three-way classification. We apply the same technique as used for the WBLESS classification, but in cases where we classify *hyper* we additionally classify the hypernymy direction, to decide between

	BLESS	WBLESS	BIBLESS
Kiela et al. (2015)	0.88	0.75	0.57
Santus et al. (2014a)	0.87	—	—
Weeds et al. (2014)	—	0.75	—
SGNS	0.44	0.48	0.34
HyperVec	**0.92**	**0.87**	**0.81**

Table 5.3: Accuracy for hypernymy directionality.

hyponym–hypernym pairs and reversed hypernym–hyponym pairs.



(a) Directionality task

(b) Hypernymy detection

Figure 5.2: Comparing *SGNS* and *HyperVec* on binary classification tasks. The y-axis shows the magnitude values of the vectors.

Table 5.3 compares our results against related work. *HyperVec* outperforms all other methods on all three tasks. In addition we see again that an unmodified *SGNS* model cannot solve any of the three tasks.

5.4.3 Supervised Hypernymy Detection

For supervised hypernymy detection, we make use of the two datasets: the full **BLESS** dataset, and **ENTAILMENT** (Baroni et al., 2012), containing 2,770 relation pairs in total, including 1,385 hypernym pairs and 1,385 other relations pairs. We follow the same procedure as Yu et al. (2015) and Tuan et al. (2016) to assess *HyperVec* on the two datasets. Regarding BLESS, we extract pairs for four types of relations: hypernymy, meronymy, co-hyponymy (or *coordination*), and add the random relation for nouns. For the evaluation, we randomly select one concept and its relatum for testing, and train the supervised model on the 199 remaining concepts and its relatum. We then report the average accuracy across all concepts. For the ENTAILMENT dataset, we randomly select one hypernym pair for testing and train on all remaining hypernym pairs. Again, we report the average accuracy across all hypernyms.

We apply an SVM classifier to detect hypernymy based on *HyperVec*. Given a hyponym–hypernym pair (u, v), we concatenate four components to construct the vector for a pair (u, v) as follows: the vector difference between hypernym and hyponym $(\vec{v} - \vec{u})$; the

Models	BLESS	ENTAILMENT
Yu et al. (2015)	0.90	0.87
Tuan et al. (2016)	0.93	0.91
HyperVec	**0.94**	0.91

Table 5.4: Classification results for BLESS and ENTAILMENT in terms of accuracy.

cosine similarity between the hypernym and hyponym vectors ($cos(\vec{u}, \vec{v})$); the magnitude of the hyponym ($\|\vec{u}\|$); and the magnitude of the hypernym ($\|\vec{v}\|$). The resulting vector is fed into the SVM classifier to detect hypernymy. Similar to the two previous works, we train the SVM classifier with the RBF kernel, $\lambda = 0.03125$, and the penalty $C = 8.0$.

Table 5.4 shows the performance of *HyperVec* and the two baseline models reported by Tuan et al. (2016). *HyperVec* slightly outperforms the method of Tuan et al. (2016) on the BLESS dataset, and is equivalent to the performance of their method on the ENTAILMENT dataset. In comparison to the method of Yu et al. (2015), *HyperVec* achieves significant improvements.

5.4.4 Graded Lexical Entailment

In this experiment, we apply *HyperVec* to the dataset of graded lexical entailment, *HyperLex*, as introduced by Vulić et al. (2017). The *HyperLex* dataset provides soft lexical entailment on a continuous scale, rather than simplifying into a binary decision. For instance, when humans are asked to give an example instance of the concept *sport*, it turns out that *football* and *basketball* are more frequently picked then *chess, softball* or *racquetball*. *HyperLex* contains 2,616 word pairs across seven semantic relations and two word classes (nouns and verbs). Each word pair is rated by a score that indicates the strength of the semantic relation between the two words. For example, the score of the hypernym pair *duck–animal* is 5.9 out of 6.0, while the score of the reversed pair *animal–duck* is only 1.0.

We compared *HyperScore* against the most prominent state-of-the-art hypernymy and lexical entailment models from previous work:

- Directional entailment measures (DEM) (Weeds and Weir, 2003; Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012)
- Generality measures (SQLS) (Santus et al., 2014a)
- Visual generality measures (VIS) (Kiela et al., 2015)
- Consideration of concept frequency ratio (FR) (Vulić et al., 2017)
- WordNet-based similarity measures (WN) (Wu and Palmer, 1994; Pedersen et al., 2004)

- Order embeddings (OrderEmb) (Vendrov et al., 2016)
- Skip-gram embeddings (SGNS) (Mikolov et al., 2013a; Levy and Goldberg, 2014)
- Embeddings fine-tuned to a paraphrase database with linguistic constraints (PARAGRAM) (Mrkšić et al., 2016)
- Gaussian embeddings (Word2Gauss) (Vilnis and McCallum, 2015)

The performance of the models is assessed through Spearman's rank-order correlation coefficient ρ (Siegel and Castellan, 1988), comparing the ranks of the models' scores and the human judgments for the given word pairs.

Measures		**Embeddings**	
Model	ρ	**Model**	ρ
FR	0.279	SGNS	0.205
DEM	0.180	PARAGRAM	0.320
SLQS	0.228	OrderEmb	0.191
WN	0.234	Word2Gauss	0.206
VIS	0.209	*HyperScore*	**0.540**

Table 5.5: Results (ρ) of *HyperScore* and state-of-the-art measures and word embedding models on graded lexical entailment.

Table 5.5 shows that *HyperScore* significantly outperforms both state-of-the-art measures and word embedding models. *HyperScore* outperforms even the previously best word embedding model PARAGRAM by .22, and the previously best measures FR by .27. The reason that *HyperVec* outperforms all other models is that the hierarchy between hypernym and hypornym within *HyperVec* differentiates hyponym–hypernym pairs from hypernym–hyponym pairs. For example, the *HyperScore* for the pairs *duck–animal* and *animal–duck* are 3.02 and 0.30, respectively. Thus, the magnitude proportion of the hypernym–hyponym pair *duck–animal* is larger than that for the pair *animal–duck*.

5.5 Generalizing Hypernymy

Having demonstrated the general abilities of *HyperVec*, this final section explores its potential for generalization in two different ways, (i) by relying on a small seed set only, rather than using a large set of training data; and (ii) by projecting *HyperVec* to other languages.

Hypernymy Seed Generalization: We utilize only a small hypernym set from the hypernymy resource to train *HyperVec*, relying on 200 concepts from the BLESS dataset. The motivation behind using these concepts is threefold: i) these concepts are distinct

and unambiguous noun concepts; ii) the concepts were equally divided between living and non-living entities; iii) concepts have been grouped into 17 broader classes. Based on the seed set, we collected the hyponyms of each concept from WordNet, and then re-trained *HyperVec*.

On the hypernymy ranking retrieval task (Section 5.4.2), *HyperScore* outperforms the baselines across all datasets (cf. Table 1) with AP values of 0.39, 0.448, and 0.585 for EVALution, LenciBenotto, and Weeds, respectively. For the graded lexical entailment task (Section 5.4.4), *HyperScore* obtains a correlation of $\rho = 0.30$, outperforming all models except for PARAGRAM with $\rho = 0.32$. Overall, the results show that *HyperVec* is indeed able to generalize hypernymy from small seeds of training data.

Generalizing Hypernymy across Languages: We assume that hypernymy detection can be improved across languages by projecting representations from any arbitrary language into our modified English *HyperVec* space. We conduct experiments for German and Italian, where the language-specific representations are obtained using the same hyper-parameter settings as for our English *SGNS* model (cf. Section 5.4.1). As corpus resource we relied on Wikipedia dumps[2]. Note that we do not use any additional resource, such as the German or Italian WordNet, to tune the embeddings for hypernymy detection. Based on the representations, a mapping function between a source language (German, Italian) and our English *HyperVec* space is learned, by relying on the least-squares error method from previous work using cross-lingual data (Mikolov et al., 2013b) and different modalities (Lazaridou et al., 2015).

To learn a mapping function between two languages, a one-to-one correspondence (word translations) between two sets of vectors is required. We obtained these translations by using the parallel Europarl[3] V7 corpus for German–English and Italian–English. Word alignment counts were extracted using *fast_align* (Dyer et al., 2013). We then assigned each source word to the English word with the maximum number of alignments in the parallel corpus. We could match 25,547 pairs for DE→EN and 47,475 pairs for IT→EN.

Taking the aligned subset of both spaces, we assume that X is the matrix obtained by concatenating all source vectors, and likewise Y is the matrix obtained by concatenating all corresponding English elements. Applying the $\ell 2$-regularized least-squares error objective can be described using the following equation:

$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d1 \times d2}}{\operatorname{argmin}} \|\mathbf{XW} - \mathbf{Y}\| + \lambda \|\mathbf{W}\| \tag{5.9}$$

Although we learn the mapping only on a subset of aligned words, it allows us to project

[2]The Wikipedia dump for German and Italian were both downloaded in January 2017.
[3]http://www.statmt.org/europarl/

every word in a source vocabulary to its English *HyperVec* position by using **W**.

Finally we compare the original representations and the mapped representation on the hypernymy ranking retrieval task (similar to Section 5.4.2). As gold resources we relied on German and Italian nouns pairs. For German we used the 282 German pairs collected via Amazon Mechanical Turk by Scheible and Schulte im Walde (2014). The 1,350 Italian pairs were collected via Crowdflower by Sucameli (2015) in the same way. Both collections contain hypernymy, antonymy and synonymy pairs. As before, we evaluate the ranking by AP, and we compare the cosine of the unmodified default representations against the *HyperScore* of the projected representations.

German	Hyp/All	Hyp/Syn	Hyp/Ant
DE-*SGNS*	0.28	0.48	0.40
DE→EN*HyperVec*	**0.37**	**0.65**	**0.47**
Italian			
IT-*SGNS*	0.38	0.50	0.60
IT→EN*HyperVec*	**0.44**	**0.57**	**0.65**

Table 5.6: AP results across languages, comparing *SGNS* and the projected representations.

The results are shown in Table 5.6. We clearly see that for both languages the default *SGNS* embeddings do not provide higher similarity scores for hypernymy pairs (except for Italian Hyp/Ant), but both languages provide higher scores when we map the embeddings into the English *HyperVec* space.

5.6 Summary

In addition to the success of approaches described in chapters 3 and 4, which aim to solve the tasks of distinguishing antonymy and synonymy by improving word vector representations and constructing pattern-based neural models, this chapter presented a novel neural model *HyperVec* to learn hierarchical embeddings for hypernymy.

Specifically, the proposed model is learned to solve the tasks of hypernymy detection and directionality. For the hypernymy detection task, *HyperVec* has been shown to strengthen hypernymy similarity, which is higher than similarity of other relations, helping to distinguish hypernymy from other relations. Regarding the hypernymy directionality task, *HyperVec* is able to capture the distributional hierarchy of hypernymy, providing an indicator to determine which of the two words is the hypernym and which is the hyponym. Moreover, an unsupervised measure *HyperScore*, used to detect hypernymy and to determine the directionality of hypernymy, is also presented. Firstly, *HyperScore* makes use of cosine similarity of hypernymy and compares it to that of

other relations. Secondly, *HyperScore* considers the comparison between embeddings of a hypernym and a hyponym in terms of the Euclidean norm.

To verify the *HyperVec*, we conducted a series of experiments in which *HyperVec* was evaluated on unsupervised hypernymy detection and directionality tasks, supervised hypernymy detection task, and graded lexical entailment task. Moreover, we also evaluated *HyperVec* on generalizing hypernymy. Together with the newly proposed unsupervised measure *HyperScore*, the experiments demonstrated (i) significant improvements over state-of-the-art measures, and (ii) the capability to generalize hypernymy and learn the relation instead of memorizing *prototypical hypernyms*.

6 Neural-based Noise Filtering from Word Embeddings

6.1 Introduction

Word embeddings aim to represent words as low-dimensional dense vectors. In comparison to distributional count vectors, word embeddings address the problematic sparsity of word vectors and have achieved impressive results in many NLP tasks such as sentiment analysis (e.g., Kim (2014)), word similarity (e.g., Pennington et al. (2014)), and parsing (e.g., Lazaridou et al. (2013)). Moreover, word embeddings are attractive because they can be learned in an unsupervised fashion from unlabeled raw corpora. There are two main approaches to create word embeddings. The first approach makes use of neural-based techniques to learn word embeddings, such as the Skip-gram model (Mikolov et al., 2013a). The second approach is based on matrix factorization (Pennington et al., 2014), building word embeddings by factorizing word-context co-occurrence matrices.

In recent years, a number of approaches has focused on improving word embeddings, often by integrating lexical resources. For example, Adel and Schütze (2014) applied coreference chains to Skip-gram models in order to create word embeddings for antonym identification. Pham et al. (2015) proposed an extension of the Skip-gram model by integrating synonyms and antonyms from WordNet. Their extended Skip-gram model outperformed the standard Skip-gram model on both general semantic tasks and tasks of distinguishing antonyms from synonyms.

In chapter 3, we presented an approach where distributional lexical contrast was integrated into every single context of a target word in a Skip-gram model for training word embeddings. The resulting word embeddings were used in similarity tasks, and to distinguish between antonyms and synonyms. In a similar spirit, chapter 5 introduced the hierarchical embeddings of hypernymy which are learned to distinguish hypernymy from other relations, and to discriminate between a hypernym and a hyponym in hypernym pairs. In contrast, Faruqui et al. (2015) improved word embeddings without relying on lexical resources by applying ideas from sparse coding to transform dense word embeddings into sparse word embeddings. The dense vectors in their models can be transformed into sparse overcomplete vectors or sparse binary overcomplete vectors. They showed that the resulting vector representations were more similar to interpretable features in

NLP and outperformed the original vector representations on several benchmark tasks. Yet, the disadvantage of these kinds of approaches is that they are supervised learning. In other words, these approaches make use of external resources beside unlabeled raw corpora to improve word embeddings. Moreover, word embeddings play an important role in NLP applications, because they are often utilized as inputs and trained to capture crucial information such as syntax and semantica. Therefore, improving word embeddings also helps NLP applications improve their performance.

In this chapter, we present two neural models that aim to improve word embeddings without using any external resources. The hypothesis behind our approaches is that word embeddings contain unnecessary information, i.e. *noise*. We start out with the idea of learning word embeddings as suggested by Mikolov et al. (2013a), relying on the distributional hypothesis (Harris, 1954) that words with similar distributions have related meanings. We address those distributions in embedded vectors of words that decrease the value of such vector representations. For instance, consider the sentence *"the quick brown fox gazing at the cloud jumped over the lazy dog."* The context *"jumped"* can be used to predict the words *"fox"*, *"cloud"* and *"dog"* in a window size of 5 words; however, a *"cloud"* cannot *"jump"*. The context `jumped` is therefore considered as noise in the embedded vector of `cloud`. We propose two novel models to smooth word embeddings by filtering out noise. In other words, we strengthen salient contexts and weaken unnecessary contexts.

The first proposed model is referred to as the *complete word denoising embeddings model (CompEmb)*. Given a set of original word embeddings, we use a filter to learn a denoising matrix, and then project the set of original word embeddings into this denoising matrix to produce a set of complete word denoising embeddings. The second proposed model is referred to as the *overcomplete word denoising embeddings model (OverCompEmb)*. We make use of a sparse coding method to transform an input set of original word embeddings into a set of overcomplete word embeddings, which is considered the "overcomplete process." We then apply a filter to train a denoising matrix, and thereafter project the set of original word embeddings into the denoising matrix to generate a set of overcomplete word denoising embeddings. The key idea in our models is to use a filter for learning the denoising matrix. The architecture of the filter is a feed-forward, nonlinear and parameterized neural network with a fixed depth that can be used to learn the denoising matrices and reduce noise in word embeddings. Using state-of-the-art word embeddings as input vectors, we show that the resulting word denoising embeddings outperform the original word embeddings on several benchmark tasks, such as word similarity and word relatedness tasks, synonymy detection and noun phrase classification. Furthermore, the implementation of our models is publicly available[1]. The approaches and experiments described in this chapter are published in Nguyen et al. (2016b).

[1] `https://github.com/nguyenkh/NeuralDenoising`

The remainder of this chapter is organized as follows: Section 6.2 presents the two proposed models, the sparse coding technique for overcomplete vectors, and the loss function. In Section 6.3, we demonstrate the experiments on evaluating the effects of our word denoising embeddings, tuning hyperparameters, and we analyze the effects of filter depth. Finally, Section 6.4 concludes the chapter.

6.2 Approach

In this section, we present the two contributions. Figure 6.1 illustrates our two models to learn denoising for word embeddings. The first model on the top, the complete word denoising embeddings model "CompEmb" (Section 6.2.1), filters noise from word embeddings $\mathbf{X}$ to produce complete word denoising embeddings $\mathbf{X}^*$, in which the vector length of $\mathbf{X}^*$ in comparison to $\mathbf{X}$ is unchanged after denoising (called *complete*). The second model at the bottom of the figure, the overcomplete word denoising embeddings model "OverCompEmb" (Section 6.2.2), filters noise from word embeddings $\mathbf{X}$ to yield overcomplete word denoising embeddings $\mathbf{Z}^*$, in which the vector length of $\mathbf{Z}^*$ tends to be greater than the vector length of $\mathbf{X}$ (called *overcomplete*).

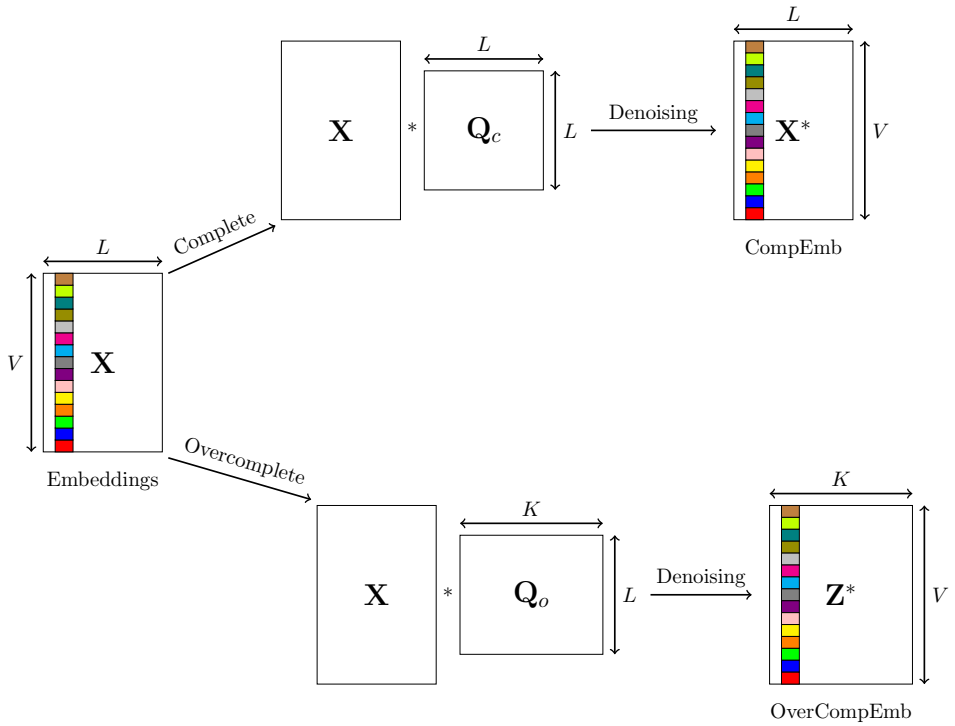


Figure 6.1: Illustration of word denoising embeddings methods, with complete word denoising embeddings at the top, and overcomplete word denoising embeddings at the bottom.

For the notations, let $\mathbf{X} \in \mathbb{R}^{V \times L}$ is an input set of word embeddings in which V is the vocabulary size, and L is the vector length of $\mathbf{X}$. Furthermore, $\mathbf{Z} \in \mathbb{R}^{V \times K}$ is the

overcomplete word embeddings in which K is the vector length of $\mathbf{Z}$ ($K > L$); finally, $\mathbf{D} \in \mathbb{R}^{L \times L}$ is the pre-trained dictionary (Section 6.2.3).

6.2.1 Complete Word Denoising Embeddings

In this subsection, we aim to reduce noise in the given input word embeddings $\mathbf{X}$ by learning a denoising matrix $\mathbf{Q_c}$. The complete word denoising embeddings $\mathbf{X}^*$ are then generated by projecting $\mathbf{X}$ into $\mathbf{Q_c}$. More specifically, given an input $\mathbf{X} \in \mathbb{R}^{V \times L}$, we seek to optimize the following objective function:

$$\underset{\mathbf{X},\mathbf{Q_c},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{x_i} - f(\mathbf{x_i}, \mathbf{Q_c}, \mathbf{S})\| + \alpha \|\mathbf{S}\|_1 \tag{6.1}$$

where f is a filter; $\mathbf{S}$ is a lateral inhibition matrix; and α is a regularization hyperparameter. Inspired by studies on sparse modeling, the matrix $\mathbf{S}$ is chosen to be symmetric and has zero on the diagonal.

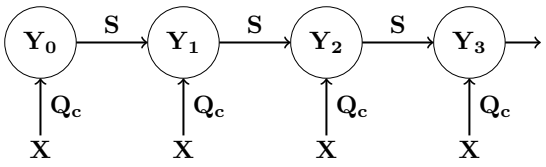
The goal of this matrix is to implement excitatory interaction between neurons, and to increase the convergence speed of the neural network (Szlam et al., 2011). More concretely, the matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are initialized with $\mathbf{I}$ and E, which are identity matrices, and the Lipschitz constant:

$$\begin{aligned}
&\mathbf{Q_c} = \tfrac{1}{E}\mathbf{D};\ \mathbf{S} = \mathbf{I} - \tfrac{1}{E}\mathbf{D^T D} \\
&E > \text{ the largest eigenvalue of } \mathbf{D^T D} \\
&\mathbf{D} \in \mathbb{R}^{L \times L} \text{ be pre-trained dictionary}
\end{aligned}$$

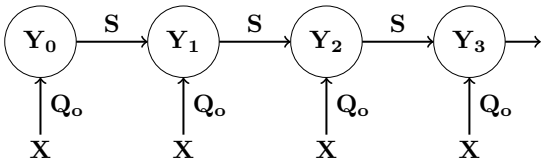
The underlying idea for reducing noise is to make use of a filter f to learn a denoising matrix $\mathbf{Q_c}$; hence, we design the filter f as a non-linear, parameterized, feed-forward architecture with a fixed depth that can be trained to approximate $f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$ to $\mathbf{X}$ as in Figure 6.2a. As a result, noise from word embeddings will be filtered by layers of the filter f. The filter f is encoded as a recursive function by iterating over the number of fixed depth T, as the following recursive Equation 6.2 shows:

$$\begin{aligned}
&\mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S}) \\
&\mathbf{Y}(0) = \mathcal{G}(\mathbf{X}\mathbf{Q_c}) \\
&\mathbf{Y}(k+1) = \mathcal{G}(\mathbf{X}\mathbf{Q_c} + \mathbf{Y}(k)\mathbf{S}) \\
&0 \leq k < T
\end{aligned} \tag{6.2}$$

$\mathcal{G}$ is a non-linear activation function. The matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are learned to produce the lowest possible error in a given number of iterations. Matrix $\mathbf{S}$, in the architecture of filter f, acts as a controllable matrix to filter unnecessary information on embedded vectors, and to impose restrictions on further reducing the computational burden (e.g.,



(a) CompEmb



(b) OverCompEmb

Figure 6.2: Architecture of the filters with the fixed depth $T = 3$.

solving low-rank approximation problem or keeping the number of terms at zero (Gregor and LeCun, 2010)). Moreover, the initialization of the matrices $\mathbf{Q_c}$, $\mathbf{S}$ and E enhances a highly efficient minimization of the objective function in Equation 6.1, due to the pre-trained dictionary $\mathbf{D}$ that carries the information of reconstructing $\mathbf{X}$.

The architecture of the filter f is a recursive feed-forward neural network with the fixed depth T, so the number of T plays a significant role in controlling the approximation of $\mathbf{X}^*$. The effects of T will be discussed later in Section 6.3.4. When $\mathbf{Q_c}$ is trained, the complete word denoising embeddings $\mathbf{X}^*$ are yielded by projecting $\mathbf{X}$ into $\mathbf{Q_c}$, as shown by the following Equation 6.3:

$$\mathbf{X}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_c}) \tag{6.3}$$

6.2.2 Overcomplete Word Denoising Embeddings

Now we introduce our method to reduce noise and overcomplete vectors in the given input word embeddings. To obtain overcomplete word embeddings, we first use a sparse coding method to transform the given input word embeddings $\mathbf{X}$ into overcomplete word embeddings $\mathbf{Z}$. Secondly, we use overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings to optimize the objective function: A set of input word embeddings $\mathbf{X} \in \mathbb{R}^{V \times L}$ is transformed to overcomplete word embeddings $\mathbf{Z} \in \mathbb{R}^{V \times K}$ by applying sparse coding method in Section 6.2.3. We then make use of the pre-trained dictionary $\mathbf{D} \in \mathbb{R}^{L \times K}$ and $\mathbf{Z} \in \mathbb{R}^{V \times K}$ to learn the denoising matrix $\mathbf{Q_o}$ by minimizing the following Equation 6.4:

$$\underset{\mathbf{X},\mathbf{Q_o},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{z_i} - f(\mathbf{x_i}, \mathbf{Q_o}, \mathbf{S})\| + \alpha \|\mathbf{S}\|_1 \tag{6.4}$$

The initialization of the parameters $\mathbf{Q_o}$, $\mathbf{S}$, E and α follows the same procedure as described in Section 6.2.1, and with the same interpretation of the filter architecture in Figure 6.2b. The overcomplete word denoising embeddings $\mathbf{Z}^*$ are then generated by

projecting $\mathbf{X}$ into the denoising matrix $\mathbf{Q_o}$ and using the non-linear activation function $\mathcal{G}$ in the following Equation 6.5:

$$\mathbf{Z}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_o}) \tag{6.5}$$

6.2.3 Sparse Coding

Sparse coding is a method to represent vector representations as a sparse linear combination of elementary atoms of a given dictionary. The underlying assumption of sparse coding is that the input vectors can be reconstructed accurately as a linear combination of some basis vectors and a few number of non-zero coefficients (Olshausen and Field, 1996).

The goal is to approximate a dense vector in $\mathbb{R}^L$ by a sparse linear combination of a few columns of a matrix $\mathbf{D} \in \mathbb{R}^{L \times K}$ in which K is a new vector length and the matrix $\mathbf{D}$ be called a *dictionary*. Concretely, given V input vectors of L dimensions $\mathbf{X} = [x_1, x_2, ..., x_V]$, the dictionary and sparse vectors can be formulated as the following minimization problem:

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{Z} \in \mathbb{R}^{K \times V}} \sum_{i=1}^{V} \|\mathbf{x_i} - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda \|\mathbf{z}_i\|_1 \tag{6.6}$$

$\mathbf{Z} = [z_1, ..., z_V]$ carries the decomposition coefficients of $\mathbf{X} = [x_1, x_2, ..., x_V]$; and λ represents a scalar to control the sparsity level of $\mathbf{Z}$. The dictionary $\mathbf{D}$ is typically learned by minimizing Equation 6.6 over input vectors $\mathbf{X}$. In the case of overcomplete representations $\mathbf{Z}$, the vector length K is typically implied as $K = \gamma L$ ($\gamma > 0$).

In the method of overcomplete word denoising embeddings (Section 6.2.2), our approach makes use of overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings reconstructed by applying a sparse coding method to word embeddings $\mathbf{X}$. The overcomplete word embeddings $\mathbf{Z}$ are then utilized to optimize Equation 6.4. To obtain overcomplete word embeddings $\mathbf{Z}$ and dictionaries, we use the `SPAMS` package[2] to implement sparse coding for word embeddings $\mathbf{X}$ and to train the dictionaries $\mathbf{D}$.

6.2.4 Loss Function

For each pair of term vectors $\mathbf{x_i} \in \mathbf{X}$ and $\mathbf{y_i} \in \mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$, we make use of the cosine similarity to measure the similarity between $\mathbf{x_i}$ and $\mathbf{y_i}$ as follows:

$$sim(\mathbf{x_i}, \mathbf{y_i}) = \frac{\mathbf{x_i} \cdot \mathbf{y_i}}{\|\mathbf{x_i}\| \|\mathbf{y_i}\|} \tag{6.7}$$

[2] http://spams-devel.gforge.inria.fr

Let Δ be the difference between $sim(\mathbf{x_i}, \mathbf{x_i})$ and $sim(\mathbf{x_i}, \mathbf{y_i})$, equivalently $\Delta = 1 - sim(\mathbf{x_i}, \mathbf{y_i})$. We then optimize the objective function in Equation 6.1 by minimizing Δ; and the same loss function is also applied to optimize the objective function in Equation 6.4. Training is done through Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012).

6.3 Experiment

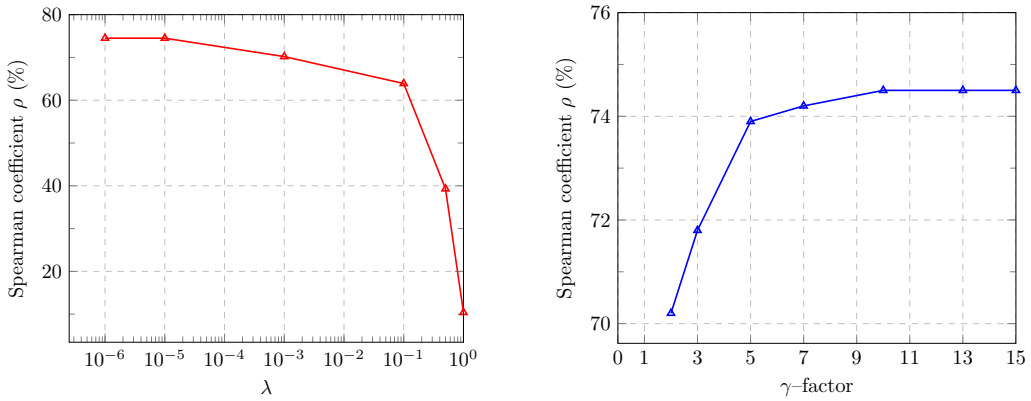
6.3.1 Experimental Settings

As input word embeddings, we rely on two state-of-the-art word embeddings methods: word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014). We use the `word2vec` tool[3] and the web corpus *ENCOW14A* (Schäfer and Bildhauer, 2012; Schäfer, 2015) which contains approximately 14.5 billion tokens, in order to train Skip-gram models with 100 and 300 dimensions. For the GloVe method, we use pre-trained vectors of 100 and 300 dimensions[4] that were trained on 6 billion words from Wikipedia and English Gigaword. The tanh function is used as the non-linear activation function in both approaches. The fixed depth of filter T is set to 3; further hyperparameters are chosen as discussed in Section 6.3.2. To train the networks, we use the `Theano` framework (Theano Development Team, 2016) to implement our models with a mini-batch size of 100. Regularization is applied by dropouts of 0.5 and 0.2 for input and output layers (without tuning), respectively.

6.3.2 Hyperparameter Tuning

In both methods of denoising word embeddings, the ℓ_1 regularization penalty α is set to 0.5 without tuning in Equation 6.1 and 6.4. The method of learning overcomplete word denoising embeddings relies on the mediate word embeddings $\mathbf{Z}$ to minimize the objective function in Equation 6.4. The sparsity of $\mathbf{Z}$ depends on the ℓ_1 regularization λ in Equation 6.6; and the length vector K of $\mathbf{Z}$ is implied as $K = \gamma L$. Therefore, we aim to tune λ and γ such that $\mathbf{Z}$ represents the nearest approximation of the original vector representation $\mathbf{X}$. We perform a grid search on $\lambda \in \{1.0, 0.5, 0.1, 10^{-3}, 10^{-6}\}$ and $\gamma \in \{2, 3, 5, 7, 10, 13, 15\}$, developing on the word similarity task WordSim353 (to be discussed on Section 6.3.3). The hyperparameter tunings are illustrated in Figures 6.3a and 6.3b for sparsity and overcomplete vector length tuning, respectively. In both approaches, we set λ to 10^{-6} and γ to 10 for the sparsity and length of overcomplete word embeddings.

[3]https://code.google.com/p/word2vec/
[4]http://www-nlp.stanford.edu/projects/glove/



(a) Sparsity of sparse coding.

(b) Length of overcomplete vectors.

Figure 6.3: Illustration of hyperparameter tuning.

6.3.3 Effects of Word Denoising Embeddings

In this section, we quantify the effects of word denoising embeddings on three kinds of tasks: similarity and relatedness tasks, detecting synonymy, and bracketed noun phrase classification task. In comparison to the performance of word denoising embeddings, we take into account state-of-the-art word embeddings (Skip-gram and GloVe word embeddings) as baselines. Besides, we also use the public source code[5] to re-implement the two methods suggested by Faruqui et al. (2015) which are vectors $\mathbf{A}$ (sparse overcomplete vectors) and $\mathbf{B}$ (sparse binary overcomplete vectors).

The effects of the word denoising embeddings on the tasks are shown in Table 6.1. The results show that the vectors $\mathbf{X}^*$ and $\mathbf{Z}^*$ outperform the original vectors $\mathbf{X}, \mathbf{A}$ and $\mathbf{B}$, except for the NP task, in which the vectors $\mathbf{B}$ based on the 300-dimensional GloVe vectors are best. The effect of the vectors $\mathbf{Z}^*$ is slightly less impressive, when compared to the overcomplete vectors $\mathbf{X}^*$. The overcomplete word embeddings $\mathbf{Z}$ strongly differ from the word embeddings $\mathbf{X}$; hence, the denoising is affected. However, the performance of the vectors $\mathbf{Z}^*$ still outperforms the original vectors $\mathbf{X}, \mathbf{A}$ and $\mathbf{B}$ after the denoising process.

Relatedness and Similarity Tasks

For the relatedness task, we use two kinds of datasets: MEN (Bruni et al., 2014) consists of 3000 word pairs comprising 656 nouns, 57 adjectives and 38 verbs. The WordSim-353 relatedness dataset (Finkelstein et al., 2001) contains 252 word pairs. Concerning the similarity tasks, we evaluate the denoising vectors again on two kinds of datasets: *SimLex-999* (Hill et al., 2015) contains 999 word pairs including 666 noun, 222 verb and 111 adjective pairs. The WordSim-353 similarity dataset consists of 203 word pairs. In addition, we evaluate our denoising vectors on the WordSim-353 dataset which contains

[5]https://github.com/mfaruqui/sparse-coding

Vectors		Simlex-999 Corr.	MEN Corr.	WS353 Corr.	WS353-SIM Corr.	WS353-REL Corr.	ESL Acc.	TOEFL Acc.	NP Acc.
SG-100	**X**	33.7	72.9	69.7	74.5	65.5	48.9	62.0	72.8
	X*	33.2	72.8	70.6	74.8	66.0	53.0	**64.5**	78.5
	Z*	**35.9**	**74.4**	**71.2**	**75.2**	**68.1**	53.0	62.0	**79.1**
	A	32.5	69.8	65.5	69.5	60.2	**55.1**	51.8	78.8
	B	31.9	70.4	65.8	72.6	62.2	53.0	58.2	74.1
SG-300	**X**	36.1	74.7	71.0	75.9	66.1	**59.1**	72.1	77.9
	X*	**37.1**	**75.8**	**71.8**	**76.4**	**66.9**	**59.1**	74.6	**79.3**
	Z*	36.5	75.0	70.6	**76.4**	64.4	57.1	**77.2**	78.6
	A	32.9	72.4	67.5	71.9	63.4	53.0	65.8	78.3
	B	32.7	71.2	63.3	68.7	56.2	51.0	70.8	78.6
GloVe-100	**X**	29.7	69.3	52.9	60.3	49.5	46.9	82.2	76.4
	X*	**31.7**	**70.9**	**58.0**	**63.8**	**57.3**	53.0	**88.6**	**77.4**
	Z*	30.0	**70.9**	56.0	62.8	53.8	**57.0**	81.0	77.3
	A	30.7	70.7	54.9	62.2	51.2	55.1	78.4	77.1
	B	31.0	69.2	57.3	62.3	53.7	46.9	73.4	76.4
GloVe-300	**X**	37.0	74.8	60.5	66.3	57.2	**61.2**	89.8	74.3
	X*	**40.2**	**76.8**	**64.9**	**69.8**	**62.0**	**61.2**	**92.4**	76.3
	Z*	39.0	75.2	63.0	67.9	59.7	57.1	86.0	75.7
	A	36.7	74.1	61.5	67.7	57.8	55.1	87.3	79.9
	B	33.1	70.2	57.0	62.2	53.0	51.0	91.4	**80.0**

Table 6.1: Effects of word denoising embeddings. Vectors **X** represent the baselines; vectors **A** and **B** were suggested by Faruqui et al. (2015); the vector length **Z*** is equal to 10 times of vector length **X**.

353 pairs for both similarity and relatedness relations. We calculate cosine similarity between the vectors of two words forming a test pair, and report the Spearman rank-order correlation coefficient ρ (Siegel and Castellan, 1988) against the respective gold standards of human ratings.

Synonymy

We evaluate on 80 TOEFL (Test of English as a Foreign Language) synonym questions (Landauer and Dumais, 1997) and 50 ESL (English as a Second Language) questions (Turney, 2001). The first dataset represents a subset of 80 multiple-choice synonym questions from the TOEFL test: a word is paired with four options, one of which is a valid synonym. The second dataset contains 50 multiple-choice synonym questions, and the goal is to choose a valid synonym from four options. For each question, we compute the cosine similarity between the target word and the four candidates. The suggested answer is the candidate with the highest cosine score. We use accuracy to evaluate the performance.

Phrase parsing as Classification

Lazaridou et al. (2013) introduced a dataset of noun phrases (NP) in which each NP consists of three elements: the first element is either an adjective or a noun, and the other elements are all nouns. For a given NP (such as *blood pressure medicine*), the

task is to predict whether it is a left-bracketed NP, e.g., *(blood pressure) medicine*, or a right-bracketed NP, e.g., *blood (pressure medicine)*.

The dataset contains 2227 noun phrases split into 10 folds. For each NP, we use the average of word vectors as features to feed into the classifier by tuning the hyper-parameters (w_1, w_2 and w_3) for each element (e_1, e_2 and e_3) within the NP: $\vec{e}_{NP} = \frac{1}{3}(w_1\vec{e}_1 + w_2\vec{e}_2 + w_3\vec{e}_3)$. We then employ the classification of the NPs by using a SVM with Radial Basis Function kernel. The classifier is tuned on the first fold, and cross-validation accuracy is reported on the nine remaining folds.

6.3.4 Effects of Filter Depth

As mentioned above, the architecture of the filter f is a feed-forward network with a fixed depth T. For each stage T, the filter f attempts to reduce the noise within input vectors by approximating these vectors based on vectors of a previous stage $T-1$. In order to investigate the effects of each stage T, we use pre-trained GloVe vectors with 100 dimensions to evaluate the denoising performance of the vectors on detecting synonymy in the TOEFL dataset across several stages of T.

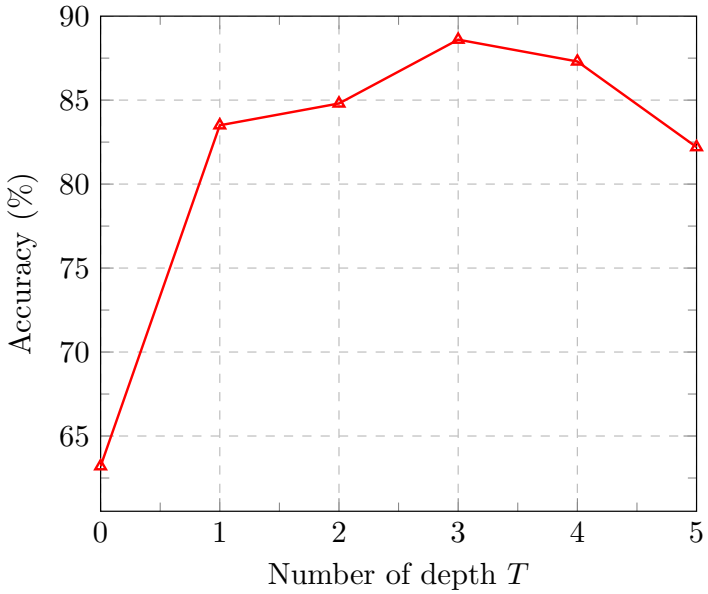


Figure 6.4: Effects of the filter with depth T on filtering noise.

The results are presented in Figure 6.4. The accuracy of synonymy detection increases sharply from 63.2% to 88.6% according to the number of stages T from 0 to 3. However, the denoising performance of vectors falls with the number of stages $T > 3$. This evaluation shows that the filter f with a consistently fixed depth T can be trained to efficiently filter noise for word embeddings. In other words, the number of stages T exceeds a consistent number T (with $T > 3$ in our case), leading to the loss of salient information in the vectors.

6.4 Summary

In this chapter, we have presented two novel models to improve word embeddings by reducing noise in state-of-the-art word embeddings models. To the best of our knowledge, we are the first to work on filtering out noise from word embeddings. The proposed models successfully filtered out noise from word embeddings to generate denoising word embeddings that are able to replace state-of-the-art word embeddings as inputs of NLP applications.

The underlying idea in our models was to make use of a filter to reduce noise. The filter is designed as a recursive feed-forward neural network with a particular depth. The first model generated complete word denoising embeddings whose length is equal to that of original word embeddings; the second model yielded overcomplete word denoising embeddings whose length tends to exceed that of original word embeddings.

In order to verify the effectiveness of word denoising embeddings, we made use of two state-of-the-art word embeddings, word2vec and GloVe, to learn word denoising embeddings. We evaluated word denoising embeddings on several benchmark tasks including similarity and relatedness tasks, synonymy detection task, and phrase parsing task. The experiments showed that the word denoising embeddings outperform the original state-of-the-art word embeddings on the benchmark tasks.

7 Evaluating Semantic Models of (Dis-)Similarity and Relatedness in Vietnamese

7.1 Introduction

Computational models that distinguish between semantic similarity and semantic relatedness (Budanitsky and Hirst, 2006) are important for many NLP applications, such as the automatic generation of dictionaries, thesauri, and ontologies (Biemann, 2005; Cimiano et al., 2005; Li et al., 2006), and machine translation (He et al., 2008; Marton et al., 2009). In order to evaluate these models, gold standard resources with word pairs have to be collected (typically across semantic relations such as synonymy, hypernymy, antonymy, co-hyponymy, and meronomy) and annotated for their degree of similarity via human judgements. These computational models are often proposed to solve NLP tasks in morphologically rich languages such as English, German, French, or Italian. Therefore, there is still room for transferring such computational models to low-resource languages.

The most prominent examples of gold standard similarity resources for English are the Rubenstein & Goodenough (RG) dataset (Rubenstein and Goodenough, 1965), the TOEFL test questions (Landauer and Dumais, 1997), WordSim-353 (Finkelstein et al., 2001), MEN (Bruni et al., 2012), SimLex-999 (Hill et al., 2015), and the lexical contrast datasets (Nguyen et al., 2016a, 2017b). For other languages, resource examples are the translation of the RG dataset to German (Gurevych, 2005), the German dataset of paradigmatic relations (Scheible and Schulte im Walde, 2014), and the translation of WordSim-353 and SimLex-999 to German, Italian and Russian (Leviant and Reichart, 2015). However, for low-resource languages there is still a lack of such datasets, which we aim to fill for Vietnamese, a language without morphological marking such as case, gender, number, and tense, thus differing strongly from Western European languages.

In contrast to previous chapters, the focus of this chapter is shifted to Vietnamese. In this chapter, we introduce two novel datasets for Vietnamese: a dataset of lexical contrast pairs **ViCon** to distinguish between similarity (synonymy) and dissimilarity (antonymy), and a dataset of semantic relation pairs **ViSim-400** to reflect the continuum between

similarity and relatedness. The two datasets are publicly available.[1] Furthermore, we transfer standard and neural co-occurrence models for English to Vietnamese to verify our novel datasets. As a result, we obtain behaviour similar to that of the corresponding English datasets *SimLex-999* (Hill et al., 2015) and the lexical contrast dataset (henceforth *LexCon*), cf. Nguyen et al. (2016a). The approaches and experiments described in this chapter are published in Nguyen et al. (2018).

7.2 Related Work

Over the years a number of datasets have been collected for studying and evaluating semantic similarity and semantic relatedness. For English, Rubenstein and Goodenough (Rubenstein and Goodenough, 1965) presented a small dataset (RG) of 65 noun pairs. For each pair, the degree of similarity in meaning was provided by 15 raters. The RG dataset is assumed to reflect similarity rather than relatedness. Finkelstein et al. (2001) created a set of 353 English noun-noun pairs (WordSim-353)[2], where each pair was rated by 16 subjects according to the degree of semantic relatedness on a scale from 0 to 10. Bruni et al. (2012) introduced a large test collection called MEN[3]. Similar to WordSim-353, the authors refer to both similarity and relatedness when describing the MEN dataset, although the annotators were asked to rate the pairs according to relatedness. Unlikely the construction of the RG and WordSim-353 datasets, each pair in the MEN dataset was only evaluated by one rater who ranked it for relatedness relative to 50 other pairs in the dataset. Recently, Hill et al. (2015) presented SimLex-999, a gold standard resource for the evaluation of semantic representations containing similarity ratings of word pairs across different part-of-speech categories and concreteness levels. The construction of SimLex-999 was motivated by two factors, (i) to consistently quantify similarity, as distinct from association, and apply it to various concept types, based on minimal intuitive instructions, and (ii) to have room for the improvement of state-of-the-art models which had reached or surpassed the human agreement ceiling on WordSim-353 and MEN, the most popular existing gold standards, as well as on RG. Scheible and Schulte im Walde (2014) presented a collection of semantically related word pairs for German and English,[4] which was compiled via Amazon Mechanical Turk (AMT)[5] human judgement experiments and comprises (i) a selection of targets across word classes balanced for semantic category, polysemy, and corpus frequency, (ii) a set of human-generated semantically related word pairs (synonyms, antonyms, hypernyms) based on the target units, and (iii) a subset of the generated word pairs rated for their

[1] www.ims.uni-stuttgart.de/data/vnese_sem_datasets
[2] www.cs.technion.ac.il/~gabr/resources/data/wordsim353
[3] clic.cimec.unitn.it/~elia.bruni/MEN
[4] www.ims.uni-stuttgart.de/data/sem-rel-database/
[5] www.mturk.com

relation strength, including positive and negative relation evidence.

For other languages, only a few gold standard sets with scored word pairs exist. Among others, Gurevych (2005) replicated Rubenstein and Goodenough's experiments after translating the original 65 word pairs into German. In later work, Gurevych (2006) used the same experimental setup to increase the number of word pairs to 350. Leviant and Reichart (2015) translated two prominent evaluation sets, WordSim-353 (association) and SimLex-999 (similarity) from English to Italian, German and Russian, and collected the scores for each dataset from the respective native speakers via crowdflower[6].

7.3 Dataset Design

7.3.1 Criteria

Semantic similarity is a narrower concept than *semantic relatedness* and holds between lexical terms with similar meanings. Strong similarity is typically observed for the lexical relations of synonymy and co-hyponymy. For example, in Vietnamese "đội" (*team*) and "nhóm" (*group*) represents a synonym pair; "ô_tô" (*car*) and "xe_đạp" (*bike*) is a co-hyponymy pair. More specifically, words in the pair "ô_tô" (*car*) and "xe_đạp" (*bike*) share several features such as physical (e.g. bánh_xe / *wheels*) and functional (e.g. vận_tải / *transport*), so that the two Vietnamese words are interchangeable regarding the kinds of transportation. The concept of semantic relatedness is broader and holds for relations such as holonymy, antonymy, functional association, and other "non-classical relations" (Morris and Hirst, 2004). For example, "xăng_dầu" (*petrol*) and "ô_tô" (*car*) represent a holonym pair. In contrast to similarity, this holonym pair expresses a clearly functional relationship; the words are strongly associated with each other but not similar.

Empirical studies have shown that the predictions of distributional models as well as humans are strongly related to the part-of-speech (POS) category of the learned concepts. Among others, Gentner (2006) showed that verb concepts are harder to learn by children than noun concepts.

Distinguishing antonymy from synonymy is one of the most difficult challenges. While antonymy represents words which are strongly associated but highly dissimilar to each other, synonymy refers to words that are highly similar in meaning. However, antonyms and synonyms often occur in similar context, as they are interchangeable in their substitution.

[6] www.crowdflower.com/

7.3.2 Resource for Concept Choice: Vietnamese Computational Lexicon

The Vietnamese Computational Lexicon (VCL)[7] (Nguyen et al., 2006) is a common linguistic database which is freely and easily exploitable for automatic processing of the Vietnamese language. VCL contains 35,000 words corresponding to 41,700 concepts, accompanied by morphological, syntactic and semantic information. The morphological information consists of 8 morphemes including simple word, compound word, reduplicative word, multi-word expression, loan word, abbreviation, bound morpheme, and symbol. For example, "bàn" (*table*) is a simple word with definition "đồ thường làm bằng gỗ, có mặt phẳng và chân đỡ ..." (*pieces of wood, flat and supported by one or more legs ...*). The syntactic information describes part-of-speech, collocations, and subcategorisation frames. The semantic information includes two types of constraints: logical and semantic. The logical constraint provides category meaning, synonyms and antonyms. The semantic constraint provides argument information and semantic roles. For example, "yêu" (*love*) is a verb with category meaning "*emotion*" and antonym "ghét" (*hate*).

VCL is the largest linguistic database of its kind for Vietnamese, and it encodes various types of morphological, syntactic and semantic information, so it presents a suitable starting point for the choice of lexical units for our purpose.

7.3.3 Choice of Concepts

Concepts in ViCon

The choice of related pairs in this dataset was drawn from VCL in the following way. We extracted all antonym and synonym pairs according to the three part-of-speech categories: noun, verb and adjective. We then randomly selected 600 adjective pairs (300 antonymous pairs and 300 synonymous pairs), 400 noun pairs (200 antonymous pairs and 200 synonymous pairs), and 400 verb pairs (200 antonymous pairs and 200 synonymous pairs). In each part-of-speech category, we balanced for the size of morphological classes in VCL, for both antonymous and synonymous pairs.

Concepts in ViSim-400

The choice of related pairs in this dataset was drawn from both the VLC and the *Vietnamese WordNet*[8] (VWN), cf. Nguyen et al. (2016c). We extracted all pairs of the three part-of-speech categories: noun, verb and adjective, according to five semantic relations: synonymy, antonymy, hypernymy, co-hoponymy and holonymy. We then sampled 400 pairs for the ViSim-400 dataset, accounting for 200 noun pairs, 150 verb pairs and 50

[7] https://vlsp.hpda.vn/demo/?page=vcl
[8] http://viet.wordnet.vn/wnms/

adjective pairs. Regarding noun pairs, we balanced the size of pairs in terms of six relations: the five extracted relations from VCL and VWN, and an "unrelated" relation. For verb pairs, we balanced the number of pairs according to five relations: synonymy, antonymy, hypernymy, co-hyponymy, and unrelated. For adjective pairs, we balanced the size of pairs for three relations: synonymy, antonymy, and unrelated. In order to select the unrelated pairs for each part-of-speech category, we paired the unrelated words from the selected related pairs at random. From these random pairs, we excluded those pairs that appeared in VCL and VWN. Furthermore, we also balanced the number of selected pairs according to the sizes of the morphological classes and the lexical categories.

7.3.4 Annotation of ViSim-400

For rating ViSim-400, 200 raters who were native Vietnamese speakers were paid to rate the degrees of similarity for all 400 pairs. Each rater was asked to rate 30 pairs on a 0–6 scale; and each pair was rated by 15 raters. Unlike other datasets which performed the annotation via Amazon Mechanical Turk, each rater for ViSim-400 conducted the annotation via a survey which detailed the exact annotation guidelines.

The structure of the questionnaire was motivated by the SimLex-999 dataset: we outlined the notion of similarity via the well-understood idea of the six relations included in the ViSim-400 dataset. Immediately after the guidelines of the questionnaire, a checkpoint question was posed to the participants to test whether the person understood the guidelines: the participant was asked to pick the most similar word pair from three given word pairs, such as kiêu_căng/kiêu_ngạo (*arrogant/cocky*) vs. trầm/bổng (*high/low*) vs. cổ_điển/biếng (*classical/lazy*). The annotators then labeled the kind of relation and scored the degree of similarity for each word pair in the survey.

7.3.5 Agreement in ViSim-400

We analysed the ratings of the ViSim-400 annotators with two different inter-annotator agreement (IAA) measures, Krippendorff's alpha coefficient (Krippendorff, 2004), and the average standard deviation (STD) of all pairs across word classes. The first IAA measure, IAA-pairwise, computes the average pairwise Spearman's ρ correlation between any two raters. This IAA measure has been a common choice in previous data collections in distributional semantics (Padó et al., 2007; Reisinger and Mooney, 2010; Hill et al., 2015). The second IAA measure, IAA-mean, compares the average correlation of the human raters with the average of all other raters. This measure would smooth individual annotator effects, and serve as a more appropriate *"upper bound"* for the performance of automatic systems than IAA-pairwise (Vulić et al., 2017). Finally, Krippendorff's α coefficient reflects the disagreement of annotators rather than their agreement, in addition to correcting for agreement by chance.

	All	**Noun**	**Verb**	**Adjective**
IAA-Mean ρ	0.86	0.86	0.86	0.78
IAA-Pairwise ρ	0.79	0.76	0.78	0.75
Krippendorff's α	0.78	0.76	0.78	0.86
STD	0.87	0.87	0.90	0.82

Table 7.1: Inter-annotator agreements measured by Spearman's ρ, Krippendorff's α, and the average standard deviation (STD) of all pairs across word classes.

Table 7.1 shows the inter-annotator agreement values, Krippendorff's α coefficient, and the response consistency measured by STD over all pairs and different word classes in ViSim-400. The overall IAA-pairwise of ViSim-400 is $\rho = 0.79$, comparing favourably with the agreement on the SimLex-999 dataset ($\rho = 0.67$ using the same IAA-pairwise measure). Regarding IAA-mean, ViSim-400 also achieves an overall agreement of $\rho = 0.86$, which is similar to the agreement in Vulić et al. (2017), $\rho = 0.86$. For Krippendorff's α coefficient, the value achieves $\alpha = 0.78$, also reflecting the reliability of the annotated dataset.

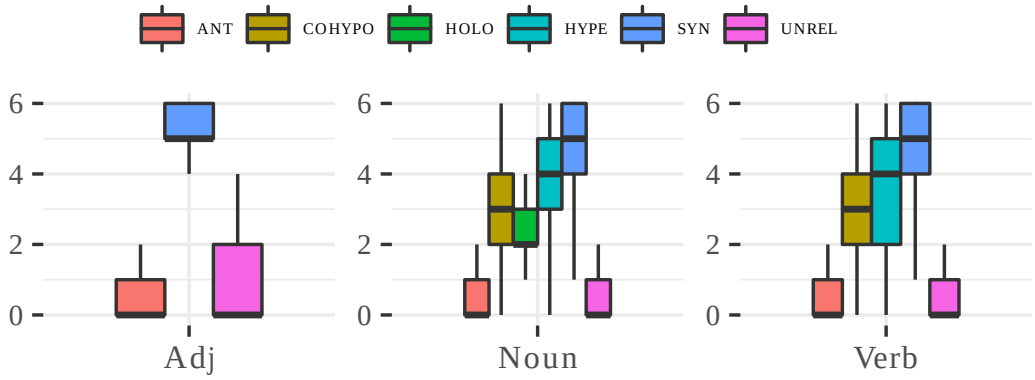


Figure 7.1: Distribution of scored pairs in ViSim-400 across parts-of-speech and semantic relations.

Furthermore, the box plots in Figure 7.1 present the distributions of all rated pairs in terms of the fine-grained semantic relations across word classes. They reveal that –across word classes– synonym pairs are clearly rated as the most similar words, and antonym as well as unrelated pairs are clearly rated as the most dissimilar words. Hypernymy, co-hyponymy and holonymy are in between, but rather similar than dissimilar.

7.4 Verification of Datasets

In this section, we verify our novel datasets ViCon and ViSim-400 through standard and neural co-occurrence models, in order to show that we obtain a similar behaviour as for the corresponding English datasets.

7.4.1 Verification of ViSim-400

We adopt a comparison of neural models on SimLex-999 as suggested by Nguyen et al. (2016a). They applied three models, a Skip-gram model with negative sampling SGNS (Mikolov et al., 2013a), the dLCE model (Nguyen et al., 2016a), and the mLCM model (Pham et al., 2015). Both the dLCE and the mLCM models integrated lexical contrast information into the basic Skip-gram model to train word embeddings for distinguishing antonyms from synonyms, and for reflecting degrees of similarity.

The three models were trained with 300 dimensions, a window size of 5 words, and 10 negative samples. Regarding the corpora, we relied on Vietnamese corpora with a total of ≈145 million tokens, including the *Vietnamese Wikipedia*,[9] *VNESEcorpus* and *VNTQcorpus*,[10] and the *Leipzig Corpora Collection for Vietnamese*[11] (Goldhahn et al., 2012). For word segmentation and POS tagging, we used the open-source toolkit *UETnlp*[12] (Nguyen and Le, 2016). The antonym and synonym pairs to train the dLCE and mLCM models were extracted from VWN consisting of 49,458 antonymous pairs and 338,714 synonymous pairs. All pairs which appeared in ViSim-400 were excluded from this set.

Table 7.2 shows Spearman's correlations ρ, comparing the scores of the three models with the human judgements for ViSim-400. As also reported for English, the dLCE model produces the best performance, SGNS the worst.

	SGNS	**mLCM**	**dLCE**
ViSim-400	0.37	0.60	**0.62**
SimLex-999	0.38	0.51	**0.59**

Table 7.2: Spearman's correlation ρ on ViSim-400 in comparison to SimLex-999, cf. Nguyen et al. (2016a).

In a second experiment, we computed the cosine similarities between all word pairs, and used the area under curve (AUC) to distinguish between antonyms and synonyms. Table 7.3 presents the AUC results of the three models. Again, the models show a similar behaviour in comparison to SimLex-999, where also the dLCE model outperforms the two other models, and the SGNS model is by far the worst.

7.4.2 Verification of ViCon

In order to verify ViCon, we applied three co-occurrence models to rank antonymous and synonymous word pairs according to their cosine similarities: two standard co-

[9] `https://dumps.wikimedia.org/viwiki/latest/`
[10] `http://viet.jnlp.org/download-du-lieu-tu-vung-corpus`
[11] `http://wortschatz.uni-leipzig.de/en/download`
[12] `https://github.com/phongnt570/UETnlp`

	Model	Noun	Verb	Adj
ViSim-400	SGNS	0.66	0.63	0.70
	mLCM	0.81	0.92	0.96
	dLCE	**0.92**	**0.95**	**0.98**
SimLex-999	SGNS	0.66	0.65	0.64
	mLCM	0.69	0.71	0.85
	dLCE	**0.72**	**0.81**	**0.90**

Table 7.3: AUC scores for distinguishing antonyms from synonyms in ViSim-400.

occurrence models based on positive point-wise mutual information (PPMI) and positive local mutual information (PLMI) (Evert, 2005) as well as an improved feature value representation **weight**$^{\mathbf{SA}}$ as suggested by Nguyen et al. (2016a). For building the vector space co-occurrence models, we relied on the same Vietnamese corpora as in the previous section. For inducing the word vector representations via **weight**$^{\mathbf{SA}}$, we made use of the antonymous and synonymous pairs in VWN, as in the previous section, and then removed all pairs which appeared in ViCon. Optionally, we applied singular value decomposition (SVD) to reduce the dimensionalities of the word vector representations.

As in Nguyen et al. (2016a), we computed the cosine similarities between all word pairs, and then sorted the pairs according to their cosine scores. Average Precision (AP) evaluated the three vector space models. Table 7.4 presents the results of the three vector space models with and without SVD. As for English, the results on the Vietnamese dataset demonstrate significant improvements (χ^2,* $p < .001$) of **weight**$^{\mathbf{SA}}$ over PPMI and PLMI, both with and without SVD, and across word classes.

	Metric	ADJ		NOUN		VERB	
		SYN	ANT	SYN	ANT	SYN	ANT
ViCon	PPMI	0.70	0.38	0.68	0.39	0.69	0.38
	PLMI	0.59	0.44	0.61	0.42	0.63	0.41
	weight$^{\mathbf{SA}}$	**0.93***	**0.31***	**0.94***	**0.31**	**0.96**	**0.31**
	PPMI + SVD	0.76	0.36	0.66	0.40	0.81	0.34
	PLMI + SVD	0.49	0.51	0.55	0.46	0.51	0.49
	weight$^{\mathbf{SA}}$ + SVD	**0.91***	**0.32***	**0.81***	**0.34***	**0.92***	**0.32***
LexCon	PLMI	0.56	0.46	0.60	0.42	0.62	0.42
	weight$^{\mathbf{SA}}$	**0.75**	**0.36**	**0.66**	**0.40**	**0.71**	**0.38**
	PLMI + SVD	0.55	0.46	0.55	0.46	0.58	0.44
	weight$^{\mathbf{SA}}$ + SVD	**0.76***	**0.36***	**0.66**	**0.40**	**0.70***	**0.38***

Table 7.4: AP evaluation of co-occurrence models on ViCon in comparison to LexCon (Nguyen et al., 2016a).

7.5 Summary

This chapter introduced two novel datasets for the low-resource language of Vietnamese to assess models of semantic similarity and relatedness. The first dataset, namely *ViCon*, comprises synonym and antonym pairs across the word classes of nouns, verbs, and adjectives. The word pairs of this dataset were extracted from VCL by balancing for the size of morphological classes in each part-of-speech category. Hence *ViCon* offers data to distinguish between similarity and dissimilarity. The second dataset, namely *ViSim-400*, contains 400 word pairs across the three word classes and five semantic relations. The choice of related pairs in this dataset was drawn from both VCL and VWN. Similar to the *ViCon* dataset, the number of selected pairs in *ViSim-400* is also balanced according to the size of the morphological classes and the lexical categories. Furthermore, each pair of *ViSim-400* was rated by human judges for its degree of similarity to reflect the continuum between similarity and relatedness.

In addition, we successfully adopted standard co-occurrence and neural network models in English for Vietnamese in order to verify the two proposed datasets. A series of experiments showed that results of these models in our two datasets are comparable to the respective English datasets.

8 Conclusion and Future Work

This chapter first summarizes the findings and contributions of this thesis. We then outline some ideas for future work to address the limitations of this thesis and to go in other directions.

8.1 Conclusion

This thesis investigated strategies to improve computational models that distinguish antonymy, synonymy, and hypernymy; and that measure semantic similarity and relatedness. Moreover, this thesis also focused on evaluating computational models on the low-resource language of Vietnamese.

The new weighted feature $weight^{SA}(w, f)$, which is used to construct distributional word vector representations, allowed computational models to significantly distinguish antonymy and synonymy by relying on distributional lexical contrast, making synonymous word pairs closer to each other, and forcing antonymous word pairs further away from each other (chapter 3). Similarly, the distributional lexical contrast was also successfully integrated into the Skip-gram model to learn *dLCE* embeddings that are better able to capture degrees of similarity and identify antonymy from synonymy, compared to other state-of-the-art word embeddings (chapter 3). Moreover, in chapter 4, the two pattern-based architectures of *AntSynNET* neural network showed an improvement in distinguishing antonymy from synonymy over prior pattern-based models. The lexico-syntactic patterns of antonymous and synonymous word pairs induced from syntactic parse trees mitigated the sparsity issue relative to standard lexico-syntactic patterns of antonymy and synonymy. Moreover, the use of the *distance* feature in lexico-syntactic patterns also helped *AntSynNET* models to better differentiate antonymy from synonymy.

In chapter 5, the hierarchical embeddings *HyperVec* significantly outperformed state-of-the-art measures and word embeddings on the tasks of hypernymy detection and directionality. By relying on hypernymy information from lexical resources, *HyperVec* was able to (i) strengthen cosine similarity of hypernymy, which is higher than cosine similarity of other semantic relations; and (ii) generate the distributional hierarchy between hyponyms and hypernyms. These two aspects of *HyperVec* were formulated as an unsupervised hypernymy measure named *HyperScore* which showed significant improve-

ments over state-of-the-art measures. Additionally, *HyperVec* also showed the ability to generalize hypernymy over other languages including German and Italian.

Regarding computational models for measuring similarity and relatedness, in comparison with state-of-the-art word embeddings, word denoising embeddings strengthened by filtering out noise from state-of-the-art word embeddings achieved improvements in several benchmark tasks including word similarity and relatedness tasks, synonymy detection, and noun phrase classification (chapter 6). The word denoising embeddings were learned via filters without using any external resources, with the architectures of filters being feed-forward, non-linear and parameterized neural networks with a fixed depth. Furthermore, in chapter 7, two datasets were introduced for the low-resource language of Vietnamese. The first dataset *ViCon* offered data for computational models to distinguish similarity and dissimilarity. The second dataset *ViSim-400*, which consisted of human-rated word pairs, reflected the continuum between semantic similarity and semantic relatedness. The standard co-occurrence and neural models were evaluated on these two Vietnamese datasets, showing behaviour similar to that of the corresponding English datasets in terms of distinguishing and measuring semantic similarity and semantic relatedness.

8.2 Future Work

Fine-grained antonymy classification: in chapter 3 and chapter 4, we proposed approaches to distinguish antonymy and synonymy by using both vector representations and neural networks. In those approaches, we assigned all types of antonymy to be only a single antonymy class. However, antonymy could be categorised into several types such as gradable antonymy, non-gradable antonymy. Hence, possible future work could focus on the task of fine-grained antonymy classification. The pattern-based neural models presented in chapter 4 can be adopted to address the task of fine-grained antonymy classification. Instead of classifying antonymy and synonymy, the models only focus on classifying types of antonymy. Moreover, the lexico-syntactic patterns that indicate the simple paths of antonymous word pairs are expected to be able to distinguish between types of antonymy. An example could be that the non-gradable antonymy pair (*dead / alive*) cannot appear in the pattern of "*neither* **X** *nor* **Y**." Nevertheless, rather than using English, this possible future work could be applied for other languages such as German, Spanish, Italian, or low-resource languages.

Fine-grained semantic relations classification: we have presented approaches to distinguish antonymy, synonymy, and hypernymy in chapters 3, 4, and 5. In future work in this direction, we could extend the proposed approaches to work with the task of fine-grained semantic relations classification, in which semantic relations could be classified

as antonymy, synonymy, hypernymy, meronymy, or co-hyponymy. Specifically, we will make use of word pairs of semantic relations to induce their lexico-syntactic patterns from the corpora. We will then use pattern-based neural models to encode lexico-syntactic patterns as vector representations for fine-grained semantic relations classification.

Hierarchical embeddings for named entities: in chapter 5, we introduced an approach to learn hierarchical embeddings *HyperVec* for hypernymy. One of the most important characteristics of *HyperVec* is that it generates the distributional hierarchy between hyponyms and hypernyms. Thus an idea for future work in this direction is to extend the *HyperVec* model to learn hierarchical embeddings for named entities. For example, this kind of hierarchical embeddings can be learned to encode the distributional hierarchy between a country and its cities such as the United States and Los Angeles. Furthermore, in knowledge-based encyclopediae such as Wikipedia, each named entity in a given encyclopedic article is described by referring to many other named entities. This kind of information is similar to the organisation of hypernymy in WordNet, in which a hierarchy exists between entities. The hierarchical embeddings for named entities can be utilized for NLP tasks, such as entity linking that links surface forms of named entities in a document to named entities in a reference knowledge base, and entity search that retrieves knowledge directly by generating a list of relevant entities in response to a search request.

Bibliography

Adel, H. and Schütze, H. (2014). Using mined coreference chains as a resource for a semantic task. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1447–1452, Doha, Qatar.

Baroni, M., Bernardi, R., Do, N.-Q., and chieh Shan, C. (2012). Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32, Avignon, France.

Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics (GEMS)*, pages 1–10, Edinburgh, Scotland.

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

Benotto, G. (2015). *Distributional models for semantic relations: A study on hyponymy and antonymy*. PhD thesis, University of Pisa.

Biemann, C. (2005). Ontology learning from text: A survey of methods. *LDV Forum*, 20(2):75–93.

Biran, O. and McKeown, K. (2013). Classifying taxonomic relations between pairs of wikipedia articles. In *Proceddings of Sixth International Joint Conference on Natural Language Processing (IJCNLP)*, pages 788–794, Nagoya, Japan.

Botha, J. A. and Blunsom, P. (2014). Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages 1899–1907, Beijing, China.

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*, pages 136–145, Jeju Island, Korea.

Bruni, E., Tran, N., and Baroni, M. (2014). Multimodal distributional semantics. *Journal of Artifical Intelligence Research (JAIR)*, 49:1–47.

Budanitsky, A. and Hirst, G. (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47.

Charles, W. G. and Miller, G. A. (1989). Contexts of antonymous adjectives. *Applied Psychology*, 10:357–375.

Chen, W., Grangier, D., and Auli, M. (2016). Strategies for training large vocabulary

neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1975–1985, Berlin, Germany.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Int. Res.*, 24(1):305–339.

Clark, E. V. (1992). Conventionality and contrast: Pragmatic principles with lexical consequences. *Adrienne Lehrer and Eva Fedder Kittay*, pages 171–188.

Clarke, D. (2009). Context-theoretic semantics for natural language: An overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics (GEMS)*, pages 112–119, Athens, Greece.

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*, pages 160–167, Helsinki, Finland.

Dagan, I., Roth, D., Sammons, M., and Zanzotto, F. M. (2013). *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Deese, J. (1965). *The Structure of Associations in Language and Thought*. The John Hopkins Press, Baltimore, MD.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 644–648, Atlanta, USA.

Elman, J. L. (1990). Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

Evert, S. (2005). *The Statistics of Word Cooccurrences*. PhD thesis, Stuttgart University.

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., and Smith, N. A. (2015). Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1491—-1500, Beijing, China.

Fellbaum, C. (1995). Co-occurrence and antonymy. *International Journal of Lexicography*, 8:281–303.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2001). Placing search in context: The concept revisited. In *Proceedings*

of the 10th International Conference on the World Wide Web, pages 406–414.

Firth, J. R. (1957). *Papers in Linguistics 1934-51*. Longmans, London, UK.

Geffet, M. and Dagan, I. (2005). The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 107–114, Michigan, US.

Gentner, D. (2006). Why verbs are hard to learn. In Hirsh-Pasek, K. A. and Golinkoff, R. M., editors, *Action meets word: How Children Learn Verbs*, pages 544–564. Oxford University Press.

Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765.

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel*, pages 399–406.

Gurevych, I. (2005). Using the structure of a conceptual network in computing semantic relatedness. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 767–778, Jeju Island, Republic of Korea.

Gurevych, I. (2006). Thinking beyond the nouns - computing semantic relatedness across parts of speech. In *Sprachdokumentation & Sprachbeschreibung, 28. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft*, page 226, Bielefeld, Germany.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, Sardinia, Italy.

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

He, X., Yang, M., Gao, J., Nguyen, P., and Moore, R. (2008). Indirect-hmm-based hypothesis alignment for combining outputs from machine translation systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 98–107, Honolulu, Hawaii.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France.

Hill, F., Kiela, D., and Korhonen, A. (2013). Concreteness and corpora: A theoretical and practical study. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 75–83, Sofia, Bulgaria.

Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computa-*

tion, 9(8):1735–1780.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

Justeson, J. S. and Katz, S. M. (1991). Co-occurrences of antonymous adjectives and their contexts. *Computational Linguistics*, 17:1–19.

Kiela, D. and Clark, S. (2014). A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30, Gothenburg, Sweden.

Kiela, D., Rimell, L., Vulić, I., and Clark, S. (2015). Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 119–124, Beijing, China.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 3294–3302, Montreal, Canada.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-Geffet, M. (2010). Directional distributional similarity for lexical inference. *Natural Language Processing*, 16(4):359–389.

Krippendorff, K. (2004). *Content Analysis: An Introduction to its Methodology*. Sage Publications.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Lazaridou, A., Dinu, G., and Baroni, M. (2015). Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 270–280, Beijing, China.

Lazaridou, A., Vecchi, E. M., and Baroni, M. (2013). Fish transporters and miracle

homes: How compositional distributional semantics can help NP parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1908—-1913, Doha, Qatar.

Lenci, A. and Benotto, G. (2012). Identifying hypernyms in distributional semantic spaces. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval)*, pages 75–79, Montréal, Canada.

Leviant, I. and Reichart, R. (2015). Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR*, abs/1508.00106.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

Levy, O., Remus, S., Biemann, C., and Dagan, I. (2015). Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 970–976, Denver, Colorado.

Li, M., Zhang, Y., Zhu, M., and Zhou, M. (2006). Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 1025–1032, Sydney, Australia.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 768–774, Montréal, Canada.

Lin, D. and Pantel, P. (2001). Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, California.

Lin, D., Zhao, S., Qin, L., and Zhou, M. (2003). Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1492–1493, Acapulco, Mexico.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Lyons, J. (1977). *Semantics*. Cambridge University Press.

Marton, Y., Callison-Burch, C., and Resnik, P. (2009). Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 381–390, Singapore.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *Computing Research Repository*, abs/1301.3781.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting Similarities among Lan-

guages for Machine Translation. *CoRR*, abs/1309.4168.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. A. and Fellbaum, C. (1991). Semantic networks of English. *Cognition*, 41:197–229.

Mohammad, S. M., Dorr, B. J., Hirst, G., and Turney, P. D. (2013). Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252.

Morris, J. and Hirst, G. (2004). Non-classical lexical semantic relations. In *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*, pages 46–51, Boston, Massachusetts.

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Gašić, M., Rojas-Barahona, M. L., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 142–148, San Diego, California.

Murphy, G. (2002). *The Big Book of Concepts*. MIT Press, Cambridge, MA, USA.

Murphy, M. L. (2003). *Semantic Relations and the Lexicon*. Cambridge University Press.

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

Navigli, R., Velardi, P., and Faralli, S. (2011). A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1872–1877, Barcelona, Catalonia, Spain.

Nguyen, K. A., Köper, M., Schulte im Walde, S., and Vu, N. T. (2017a). Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243, Copenhagen, Denmark.

Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2016a). Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.

Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2016b). Neural-based noise filtering from word embeddings. In *Proceedings of the 26th International Conference on*

Computational Linguistics (COLING), pages 2699–2707, Osaka, Japan.

Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2017b). Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 76–85, Valencia, Spain.

Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2018). Introducing two vietnamese datasets for evaluating semantic models of (dis-)similarity and relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HTL)*, pages 199–205, New Orleans, Louisiana.

Nguyen, P.-T., Pham, V.-L., Nguyen, H.-A., Vu, H.-H., Tran, N.-A., and Truong, T.-T.-H. (2016c). A two-phase approach for building vietnamese wordnet. In *Proceedings of the Eighth Global WordNet Conference*, pages 259–264, Bucharest, Romania.

Nguyen, T. M. H., Romary, L., Rossignol, M., and Vu, X. L. (2006). A lexicon for Vietnamese language processing. *Language Resources and Evaluation*, 40(3-4):291–309.

Nguyen, T.-P. and Le, A.-C. (2016). A hybrid approach to vietnamese word segmentation. In *2016 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 114–119, Hanoi, Vietnam.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Ono, M., Miwa, M., and Sasaki, Y. (2015). Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989, Denver, Colorado.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Padó, S., Padó, U., and Erk, K. (2007). Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of EMNLP/CoNLL 2007*, Prague, Czech Republic.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet: : Similarity - measuring the relatedness of concepts. In *Proceedings of the 19th National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence (AAAI)*, pages 1024–1025, California, USA.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Pham, N. T., Lazaridou, A., and Baroni, M. (2015). A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the*

Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 21–26, Beijing, China.

Phan, X.-H., Nguyen, L.-M., and Horiguchi, S. (2008). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*, pages 91–100, Beijing, China.

Reisinger, J. and Mooney, R. (2010). A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Cambridge, Massachusetts.

Rimell, L. (2014). Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 511–519, Gothenburg, Sweden.

Roller, S., Erk, K., and Boleda, G. (2014). Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 1025–1036, Dublin, Ireland.

Roth, M. and Schulte im Walde, S. (2014). Combining word patterns and discourse markers for paradigmatic relation classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 524–530, Baltimore, MD.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Santus, E., Lenci, A., Chiu, T., Lu, Q., and Huang, C. (2016). Unsupervised measure of word similarity: How to outperform co-occurrence and vector cosine in vsms. In *Proceedings of the Thirtieth Conference on Artificial Intelligence AAAI)*, pages 4260–4261, Arizona, USA.

Santus, E., Lenci, A., Lu, Q., and Schulte im Walde, S. (2014a). Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 38–42, Gothenburg, Sweden.

Santus, E., Lu, Q., Lenci, A., and Huang, C. (2014b). Taking antonymy mask off in vector space. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation*, pages 135–144.

Santus, E., Yung, F., Lenci, A., and Huang, C.-R. (2015). Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, Beijing, China.

Schäfer, R. (2015). Processing and querying large web corpora with the COW14 ar-

chitecture. In *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora*, pages 28–34.

Schäfer, R. and Bildhauer, F. (2012). Building large corpora from the web using a new efficient tool chain. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 486–493, Istanbul, Turkey.

Scheible, S., im Walde, S. S., and Springorum, S. (2013). Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 489–497, Nagoya, Japan.

Scheible, S. and Schulte im Walde, S. (2014). A Database of Paradigmatic Semantic Relation Pairs for German Nouns, Verbs, and Adjectives. In *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, pages 111–119, Dublin, Ireland.

Schulte im Walde, S. and Köper, M. (2013). Pattern-based distinction of paradigmatic relations for german nouns, verbs, adjectives. In *Proceedings of the 25th International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 189–198, Darmstadt, Germany.

Schulte im Walde, S., Müller, S., and Roller, S. (2013). Exploring vector space models to predict the compositionality of german noun-noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 255–265, Atlanta, Georgia, USA.

Schwartz, R., Reichart, R., and Rappoport, A. (2015). Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the 19th Conference on Computational Language Learning (CoNLL)*, pages 258—-267, Beijing, China.

Shwartz, V., Goldberg, Y., and Dagan, I. (2016). Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2389–2398, Berlin, Germany.

Shwartz, V., Santus, E., and Schlechtweg, D. (2017). Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Siegel, S. and Castellan, N. J. (1988). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Boston, MA.

Snow, R., Jurafsky, D., and Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 801–808, Sydney, Australia.

Sucameli, I. (2015). Analisi computazionale delle relazioni semantiche: Uno studio della

lingua italiana. B.s. thesis, University of Pisa.

Szlam, A. D., Gregor, K., and Cun, Y. L. (2011). Structured sparse coding via lateral inhibition. *Advances in Neural Information Processing Systems (NIPS)*, 24:1116–1124.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Tuan, L. A., Tay, Y., Hui, S. C., and Ng, S. K. (2016). Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 403–413, Austin, Texas.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, pages 491–502.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Vendrov, I., Kiros, R., Fidler, S., and Urtasun, R. (2016). Order-embeddings of images and language. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Vilnis, L. and McCallum, A. (2015). Word representations via gaussian embedding. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, California, USA.

Voorhees, E. M. and Harman, D. K. (1999). The 7th Text REtrieval Conference (trec-7). *National Institute of Standards and Technology, NIST*.

Vu, N. T., Adel, H., Gupta, P., and Schütze, H. (2016). Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 534–539.

Vulić, I., Gerz, D., Kiela, D., Hill, F., and Korhonen, A. (2017). Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

Weeds, J., Clarke, D., Reffin, J., Weir, D. J., and Keller, B. (2014). Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 2249–2259, Dublin, Ireland.

Weeds, J. and Weir, D. (2003). A general framework for distributional similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–88, Stroudsburg, PA, USA.

Weeds, J., Weir, D., and McCarthy, D. (2004). Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 1015—-1021, Geneva, Switzerland.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages

133–138, Las Cruces, New Mexico.

Yih, W., Zweig, G., and Platt, J. C. (2012). Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 1212–1222, Jeju Island, Korea.

Yu, Z., Wang, H., Lin, X., and Wang, M. (2015). Learning term embeddings for hypernymy identification. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, pages 1390–1397, Buenos Aires, Argentina.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Zhitomirsky-Geffet, M. and Dagan, I. (2009). Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35(3):435–461.