Compositional Matrix-Space Models: Learning Methods and Evaluation

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Shima Asaadi

geboren am 07. April 1990 in Zanjan, Iran

eingereicht am 03. Februar 2020
verteidigt am 08. Juni 2020

Gutachter:
Prof. Dr. rer. nat. Sebastian Rudolph
Technische Universität Dresden

Dr. Michael Roth
Universität Stuttgart

Dresden, im September 2020

Abstract

There has been a lot of research on machine-readable representations of words for natural language processing (NLP). One mainstream paradigm for the word meaning representation comprises vector-space models obtained from the distributional information of words in the text. Machine learning techniques have been proposed to produce such word representations for computational linguistic tasks. Moreover, the representation of multi-word structures, such as phrases, in vector space can arguably be achieved by composing the distributional representation of the constituent words. To this end, mathematical operations have been introduced as composition methods in vector space. An alternative approach to word representation and semantic compositionality in natural language has been compositional matrix-space models. In this thesis, two research directions are considered. In the first, considering compositional matrix-space models, we explore word meaning representations and semantic composition of multi-word structures in matrix space. The main motivation for working on these models is that they have shown superiority over vector-space models regarding several properties. The most important property is that the composition operation in matrix-space models can be defined as standard matrix multiplication; in contrast to common vector space composition operations, this is sensitive to word order in language. We design and develop machine learning techniques that induce continuous and numeric representations of natural language in matrix space. The main goal in introducing representation models is enabling NLP systems to understand natural language to solve multiple related tasks. Therefore, first, different supervised machine learning approaches to train word meaning representations and capture the compositionality of multi-word structures using the matrix multiplication of words are proposed. The performance of matrix representation models learned by machine learning techniques is investigated in solving two NLP tasks, namely, sentiment analysis and compositionality detection. Then, learning techniques for learning matrix-space models are proposed that introduce generic task-agnostic representation models, also called word matrix embeddings. In these techniques, word matrices are trained using the distributional information of words in a given text corpus. We show the effectiveness of these models in the compositional representation of multi-word structures in natural language.

The second research direction in this thesis explores effective approaches for evaluating the capability of semantic composition methods in capturing the meaning representation of compositional multi-word structures, such as phrases. A common evaluation approach is examining the ability of the methods in capturing the semantic relatedness between linguistic units. The underlying assumption is that the more accurately a method of semantic composition can determine the representation of a phrase, the more accurately it can determine the relatedness of that phrase with other phrases. To apply the semantic relatedness approach, gold standard datasets have been introduced. In this thesis, we identify the limitations of the existing datasets and develop a new gold standard semantic relatedness dataset, which addresses the issues of the existing datasets. The proposed dataset allows us to evaluate meaning composition in vector- and matrix-space models.

Acknowledgements

It was a great privilege to work with Sebastian Rudolph. Without his advice and encouragement, I would have had a much harder time forming some of the fundamental questions of this dissertation, and an even harder time addressing them. I thank him for his support, helping me develop ideas, and working patiently on my writing.

I am deeply grateful to Dagmar Gromann for her sound advice, encouraging feedback, her readiness to help at any time, and lots of interesting discussions. She was incredibly supportive and helped move this research in interesting directions. I thank her for proofreading this dissertation and giving constructive feedback and suggestions for improving it.

A special thanks goes to Saif Mohammad and Svetlana Kiritchenko for giving me the opportunity to visit and collaborate with them on a very fascinating topic. I want to thank them for their gentle guidance and teaching me new ways of thinking. I want to thank Peter Turney for helpful discussions on this research project.

I am grateful for the opportunity of being a scholarship holder of the Research Training Group QuantLA. I want to thank my second supervisor, Heiko Vogler, for his constructive guidance.

Further, I would like to thank Michael Roth for taking the time to review this dissertation. I thank the doctoral committee.

I thank my colleagues in the computational logic group. I had a great time with them.

I wouldn't be where I am today without the amazing support, encouragement, and love from my parents, Robab and Houshang, and my sweet sister and brother-in-law, Shirin and Mahmoud. Thank you for always being there for me and supporting me in every phase of my life. I want to thank my parents-in-law for their love and support.

And last but never least, I thank my husband Danial, whom I am so lucky to have in my life. I thank him for our many wonderful years, his love, and for letting me follow my dreams. There aren't words to express my gratitude for having him in my life. Thank you!

Contents

Acronyms

BiRD Bigram Semantic Relatedness Dataset. 8

BOWs Bag-Of-Words. 17

BWS Best–Worst Scaling. 9

CBOW Continuous Bag Of Words. 22

CDSMs Compositional Distributional Semantic Models. 4

CMOW Continual Multiplication Of Words. 101

CMSMs Compositional Matrix-Space Models. 4

CNNs Convolutional Neural Networks. 19

COSMo Compositional Order-Sensitive Matrix Model. 10

DSMs Distributional Semantic Models. 4

LSTMs Long Short-Term Memory. 64

MPQA Multi-Perspective Question Answering. 56

MSE Mean Squared Error. 21

MWEs Multi-Word Expressions. 41

NLP natural language processing. iii

NLTK Natural Language ToolKit. 80

NNs Neural Networks. 19

NPMI Normalized Point-wise Mutual Information. 98

OLogReg Ordered Logistic Regression. 57

PMI Point-wise Mutual Information. 9

POS Part-Of-Speech. 46

PPMI Positive Pointwise Mutual Information. 90

RBF-SVR Radial Basis Function kernel-based Support-Vector Regression. 62

RNNs Recurrent Neural Networks. 19

SCL-OPP Sentiment Composition Lexicon with Opposing Polarity Phrases. 56

SE Squared Euclidean. 65

SHR Split-Half Reliability. 9

SSE Summed Squared Error. 51

STS Semantic Textual Similarity. 10

SVD Singular Value Decomposition. 91

VSMs Vector-Space Models. 3

WFA Weighted Finite Automata. 5

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

As the amount of text data that humans produce grows overall, the need to intelligently and automatically process it in order to extract different types of knowledge also increases (Socher, 2014). Computational linguistics is the study of how to understand, represent, and produce human (natural) language from a computational perspective. Natural language processing (NLP) is an interdisciplinary field that draws on artificial intelligence and computational linguistics and refers to the automatic computational processing of human languages. It explores how human language can be intelligently and automatically processed using various automated techniques. In general, it deals with the interaction between computers and humans, and the purpose is to make computers (machines) capable of understanding the human language. The ultimate goal of computational linguistics and NLP is to build autonomous machines that can communicate freely in natural language (Hausser, 2001).

NLP is very challenging, as human language is ambiguous and always changing (Goldberg, 2017). For instance, "The man saw the girl with the telescope." can be interpreted in different ways. New idioms may be constructed that do not exist before (Strongman, 2017). Also, another major challenge is that human language is unstructured. Unstructured data usually refers to information that cannot be represented with a predefined data model, such as a relational database, and data is not tabular (Gandomi and Haider, 2015). In terms of natural language, Email messages and word processing documents are examples of unstructured linguistic data. Humans produce unstructured natural language, and are capable of interpreting its *semantics*. The term semantics in computational linguistics generally refers to the study of how symbols in natural language carry information about the world, or specifically, the study of the meaning of a word, phrase, or any piece of text in human language (Saeed, 1997).[1] In the theories of meaning in linguistics, Akmajian et al. (2010, p. 226-234) and Riemer (2010, p. 24-38) point out to the different answers that have been proposed to the question of "what is meaning?". Some answers are as follows: the denotational theory of meaning, which states that the meaning of a linguistic expression is the thing it refers to or the (actual) object it denotes; the mentalist theory, which states that if the meaning of a linguistic expression is not an actual object, it is an idea, feeling, concept, or thought, that humans produce in their minds when the expression is used; the use theory of meaning, which says that the meaning of a linguistic expression consists in the way it is used in the language when humans communicate. Riemer (2010) argues that all theories of meaning are pertinent to the notion of meaning in linguistics; that is, "recognizing that the notion of meaning in linguistics is a way of talking about the factors which explain the language use, we can see that all theories are relevant to this task" (p. 42). In this thesis, following the idea of Harris (1954), we rely on a specific notion, that is, the meaning of a linguistic expression comes from its usage. This notion will be made more evident in the next subsection under the distributional hypothesis definition.

Unstructured linguistic data, however, is not readily understandable by tools and machines for the automatic computational processing of language. Therefore, tools and

[1] In this thesis, the term *semantics* in a general sense is used and refers to *meaning*. Therefore, these terms are used interchangeably.

machines require algorithms that take unstructured human-produced text as input and produce a structured machine-processable representation of data as output. Machine learning techniques have been introduced to automatically produce such structured representations of natural language. They are used to understand the characteristics and features of the linguistic units[1] and to automatically produce the desired representations of natural language for computational processing. This sub-discipline of machine learning in NLP is called *representation learning*. More specifically, machine learning techniques train a model for natural language representation, called a *representation model*.

Besides the challenges mentioned above, natural language has a number of properties that makes representation learning more challenging; specifically, it is *discrete symbolic* and *compositional* (Goldberg, 2017, p. 1-2).

- *Discrete symbolic*: Letters are the basic linguistic units of natural language. The combination of letters as discrete symbols creates morphemes and subsequently words that convey meanings.[2] Words denote objects, concepts, and knowledge. For instance, the words *tree* and *flower* are distinct symbols denoting distinct objects and maintaining different meanings. Then, words as discrete symbolic representations of objects form sentences, and sentences form a piece of text.
- *Compositional*: Words form phrases (sequences of words) and sentences. The meaning of a phrase can be broader than the meaning of its constituent words. To understand a phrase or sentence, we need to work beyond the level of words (Goldberg, 2017). Also, words can be combined in many ways to form new meanings. Compositionality is a natural property of human language. According to Frege's *principle of compositionality*, "The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined" (Partee, 2004, p. 153). Therefore, the meaning of a phrase can be obtained from combining the meaning of its words (Cresswell, 1976; Halvorsen and Ladusaw, 1979).

In view of the above-mentioned natural language properties that pose challenges for representation learning, this thesis focuses on two research directions:

1. Approaches and solutions to representation learning (i.e., representation models), in computational linguistics
2. Approaches for evaluating the capability of the representation models to overcome the challenges specific to semantic composition.

1.1.1 Representation Models in Computational Linguistics

Developing machine-processable word representations has been a main challenge in computational linguistics. Among the different representation models to overcome the

[1] A unit in natural language may refer to a letter, morpheme, word, phrase, clause, sentence, or text document.

[2] Morphemes are the smallest meaningful linguistic units in a natural language. However, sometimes they do not stand alone (i.e., they do not convey any meaning and appear as affixes to attach to a word, creating a new one). Words always exist on their own and have a meaning (Spencer, 1991). Therefore, in this thesis, we are interested in words.

challenges mentioned above are successful continuous and numeric high-dimensional Vector-Space Models (VSMs). VSMs can be used by various tools and techniques to automatically process natural language. These models represent characters, words, and documents by dense numeric vectors in a vector space.

Morphemes are the smallest meaningful linguistic units, however, sometimes they do not stand alone. Words are the smallest meaningful units that always stand on their own and maintain their meaning (Spencer, 1991). Therefore, we are interested in word meaning representations. Popular VSMs represent words by continuous numeric vectors (mathematical objects). One influential approach to produce word vector representations in VSMs are distributional representations, which are generally based on the distributional hypothesis first introduced by Harris (1954). The distributional hypothesis presumes that "difference of meaning correlates with difference of distribution" (Harris, 1954, p. 156). Therefore, Based on this hypothesis, "words that occur in the same contexts tend to have similar meanings" (Pantel, 2005, p. 126). Context can be defined generally as a set of sentences in which the word occurs, or the words that co-occur with the target word in a give text document. A category of approaches that are based on the idea of the distributional hypothesis are Distributional Semantic Models (DSMs)., which produce word vectors summarizing their patterns of co-occurrence in a given context (Baroni and Lenci, 2010). In these models, words with similar distributions in a context tend to have closer representations in the vector space. These approaches to semantics share the usage-based perspective on meaning; that is, the representations focus on the meaning of words that comes from their usage in a context. Therefore, the context surrounding an unknown word gives an idea about the meaning of the word. We consider this perspective on the meaning of linguistic units throughout this thesis. Semantic relationships between words in NLP can be also understood using the distributional representations. Many recent approaches utilize machine learning techniques with the distributional hypothesis to obtain continuous vector representations that reflect the meanings in natural language, such as word2vec proposed by Mikolov et al. (2013b).

To obtain the representation of larger linguistic units in VSMs, such as phrases (sequences of words) or sentences, mathematical approaches to composition are introduced that combine the mathematical objects of the word representations to obtain the representations of phrases and sentences. In this way, NLP tools and systems can process the larger units using their composed representations. These approaches have been proposed to address compositionality challenges. Compositional Distributional Semantic Models (CDSMs) obtain the compositional representation of larger units by composing the distributional representations of words using various composition methods. Successful mathematical operations for composition in VSMs have been vector addition and element-wise vector multiplication. However, these operations are not word-order sensitive. Therefore, these approaches for compositionality simplify language assumptions by ignoring word order and grammatical structure. For instance, the vector representation of the sequence "green light" will be identical to the vector representation of the sequence "light green," and therefore, the NLP system cannot differentiate between the meanings of the two sequences.

The first research direction in this thesis is considering the challenges mentioned above and the shortcomings of the proposed approaches to composition in VSMs and investigating the representation models in matrix space introduced by Rudolph and

Giesbrecht (2010), called Compositional Matrix-Space Models (CMSMs). As opposed to vector space, matrix space is used to reflect the meaning of words in matrix representations. Rudolph and Giesbrecht (2010) provide a number of advantageous properties of these models for NLP, showing that they can simulate most of the known vector-based mathematical operations for composition. Our main motivation for selecting these models is that they have shown superiority over VSMs regarding several properties. The most important property is that the composition operation in the matrix-space models can be defined as standard matrix multiplication (Rudolph and Giesbrecht, 2010); in contrast to common vector space operations, this is word-order sensitive. Therefore, for instance, the matrix representation of the sequence "green light" will not be identical to the matrix representation of the sequence "light green" in the matrix space. CMSMs have remained largely unexplored in NLP with a few notable exceptions (Yessenalina and Cardie, 2011; Irsoy and Cardie, 2015).

The main goal in this thesis is designing and developing machine learning techniques that can automatically induce continuous and numeric representations of discrete and symbolic human language in matrix space. More specifically, different machine learning approaches to train word meaning representations in CMSMs and capture the compositionality of larger units using matrix multiplication of words are investigated. The input to these machine learning techniques is human language, and the output is a representation model in the matrix space. Thus, learning techniques transform (map) distinct symbols in natural language into mathematical objects (matrices) that can be operated on for computational processing. The main goal of introducing representation models is to enable NLP systems to understand the natural language to solve multiple natural language-related tasks. Therefore, the performance of the CMSMs, as representation models learned by machine learning techniques, is investigated in solving several NLP tasks. For this purpose, for each NLP task, a learning technique for the CMSMs is proposed.

Another contribution of this research is studying how CMSMs can be represented by a type of mathematical model of computation called Weighted Finite Automata (WFA). The main idea in this study is to show that, if WFA are obtained using learning algorithms, based on the correspondence between CMSMs and WFA, the learned automata can be mapped to CMSMs for use in NLP.

1.1.2 Evaluating Semantic Composition in Representation Models

As mentioned above, influential approaches to represent word meanings are through word vectors that capture the patterns of co-occurrence in a given context (i.e., the distributional representations; Baroni and Lenci, 2010). In addition, different composition methods have been introduced to achieve the representation of larger linguistic units (e.g., phrases) from word representations and capture semantic compositionality. Each semantic composition method has advantages and limitations. Approaches for evaluating the capability of these methods to capture the compositional meaning representations have been one of the challenges in NLP.

A common approach to evaluating word representation models is through the models' ability to rank pairs of words by their closeness in meaning. Closeness is a measure of

how close two terms[1] are in relation to their semantics; that is, two terms are considered to be semantically close if there is a sharing of some meaning (Mohammad, 2008). For instance, the two words *teacher* and *tutor* are closer in meaning than the two words *teacher* and *fish*. Sharing of meaning is defined based on the lexical–semantic relations (i.e., the semantic relations between the lexical items; Cruse, 1986). He points out that the number of lexical–semantic relations is innumerable, however, certain relations, such as synonymy, hypernymy (hyponymy), meronymy (holonymy), and antonymy, received more attention as they are systematic. Morris and Hirst (2004) call the systematic relations that form predetermined structures as *classical* lexical–semantic relations, and the others as *non-classical* lexical–semantic relations. Closeness in meaning, defined based on the lexical–semantic relations, can be of two kinds: *semantic similarity* and *semantic relatedness*. Semantically similar terms tend to share several properties. For example, *apples* and *bananas* are both edible, grow on trees, have seeds, and so on. In terms of lexical–semantic relations, two terms are considered semantically similar if there is a hypernymy, (co-)hyponymy, synonymy, or antonymy relationship between them as these relations share common properties (Budanitsky and Hirst, 2001; Mohammad, 2008; Agirre et al., 2009; Baroni and Lenci, 2010).

Semantically related terms may not have many shared properties, but have at least one classical or non-classical relation between them that allows them to be considered semantically close. For example, consider the meanings of *coffee* and *cup*. They are not similar as they have practically no common properties: *coffee* is a drink, while *cup* is an object with a specific shape. However, both are related as they are associated in the world by commonly co-occurring in a shared event of *drinking a cup of coffee* (Mohammad and Hirst, 2005; Harispe et al., 2015). Therefore, two terms are considered semantically related if there is any lexical–semantic relation between them, classical or non-classical. In general, similar terms are also related but there are terms that are not similar but strongly related (associated) and co-occur in shared events. For instance, *surgeon* and *scalpel* are related through the non-classical relation, and *doctor* and *surgeon* are also semantically related through the hypernymy relation; however, they are also similar as *surgeon* is a hyponym of *doctor*. Therefore, semantic relatedness is the broader class subsuming semantic similarity. Lexical–semantic relations, semantic similarity, and semantic relatedness are explained in more detail in Chapter 2.4.

Previous studies have shown that the ability to assess semantic relatedness between terms is central to the understanding of natural language (Hutchison, 2003; Huth et al., 2016). Therefore, the quantification of semantic relatedness between pairs of terms in natural language is needed for evaluation purposes in NLP. For this, gold standard evaluation datasets consisting of pairs of terms with semantic relatedness scores are created. A gold standard dataset is a dataset that is accepted as a reliable and standard reference and the best available resource for evaluating methods and models. It is usually annotated and corrected by human experts. To obtain the semantic relatedness scores in datasets, human judgments are needed to rank the semantic relatedness between pairs of terms.

The approach to evaluating a representation model is as follows: given term pairs in a gold standard semantic relatedness dataset, the closeness of the two term representations

[1] Terms can be words, phrases, or sentences.

obtained from a representation model are computed using some metrics, such as the cosine between the two vector or matrix representations. Then, the computed values are compared with the gold standard scores in the dataset.

Existing datasets of semantic relatedness in English, such as that by Finkelstein et al. (2002), only focus on single words (unigrams). However, the concept of semantic relatedness applies to larger linguistic units produced by composition, such as phrases or sentences. Therefore, we argue that semantic relatedness can be used to evaluate methods of semantic composition. Bigrams (two-word sequences) are important in semantic composition as they are the smallest units formed by composing words. Even though there is a large body of work on how to represent the meanings of sentences (Le and Mikolov, 2014; Kiros et al., 2015; Lin et al., 2017), there is relatively little work on how best to compose the meanings of two words to represent the meaning of a bigram. One reason for this is a lack of gold standard evaluation resources. Therefore, phrase-based gold standard semantic relatedness datasets for the development and evaluation of representation models are useful.

Existing datasets also suffer from shortcomings due to the techniques employed for annotating data with human judgments. Except in the case of a few small but influential datasets, such as those by Miller and Charles (1991) and Rubenstein and Goodenough (1965), annotations were obtained using the *rating scales* technique. Here, annotators are asked to choose from categorical or discrete numerical values to rate the data. For instance, when annotating a pair of words for semantic relatedness, the annotator can be asked to choose among integer values from 1 to 5, with 1 representing that the two words are least semantically related or semantically unrelated, and 5 representing that the words are strongly semantically related. The rating scales technique suffers from significant known limitations, including inconsistencies in annotations by different annotators, inconsistencies in annotations by the same annotator at different times, bias toward a portion of the scale, and problems associated with a fixed granularity (an annotator may want to choose 1.5 instead of 1 or 2; Presser and Schuman, 1996).

The second research direction in this thesis is identifying the limitations of the existing gold standard datasets and developing a semantic relatedness dataset that addresses the issues of the existing datasets. The introduced dataset can then be used to evaluate semantic composition methods.

1.2 Contributions and Outline of the Thesis

In the following, we outline the contributions of this thesis and clarify the publications that the results in this thesis are based on. Chapter 2 first sets up the foundation of the thesis. The next three chapters outline our main contributions to the proposed two research goals. Chapters 3 and 5 focus on the first research direction, while Chapter 4 concentrates on the second research direction. The conclusion in Chapter 6 distills the findings and discusses shortcomings and potential future directions of the research. We give a more detailed chapter summaries in the following.

Chapter 2 Summary

Chapter 2 sets up the foundation of the thesis. It covers the relevant background

topics, including representation learning and representation models (distributional and distributed representations) in computational linguistics, the basics of neural networks, compositionality in computational linguistics, semantic composition methods, CMSMs and their applications in NLP, and evaluation approaches for semantic composition methods in compositional representation models.

CMSMs were introduced by Rudolph and Giesbrecht (2010) as word-level representation models in matrix space that can capture compositional representations of larger linguistic units, such as phrases, using standard matrix multiplication. This chapter reviews the advantageous properties of these models for NLP on the theoretical side, showing that they can simulate most of the known vector-based semantic composition operations.

While this chapter focuses on the background and foundations, Section 2.3.3 is devoted to our contribution about the correspondence between CMSMs and WFA which is based on our publication, Asaadi and Rudolph (2016). Since this contribution is closely related to the introduction and fundamentals of CMSMs, it is presented in this chapter.

Chapter 3 Summary

Chapter 3 provides an experimental investigation of CMSMs as an alternative to VSMs in NLP applications. For this purpose, the following two NLP tasks are considered: sentiment analysis and compositionality detection. Compositionality plays an important role in such tasks, and therefore, they are suitable for evaluating the capability of CMSMs in capturing word semantic representations and compositionality [1].

Considering the task of sentiment analysis, a supervised learning technique based on linear regression is proposed to train word matrices in CMSMs from training datasets, which consist of phrases (sequence of words) and their real-valued sentiment scores. During the training of the model, compositional matrix representations of phrases are realized via standard matrix multiplication. Word matrices are trained to contain semantic and sentiment-related information. The novelty of our approach is a two-step learning procedure, where the result of the first step is used as initialization for the second step. After training, the learned model is used to predict the sentiment scores of previously unseen phrases using the trained word matrices and matrix multiplication as the composition operation to obtain phrase matrices. Experiments are conducted with two different training datasets and compared with previous learning approaches to CMSMs in sentiment analysis (Yessenalina and Cardie, 2011; Irsoy and Cardie, 2015), as well as to VSMs. The results show the outperformance of the proposed learning approach for CMSMs over existing approaches for learning CMSMs. It also achieves competitive performance with learning approaches for VSMs.

In the task of compositionality detection, a supervised learning technique based on linear regression is also proposed for training CMSMs. The learning approach trains a model that captures compositional phrase matrices from their word matrices. After the model is trained, an evaluation method is provided to compare the representations of previously unseen phrases produced by CMSMs with the gold standard representations of those phrases obtained from standard and reliable resources. This comparison method presents how well CMSMs detect the compositionality of phrases. The performance of CMSMs in the compositionality detection task is compared with various compositional

[1] This task is also called *compositionality prediction* task.

VSMs, from unsupervised to supervised models. The results demonstrate outperformance of CMSMs over some VSMs and competitive performance with some other VSMs in this task. The investigations in this chapter show that there are suitable learning methods for training CMSMs in NLP downstream tasks while requiring fewer training parameters.

Section 3.2 of this chapter which is about learning CMSMs for sentiment analysis is based on our publication, Asaadi and Rudolph (2017). Section 3.3 is an extension to our publication.

Chapter 4 Summary

Chapter 4 proposes a gold standard Bigram Semantic Relatedness Dataset (BiRD), which is then used for examining semantic composition methods. BiRD consists of 3,345 English term pairs (i.e., bigram–bigram and bigram–unigrams) with an associated semantic relatedness score between the two terms obtained from human annotations. Each bigram occurs in about eight distinct pairs in BiRD. This is yet another aspect that makes BiRD unique, as existing datasets were not designed to include terms in multiple pairs.

The first step to create the dataset is collecting the pairs of terms for annotation. A lexical database and text resources are used to collect the term pairs. A second step is instructing the annotators (humans) to annotate our collected data using a comparative annotation technique, called Best–Worst Scaling (BWS; Louviere, 1991; Cohen, 2003; Louviere et al., 2015). BWS addresses the limitations of the rating scales technique by applying comparative annotations (Louviere et al., 2015; Kiritchenko and Mohammad, 2017). In the employed technique, annotators are not asked to assign a semantic relatedness score to the terms in pairs. Instead, they are asked to answer some provided questions by comparing a given set of term pairs. That is, annotators are given n term pairs at a time. They are asked which pair is the *best* (highest in terms of the property of interest, in our case, the closest in meaning or most related) and which is the *worst* (lowest in terms of the property of interest, in our case, least close or least related). After the annotation task is completed by the annotators, the semantic relatedness score for each pair is computed based on the best–worst responses of the annotators using a simple counting procedure (Orme, 2009; Flynn and Marley, 2014). The final step is investigating the quality of the obtained scores. A commonly used measure of quality in dimensional annotation tasks is the reproducibility of the final scores—the extent to which repeated independent manual annotations produce similar results. To assess this reproducibility, consistency of annotations is determined using a common approach called Split-Half Reliability (SHR; Cronbach, 1951), which provides a measurement of the reliability of the created dataset. SHR shows that the semantic relatedness annotations are highly reliable; that is, if the annotations were repeated, then similar scores and rankings would be obtained. Therefore, BiRD is introduced as a gold standard semantic relatedness dataset.

We use BiRD to evaluate semantic composition methods on their ability to score the semantic relatedness between term pairs. The underlying assumption is that the more accurately a method of semantic composition can determine the representation of a bigram, the more accurately it can determine the relatedness of that bigram with other terms. More specifically, we apply various semantic composition methods to word

representation models to obtain a compositional representation of bigrams. Then, we study which composition methods capture the semantic representation of terms in pairs more accurately. For this purpose, we first compute the semantic relatedness scores between the representations of terms in pairs using some measures, such as the cosine between the two vector representations. Then, we compare the obtained scores to the gold standard scores in BiRD.

The dataset is analyzed to obtain insights into the distributions of semantic relatedness scores for pairs associated through various lexical–semantic relations. Observations provide useful suggestions for the creation of new datasets. This chapter is based on our publication, Asaadi et al. (2019).

Chapter 5 Summary

Chapter 5 introduces CMSMs as generic word representation models in matrix space (also called word matrix embeddings). Machine learning techniques are proposed to obtain word embeddings. Word matrices are trained using distributional information of words based on the distributional hypothesis and Point-wise Mutual Information (PMI) between words in a given text corpus. As opposed to Chapter 3, where word matrices are trained on a specific task, such as sentiment analysis, in this chapter, the proposed models are not trained to capture task-specific information. Therefore, they are called task-agnostic models, meaning that the semantic information embedded in the word matrices are not specific to any NLP task. The embeddings provide continuous word representations of natural language and reflect the semantic relationships between words.

Two learning techniques to obtain word matrix embeddings are proposed in this chapter. The first is a supervised learning technique based on linear regression, which is called PMI-based CMSM. It utilizes PMI values for training word matrices, which present global information about the association between words co-occurring in a given text corpus. The second technique, called the Compositional Order-Sensitive Matrix Model (COSMo), is a self-supervised method for training a two-layer linear neural network, inspired by the skip-gram method (Mikolov et al., 2013a). Self-supervised learning refers to a learning task in which there are no predetermined labeled training data for training the matrices and the input data to the neural network determine the labels during the training procedure. At the core of the method is non-commutative matrix multiplication during training, which, as opposed to the skip-gram method, is word-order sensitive when training the word matrices. It is also trained based on the distributional hypothesis, which provides local information about the association between words. Based on this hypothesis, words with similar distributions tend to have a close meaning and close representations in the space.

To investigate the performance of the introduced word matrix embeddings on semantic representation and compositionality, we evaluate them using the two following evaluation tasks: (1) semantic relatedness (closeness) and (2) Semantic Textual Similarity (STS). These tasks refer to determining the relatedness and similarity degrees between pairs of phrases and sentences, respectively. The performances of the proposed matrix embeddings in capturing the compositional representation of terms are compared against two existing matrix representation models (Mai et al., 2019; Chung et al., 2018), as well as skip-gram word vector embedding (Mikolov et al., 2013b). Compositional representation in

matrix space is obtained using the multiplication of word matrices. In vector space, it is obtained using the two following operations: (1) vector addition and (2) element-wise multiplication; these approaches are reported separately. The results show that the introduced matrix embeddings outperform the model proposed by Chung et al. (2018) and perform competitively with the model proposed by Mai et al. (2019). Moreover, the skip-gram with vector addition as the composition operation outperforms all other models, while the skip-gram with vector multiplication fails to outperform any model.

Publications:

- Shima Asaadi and Sebastian Rudolph. 'On the Correspondence between Compositional Matrix-Space Models of Language and Weighted Automata'. In: *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata (StatFSM 2016)*. Ed. by Bryan Jurish, Andreas Maletti, Kay-Michael Würzner, and Uwe Springmann. Association for Computational Linguistics, pp. 70–74.
- Shima Asaadi and Sebastian Rudolph. 'Gradual Learning of Matrix-Space Models of Language for Sentiment Analysis'. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP (RepL4NLP 2017)*. Ed. by Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston, and Scott Yih. Association for Computational Linguistics, pp. 178–185.
- Shima Asaadi, Saif M. Mohammad, and Svetlana Kiritchenko. 'Big BiRD: A Large, Fine-Grained, Bigram Relatedness Dataset for Examining Semantic Composition'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (NAACL-HLT 2019)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, pp. 505–516.

Chapter 2

Background and Foundations

This chapter introduces preliminaries and technical concepts of this thesis and provides references to the relevant history and prior work. Section 2.1 introduces representation models in computational linguistics. Section 2.2 introduces the principle of compositionality in natural language, and Section 2.3 describes compositional matrix-space models of language and their properties. Finally, Section 2.4 introduces the concept of semantic relatedness as an approach to examine semantic composition in NLP.

Before continuing, we introduce some notations and aspects of linear algebra that will be used throughout this thesis.

Vectors: a vector of dimension d, with d as a natural number, is a list of d real numbers $r_1, \ldots, r_d \in \mathbb{R}$, written as $\mathbf{v} = (r_1, r_2, \ldots, r_d)$. Each element r_i shows a dimension of the vector. We use lowercase bold font letters to denote vectors and $\mathbf{v}(i)$ to refer to the ith element of vector $\mathbf{v}$. $\mathbb{R}^d$ indicates the set of all d-dimensional vectors with real-valued elements. Element-wise arithmetic operations can be done on vectors such as vector addition defined as $(r_1, \ldots, r_n) + (r'_1, \ldots, r'_n) = (r_1 + r'_1, \ldots, r_n + r'_n)$, and vector multiplication defined as $(r_1, \ldots, r_n) \odot (r'_1, \ldots, r'_n) = (r_1 \cdot r'_1, \ldots r_n \cdot r'_n)$. Vector addition and multiplication are commutative (i.e., $\mathbf{v} + \mathbf{v}' = \mathbf{v}' + \mathbf{v}$ and $\mathbf{v} \odot \mathbf{v}' = \mathbf{v}' \odot \mathbf{v}$), which means changing the order of operands does not change the result.

Matrices: an $n \times m$ matrix over the reals is an array of real numbers with n rows and m columns, given two natural numbers n and m. We use uppercase letters to denote matrices and, given a matrix M, $M(i, j)$ refers to the element in the ith row and the jth column:

$$M = \begin{pmatrix} M(1,1) & \cdots & M(1,j) & \cdots & M(1,m) \\ \vdots & & & & \vdots \\ M(i,1) & & M(i,j) & & M(i,m) \\ \vdots & & & & \vdots \\ M(n,1) & \cdots & M(n,j) & \cdots & M(n,m) \end{pmatrix}.$$

$M(i, :)$ and $M(:, j)$ refer to all elements in row i and column j in the matrix, respectively. The set of all $n \times m$ matrices with real number elements is denoted by $\mathbb{R}^{n \times m}$. The transpose of a matrix can be obtained by switching the row and column indices, in other words, given a matrix $M \in \mathbb{R}^{n \times m}$, its transpose $M^\top$ is a $m \times n$ matrix where $M^\top(i, j) = M(j, i)$.

Matrices can operate as *linear mapping* in vector space. An $n \times m$ matrix applied to an m-dimensional vector results in an n-dimensional vector $\mathbf{u}$ which is a linear mapping from the input vector $\mathbf{v}$ to the output vector $\mathbf{u}$:

$$\mathbf{u}(i) = \sum_{j=1}^{m} \mathbf{v}(j) \cdot M(i, j)$$

for $1 \leq i \leq n$. Linear mapping is also realized using standard matrix multiplication in matrix space. Therefore, MM' denotes the linear mapping defined by applying first M and then M'. Matrix multiplication of the $n \times \ell$ matrix M and the $\ell \times m$ matrix M' is an $n \times m$ matrix $N = MM'$ defined by

$$N(i, j) = \sum_{k=1}^{\ell} M(i, k) \cdot M'(k, j).$$

Matrix product is associative (i.e., $(MM')M'' = M(M'M'')$ always holds), and parentheses can be discarded, but it is a non-commutative operation (i.e., MM' does not equal $M'M$ in general). In an $n \times m$ matrix, row and column indices can also range from 0 to $n-1$ and $m-1$, respectively.

Third-order Tensors: a third-order tensor of dimension $m \times n \times d$ over real values is a m array of $n \times d$ matrices. Third-order tensors are denoted by uppercase bold font letters, and $\mathbf{T}(i, j, k)$ refers to row j and column k of matrix i in $\mathbf{T}$. $\mathbb{R}^{m \times n \times d}$ indicates the set of all tensors with real number elements.

2.1 Representation in Computational Linguistics

When dealing with NLP tasks, tools and systems require a structured machine–processable encoding of the input linguistic data (e.g., words of a natural language) to understand the natural language for automatic computational processing. Encoding the natural language in NLP is called *representation*, and encoding the meaning of linguistic units, such as words and sentences, in formal structures is called *meaning representation* (Jurafsky and Martin, 2014). Work in meaning representation explores how best to represent the meanings of linguistic units (e.g., words, phrases, or sentences) to the NLP tasks.

Main approaches to representing natural language differ in a few fundamental ways. A popular technique is to convert text (a sequence of words) to numeric representations. One of the most natural approaches is *categorical encoding* (also called *label encoding*), where words of a given text are represented in an increasing order. In this approach, we first extract the vocabulary from the text. Then, we build a lookup dictionary by creating a mapping between words and IDs (i.e., each unique word in the vocabulary is assigned an ID), as for instance shown in Fig. 2.1. One drawback of this approach is that by treating words as integers, the system may incorrectly assume the existence of natural ordering. For instance, the dictionary contains entries such as 1: "book" and 2: "library". The word with greater ID value may be considered more important by the NLP system and therefore given a higher weight, which is a wrong assumption.

Word	**ID**
book	1
library	2
magazine	3
journal	4
in	5
exist	6
and	7

Figure 2.1: Example of categorical encoding for a sample sentence "book, magazine and journal exist in library".

As Goldberg (2017) asks,"How do we encode such categorical data in a way which is amenable for us by a statistical classifier?" (p. 89). We need a way to represent natural language that eliminates the drawback of categorical encoding.

One–hot representation (Harris and Harris, 2007) is an alternative approach to word representation. In this approach, given a vocabulary Σ with size $|\Sigma| = V$, an ID or index is assigned to each word in Σ. Then, a function $f : \Sigma \rightarrow \{0, 1\}^V$ creates a vector of length V for each word, where the element corresponding to the word's ID or index in the vocabulary is 1 and the other elements are zero as for instance shown in Fig. 2.2. Therefore, a unique element is associated with each word of the vocabulary. A drawback of this approach is the immense and sparse representation. For instance, when a text contains V words, a matrix of size $V \times V$ is created where only V elements are one and the rest of elements are zero, which consumes a lot of storage.

ID \ Word	book	library	magazine	journal	in	exist	and
1	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	0	1	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	0	1

Figure 2.2: Example of one-hot representation of the words in a sample sentence "book, magazine and journal exist in library".

A limitation of the mentioned approaches is that information about the syntactic (arrangement of the words in a sentence) and semantic (meaning of the words in a sentence) relationships between the words of a text cannot be captured and word representations are created independently. As an example of a semantic relationship, consider the word *book* which can have the meaning of an object to read like "borrow a book from the library" or of a verb to reserve something like "book a flight". These approaches create one representation for *book* without looking at the semantic relationship between *book* and other words in the given sentence. Thus, the representation of *book* does not provide information regarding which meaning is used and it is open to multiple interpretations.

To eliminate the drawbacks of the mentioned approaches, alternative methods have been introduced such as distributional representations and distributed representations, which are described in the following.

2.1.1 Distributional Representations

One influential approach to semantic representation is the *distributional representation* (also called distributional semantics), which is generally based on the *Distributional Hypothesis* first introduced by Harris (1954). The Distributional Hypothesis presumes that "difference of meaning correlates with difference of distribution" (Harris, 1954, p. 156). John Firth, a linguist, popularized this hypothesis by stating "you shall know a word by the company it keeps" (Firth, 1962, p. 11).[1] More particularly and according to Pantel (2005), the distributional hypothesis states that "words that occur in the same contexts tend to have similar meanings" (p. 126). The context of a word can be defined

[1] Also in (Firth, 1957).

generally as a set of sentences referred to as a *text document* where the word occurs, or as the neighboring words of the target word in a given text document. Therefore, the context surrounding an unknown word gives an idea about the meaning of the word. A category of the representation models that produce word representations based on the distributional hypothesis are Distributional Semantic Models (DSMs). In DSMs, the meanings of words come from their usage in a context. DSMs are one of the dominant approaches in VSMs. VSMs introduced by Salton et al. (1975) is an algebraic model for the representation of any object as a point in a high-dimensional vector space. In DSMs, vectors in a high-dimensional vector space are used to represent words. Word meaning representation is then derived quantitatively based on the word's distributional information and statistics on word usage in a given text (i.e., from its context) (Goldberg, 2017, p.118). Therefore, word representations in DSMs provide information about the semantic relationship between them. DSMs are useful in meaning-related tasks in NLP, such as paraphrase detection[1] (Turney, 2013). To produce the distributional representations of words, Salton et al. (1975) and Deerwester et al. (1990) introduce the co-occurrence matrix. A term–document co-occurrence matrix computes how often a term (e.g., a word) occurs in a document (e.g., a sentence or a web page) (Salton et al., 1975). Rows in the matrix show the words and columns represent the documents. A word–context matrix shows how often a word occurs in a context if the context is a phrase, sentence, etc., or how often the word co–occurs with the context if the context is a word (Deerwester et al., 1990). In the latter case, a *context window size* is defined that determines the number of surrounding words in the range of the given word to be considered as the context words. For instance, a window size of 3 refers to three words preceding and three words succeeding the given word as the context. A word–context matrix consists of words in rows and contexts in columns. Each element in the matrix quantifies the association between a word and a context. There are different ways of measuring the association between words and contexts, such as raw frequency count of a word in a context (or the count of co-occurrence of a word with a context). Finally, each row in the matrix represents the target word vector and each column represents the context vector (Deerwester et al., 1990). Therefore, each element in the word vector corresponds to a specific context the word occurs in, and it is separately interpretable. This way, words that happen in similar contexts have a close vector representation in the vector space, and thus, a close semantic relationship.

Fig. 2.3 illustrates an example of a word–context matrix given the following three document excerpts:[2]

D1. banana and pineapple are tropical fruits.

D2. Boston has available flights to major US cities.

D3. flights to major cities were canceled due to bad weather conditions.

In this example, contexts are defined as the surrounding words of the target words with a window size of $k = 5$ (i.e., five words preceding and five words succeeding the target words in the given document excerpts). An association measure is the number of times

[1]Paraphrase detection is the task of determining if two phrases or sentences have the same meaning.
[2]We use document excerpts instead of documents for simplicity purposes.

Word \ Context	banana	···	tropical	fruit	Boston	cities
banana	0	···	1	1	0	0
pineapple	1	···	1	1	0	0
···	···	···	···	···	···	···
flights	0	···	0	0	1	2

Figure 2.3: Example of a word–context matrix with a context window size of five.

the word co-occurs with the context, for instance, pineapple and banana co-occurring once and flights and cities co-occurring twice.

A text, such as a sentence or a document, can also be represented by vectors. Among the first attempts to represent a text is the Bag-Of-Words (BOWs) representation model (Harris, 1954) in which the text is represented with an unordered list of its words and their count in the text. Therefore, in BOWs any information about the order of words is discarded. This simplifies language assumptions and leads to problems when trying to understand a text. For instance, consider each document as a sentence and the words in sentences as the terms:

D1. Banana and pineapple are tropical fruits.
D2. Banana and rambutan are tropical fruits.
D3. Banana and pineapple are yellow fruits.

First, a vocabulary from all sentences is created. In this example, there are eight unique words in all sentences: Banana, and, pineapple, rambutan, are, tropical, fruits, and yellow. Using the arbitrary ordering of words listed in our vocabulary, each document can now be represented with a vector of size $V = 8$, where each element shows to the frequency of the corresponding word in the document:

Vocabulary= {Banana, and, pineapple, rambutan, are, tropical, fruits, yellow}
D1 = [1,1,1,0,1,1,1,0]
D2 = [1,1,0,1,1,1,1,0]
D3 = [1,1,1,0,1,0,1,1]

This model has drawbacks at representing texts, such as sparsity and discarding the word order. As the vocabulary size increases, the dimension of vector representations also increase. Different arrangements of the same set of words offer different meanings, which is discarded in BOW.

2.1.2 Distributed Representations

Another approach to obtain distributional representations in VSMs is the *distributed representation* (Hinton et al., 1986). In a distributed representation, each word in the vocabulary set is associated with a low dimensional vector $f : \Sigma \rightarrow \mathbb{R}^d$ where the "meaning" of the word is distributed over many dimensions and captured by different dimensions of the vector. This means that a combination of several dimensions may capture a given aspect of meaning and that each dimension may contribute to capturing

several aspects of meaning (Goldberg, 2017, p. 122). Thus, each dimension is not interpretable separately as opposed to the above-mentioned representations, such as the word–context matrix in which each dimension corresponds to a specific context the word occurs in.

Distributed representation models are based on the distributional hypothesis, as they try to capture the similarity between the words occurring in similar contexts (Goldberg, 2017). Levy and Goldberg (2014) show that distributed representations are deeply connected to distributional models and current distributed representation models use distributional signals to learn their representations.

Distributed representations are obtained by machine learning techniques, referred to as *embeddings*. More specifically, utilizing neural networks, low-dimensional vector representations for linguistic units are produced. These representations can be subdivided into *task-specific representations* and *task-agnostic representations*. Task-specific representations are trained on a specific NLP task (e.g., sentiment analysis) to obtain specific information. The definitions are made clear throughout the following section.

In machine learning techniques, a model learns from training samples with respect to performance measure $\mathcal{P}$ and some task $\mathcal{T}$, if its performance as measured by $\mathcal{P}$ improves with more training samples. We can reduce the problem of improving performance $\mathcal{P}$ at task $\mathcal{T}$ to the problem of learning a particular target (objective) function f^*. Therefore, the type of knowledge that the model will learn is formulated as an objective function. Adjustable model parameters θ are defined to achieve the best approximation f of the target function f^* in the given task $\mathcal{T}$ (Mitchell, 1997, p. 2).

In supervised learning tasks in machine learning, labeled training datasets are used to train the model (i.e., a set of n input–output pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$). The outputs (or target values) are predetermined and usually obtained by human annotations. The objective is to train the model parameters θ, i.e., trainable weights, using the labeled dataset that result in the best approximation of the target values $f(x_i; \theta) \approx y_i$ for $1 \leq i \leq n$.

A widely popular method to obtain distributed word representations is word2vec introduced by Mikolov et al. (2013a,b). In the following, we first introduce neural networks and then word2vec.

Neural Networks

Neural Networks (NNs), originally inspired by the biological neural networks, have received much attention in artificial intelligence, machine learning, and computational linguistics. Neural networks provide a class of machine learning methods for different problems "to approximate real-valued, discrete-valued and vector-valued target functions" (Mitchell, 1997, p. 81). A neural network connects several layers of neurons of which the first is the input, the last is the output, and in the middle there is a number of user-definable hidden layers. Each neuron is a computational unit that has inputs and outputs. Neurons of different layers are connected with connecting weights. These weights are adjustable parameters that are tuned to closely approximate the desired target function, given input–output data to the network.

There are several types of neural network architectures from Recurrent Neural Networks (RNNs) to Convolutional Neural Networks (CNNs), the simplest of which being

feedforward NNs (Rosenblatt, 1957; Rumelhart et al., 1986).

Feedforward Neural Networks:

The reason these networks are called feedforward is that the information from the input layer is forward-propagated through the hidden layers to the output layer, and there is no recurrence during the feed-forward process. That is, there are no loops between neurons in different layers. The main goal in a feedforward NN is to approximate the target function f^* which maps an input x_i to a target value y_i, given a training set of n input–output pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. A feedforward NN defines a mapping function $y = f(x; \theta)$ and learns the value of the adjustable parameters θ, i.e., the weight matrices W connecting the layers and the bias vectors $\boldsymbol{b}$ for each layer, that result in the best function approximation (Goodfellow et al., 2016). Given a large set of input–output data as the training set, the goal is to generalize the function f to best approximate the output of previously unseen input data.

Each neuron in network layers takes input from neurons of previous layers and computes its activation value. Therefore, each layer's output is the subsequent layer's input. Fig. 2.4 illustrates an example of a two-layer fully connected feedforward NN with two input neurons, two neurons on the single hidden layer and one output neuron. W^1 is a 2×2 weight matrix and W^2 is a 2×1 weight matrix connecting input to hidden and hidden to output layer, respectively.

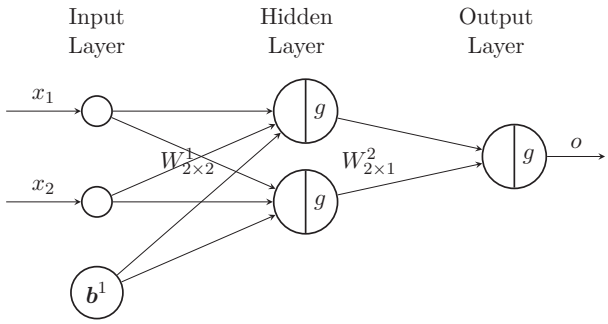


Figure 2.4: Example of a fully connected feedforward neural network with input $\boldsymbol{x} = (x_1, x_2)$, output o, bias $\boldsymbol{b}^1$, and one hidden layer. An activation function g is applied to its input argument at each layer.

Given a set of training input–output data, the objective is to approximate the target function f^* producing the output value by training the weight matrices and bias vectors (parameters θ) and optimizing the performance of the neural network in the approximation.

The output value o of a given input $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$ is predicted by a network of L layers using the following computation:

$$o = g^{L-1}(\cdots g^2(g^1(\boldsymbol{x}W^1 + \boldsymbol{b}^1)W^2 + \boldsymbol{b}^2)\cdots W^{L-1} + \boldsymbol{b}^{L-1})W^L,$$

where $W^l \in \mathbb{R}^{d_{l-1} \times d_l}$ shows the weight matrix connecting layer $l-1$ with dimension d_{l-1} to layer l with dimension d_l, and each g^l is a differentiable activation function, such as the *sigmoid* function that is applied element-wise to its argument. $\boldsymbol{b}^l \in \mathbb{R}^{d_l}$ is a bias

vector at each layer l. In our example, given two inputs x_1 and x_2 shown as $\boldsymbol{x} = (x_1, x_2)$, the output is computed as follows:

$$\boldsymbol{h} = g(\boldsymbol{x}W^1 + \boldsymbol{b}^1),$$
$$o = \boldsymbol{h}W^2.$$

To best approximate the target values, an optimizer and an objective function E (also called loss function (Goldberg, 2017)) are defined to train the parameters of the network. The objective function E at any point of training is a function of the difference between approximation o made by the network and the target value y it is trying to reach. There are different objective functions. A common one is the Mean Squared Error (MSE) defined as follows:

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - o_i)^2,$$

where o_i is the output value that is compared with the target value y_i, and n is the size of training set.

To train the weight matrices, different algorithms, such as backpropagation, are applied. Backpropagation is one of the most popular methods in NNs that minimizes the objective function by optimizing the weights using an optimizer, such as gradient descent (Cauchy, 1847; Robbins and Monro, 1951; Berstsekas, 1999). Gradient descent is an iterative optimization algorithm, which is applied to machine learning problems. In gradient descent, the goal is to find the local minimum/maximum (i.e., local optimum) of an objective function by taking steps proportional to the negative/positive gradient of the function at the current point toward the local optimum. In a NN, it computes the gradient of the objective function E with respect to each element in the weight matrix W^l of layer l for a set of n input–output data:

$$\frac{\partial E}{\partial W^l(j,k)} = \sum_{i=1}^{n}\frac{\partial E}{\partial o_i} \times \frac{\partial o_i}{\partial W^l(j,k)}.$$

In our example, the gradient of the objective function E with respect to each weight element is computed using partial derivatives of the functions computed in the forward propagation:

$$\frac{\partial E}{\partial W^2(j,k)} = \sum_{i=1}^{n}\frac{\partial E}{\partial o_i} \times \frac{\partial o_i}{\partial W^2(j,k)},$$
$$\frac{\partial E}{\partial W^1(j,k)} = \sum_{i=1}^{n}\frac{\partial E}{\partial o_i} \times \frac{\partial o_i}{\partial W^1(j,k)},$$

where j and k range from 1 to 2, and $\frac{\partial o_i}{\partial W^2(j,k)}$ and $\frac{\partial o_i}{\partial W^1(j,k)}$ are computed using the chain rule as follows:

$$\frac{\partial o_i}{\partial W^2(j,k)} = \frac{\partial o_i}{\partial g} \times \frac{\partial g}{\partial W^2(j,k)},$$
$$\frac{\partial o_i}{\partial W^1(j,k)} = \frac{\partial o_i}{\partial g} \times \frac{\partial g}{\partial h} \times \frac{\partial h}{\partial W^1(j,k)}.$$

Then, each weight element is updated using the negative of the gradient of the objective

function as follows:

$$W^l(j,k) = W^l(j,k) - \eta \frac{\partial E}{\partial W^l(j,k)},$$

where η is a constant called *learning rate* that controls how much the step size is adapted towards the local minimum of the objective function. This way, the error propagates backward in the network and all weights of the layers are updated. The forward and backward propagations are repeated many times (i.e., in many training steps), until there is no significant decrease in the error computation (i.e., the amount of the decrease is less than a threshold value ϵ). Thus, the weights are optimized and the network reached its best approximation of input–output pairs from training set. The network can then be used to predict the output of previously unseen input data.

Recurrent Neural Networks (RNNs):

RNNs (Rumelhart et al., 1986; Elman, 1990) allow to operate on inputs and outputs as sequences. The input data x to RNNs is an arbitrary length sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ with $\boldsymbol{x}_t \in \mathbb{R}^{d_{in}}$ (fixed length vector) where at each time step t the input $\boldsymbol{x}_t$ is fed to the network. RNNs have a *memory* that remembers information from all previous time steps in a hidden state (also called hidden layer). At each time step, the previous hidden layer $\boldsymbol{h}_{t-1}$ is fed to the network at the current time step and is combined with input $\boldsymbol{x}_t$ to compute the current hidden layer as follows:

$$\boldsymbol{h}_t = g(\boldsymbol{x}_t U + \boldsymbol{h}_{t-1} W + \boldsymbol{b}_t),$$

where U and W are shared weight matrices for input-to-hidden and hidden-to-hidden layers, respectively, and $\boldsymbol{b}_t$ is the bias of the network. g is a nonlinear activation function, such as the *tanh* function. This way, historical information about the input sequence is kept in the hidden layers. The output $\boldsymbol{y}_t$ at each time step is a vector $\boldsymbol{y}_t \in \mathbb{R}^{d_{out}}$, which is computed as follows:

$$\boldsymbol{y}_t = g(\boldsymbol{h}_t V + \boldsymbol{c}),$$

where V is a shared weight matrix for the hidden-to-output layer. $\boldsymbol{c}$ is a bias and g is either a linear or a nonlinear activation function. In general, the output at the final time step T (i.e., $\boldsymbol{y}_T$), is used to approximate the prediction of the network for a given sequence of inputs.

To train the network, training data is fed to the network and the objective function is computed. Then, the error is back-propagated through the network to update all weight matrices. We are not concerned with the detail of RNNs in this thesis and the interested reader is referred to the Deep Learning book by Goodfellow et al. (2016) for details.

2.1.3 Word2vec

Word2vec is a method for learning distributed word vector representations in NLP using a feedforward NN with one hidden layer. The goal of training the network is to map the words of a large text corpus (a set of sentences) to vector representations. The word vectors are obtained based on the distributional hypothesis. Therefore, words with similar contexts are located in close proximity in vector space. There are two methods of word2vec to learn word vector representations: Continuous Bag Of Words (CBOW)

and skip-gram.

The training dataset used in word2vec is a large text corpus of sentences that is not previously labeled, and therefore, the learning task in word2vec is not a supervised learning task. To train the weight matrices and approximate the target function, however, a target label is created for each input during the training procedure. This way, training is done in a supervised manner. This type of learning is called *self-supervised* (Sa, 1994). Note that in some literature, such as the work by Pennington et al. (2014), the learning of word representations is called as an unsupervised method.

In the self-supervised learning task for representation learning, we are not interested in the final performance of the model in the given task. Instead, we extract the learned intermediate parameters as representations that carry the semantic representations. Similar to other networks, the weight matrices in word2vec are trained to best approximate the target value in a self-supervised learning task. However, the network is not used for the task that it is trained on. Instead, the learned weight matrices are extracted after training and introduced as the word vector representations that carry the semantic information of words.

Moreover, word vector representations obtained from word2vec capture the semantic relationship between words based on the distributional hypothesis. These representations are not trained on a specific NLP task (e.g., sentiment analysis) to obtain specific information. Therefore, a task-agnostic representation model is obtained from word2vec, which is in contrast to task-specific representation models, for example, sentiment analysis.

In the following, we explain both methods of word2vec introduced in (Mikolov et al., 2013a,b) and how target labels are created for training the network.

Skip-gram:

As mentioned before, the goal of training the network is to obtain vector representations for all words in a given vocabulary Σ of size V. The vocabulary is created from a large text corpus of sentences T. In skip-gram the task is to predict the context words of a word (called center word) based on a given corpus of sentences.

The input to the network is a center word $w_t \in \Sigma$ with $t \in \{1, \ldots, V\}$. Since we cannot feed the words as strings to the network, we need to map all words to numeric representations. For this purpose, we assign an index to each word of the vocabulary and a one-hot encoding of size V is created as a numeric representation of the words based on their index. For instance, a word with index t in vocabulary is assigned to a vector of size V, where the tth element of the vector is one and all other elements of the vector are zero. This way, the input words to the network are one-hot encodings.

The learning task is to train the network to predict the context words of any input center word w_t. Therefore, for each word $w_t \in \Sigma$, $t \in \{1, \ldots, V\}$, we extract a set of context words (with window size k) $C_t = \{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$ co-occurred with w_t in the corpus T from which we created the vocabulary. Recall window size of k means k words preceding and k words succeeding w in the sentences of T. This way, target labels for input data are created for a self-supervised learning task.

Fig. 2.5 illustrates the sketch of the network for context window size of $k = 2$. w_t and each of its context words are represented with one-hot encodings of size V. Now, if we

look at each context of the input word w_t separately, the network architecture for each output context w_c can be illustrated as Fig. 2.6. Considering the input center word w_t and the output context word w_c, at each iteration of training, the network produces a probability of being the context of w_t for every word in Σ. Thus, the output vector of size V in Fig. 2.6 presents the probability of each word at its corresponding index being the context of w_t. The objective here is to produce a probability distribution close to the one-hot encoding of the context word shown in the figure (i.e., to maximize the probability of w_c and minimize the probability of all other words). Therefore, the predicted probability distribution is compared with the one-hot encoding of the context word w_c, and the error is computed as a function of the difference between the target one-hot encoding and the predicted probability distribution. A similar computation is done for each output context word shown in Fig. 2.5 at the same time and the total error back-propagates to the network to update all weight matrices accordingly.

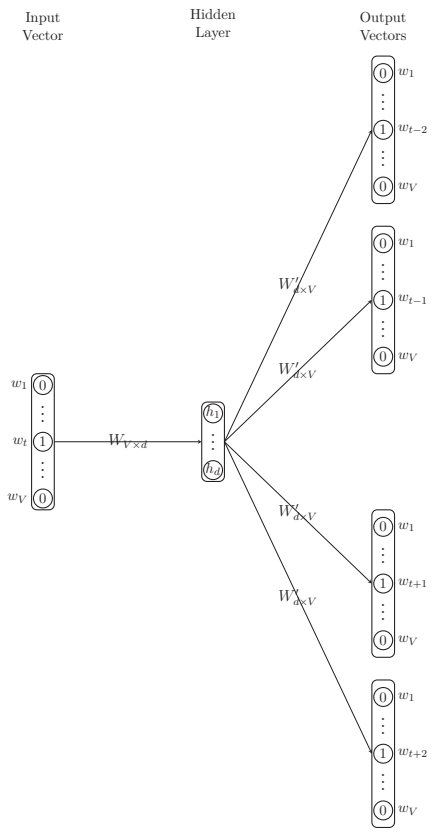


Figure 2.5: Skip-gram architecture for an input word w_t and its context words $\{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$ as output. Context window size is 2 and V is the size of vocabulary. input and output words are represented with one-hot encodings.

As shown in Fig. 2.6, the network consists of two layers with trainable weight matrices $W_{V\times d}$ and $W'_{d\times V}$. V is the size of the vocabulary Σ created from T and d is the size of hidden layer, which as to be decided upon and equals the final number of embedding

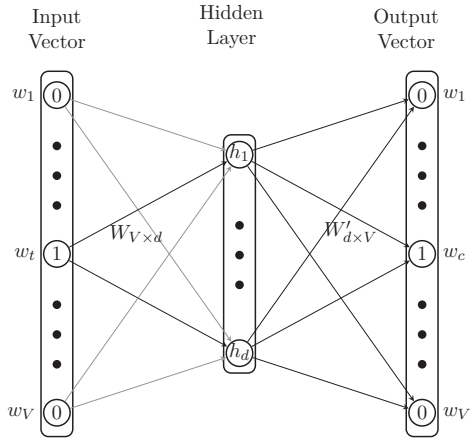


Figure 2.6: skip-gram architecture for an input word w_t and an output context word w_c.

dimensions. The size of input and output layers is equal to the vocabulary size V. Note that with a one-hot encoding input vector, the matrix row connected to the vector element with value 1 is considered in forward propagation since the other elements of the input vector are zero.

For training the network, each word $w_t \in \Sigma$ is paired with its contexts $C_t = \{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$. The objective is to train the model parameters θ so as to maximize the log probability of context words co-occurring with w_t, or to minimize the negative of log probability, called objective function (also called loss function (Goldberg, 2017)), defined in the following:

$$\min \mathcal{O} = -\log P(w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k} | w_t; \theta).$$

Assuming that the word occurrences are completely independent, the objective function for each word is defined as

$$\min \mathcal{O} = -\log \prod_{j=0, j\neq k}^{2k} P(w_{t-k+j} | w_t; \theta).$$

Given a sequence of training words $w_1, w_2, \ldots, w_V$, the final objective function is to minimize the average log probability over all words as in Equation 2.1.

$$\begin{aligned} min\mathcal{O} &= -\frac{1}{V}\sum_{t=1}^{V} \log \prod_{j=0, j\neq k}^{2k} P(w_{t-k+j} | w_t; \theta) \\ &= -\frac{1}{V}\sum_{t=1}^{V} \sum_{j=0, j\neq k}^{2k} \log P(w_{t-k+j} | w_t; \theta). \end{aligned} \tag{2.1}$$

The probability function is defined using the softmax function:

$$P(w_c | w_t) = \frac{e^{\mathbf{v}_{w_t} \cdot \mathbf{v}'_{w_c}}}{\Sigma_{w=1}^{V} e^{\mathbf{v}_{w_t} \cdot \mathbf{v}'_{w}}}, \tag{2.2}$$

where $\mathbf{v}_{w_t}$ and $\mathbf{v}'_{w_c}$ are the model parameters θ with $\mathbf{v}_{w_t} = W(t,:)$ (i.e., row t of matrix W), and $\mathbf{v}'_{w_c} = W'(:,c)$ (i.e., column c of matrix W'). The vectors $\mathbf{v}_w$ and $\mathbf{v}'_w$ are the center and context word vector representations, respectively, extracted from weight matrices W and W'. Therefore, after training the network with the given objective function, we extract $W_{V\times d}$ as the final vector representation of words with each row as the vector representation of the corresponding word. d is the dimensionality of the vectors. Also, $W'_{d\times V}$ as the final vector representation of context words with each column as the vector of the corresponding word.

Mikolov et al. (2013b) discuss that the above formulation in Equations 2.1 and 2.2 is impractical as the error of computing $\log P(w_c|w_t)$ at each iteration of training is proportional to V, which is a large number. Therefore, an efficient approximation of the full softmax is proposed as an alternative to the above computations, called *negative sampling*.

Negative sampling is an efficient approximation to Equation 2.1 (Mikolov et al., 2013b). In this method, at each iteration of training, random words are drawn from the vocabulary to be considered as the false contexts (called negative samples) of the current center word. A negative sample $w_{c'}$ is selected using a unigram probability distribution with a smoothing parameter α, computed as follows:

$$q(w_{c'}) = \frac{f(w_{c'})^\alpha}{\Sigma_{i=1}^{V}(f(w_i)^\alpha)}, \tag{2.3}$$

where $f(w)$ is the frequency of the word w in the corpus T and V is the size of vocabulary. Frequencies are raised to the power of α ($0 < \alpha \leq 1$), which is the smoothing parameter. This parameter increases the selection probability of less frequent words and decreases the probability of more frequent words. Mikolov et al. (2013b) reported that $\alpha = 3/4$ results in the best performance.

If we create a training data D consisting of pairs of center–context words, (w_t, w_c), we denote by $P(D = 1|w_t, w_c)$ the probability that the pair is in D and by $P(D = 0|w_t, w_c)$ the probability that the pair is not in D. If we assume D' as the set of drawn negative samples paired with center words, $(w_t, w_{c'})$, the objective is now to maximize the probability of correct contexts:

$$\arg\max_{\theta} \prod_{(w_t,w_c)\in D} P(D = 1|w_t, w_c; \theta) \prod_{(w_t,w_{c'})\in D'} P(D = 0|w_t, w_{c'}; \theta) \tag{2.4}$$

with P as the *sigmoid* function:

$$P(D = 1|w_t, w_c; \theta) = \frac{1}{1 + e^{-\mathbf{v}_{w_t}.\mathbf{v}'_{w_c}}},$$

where $\mathbf{v}_{w_t}$ and $\mathbf{v}'_{w_c}$ are extracted from W and W' as described above and are considered as the model parameters θ to be trained. We can maximize the log of the objective function in Equation 2.4. The training procedure, backpropagation and weight updates are the same as explained for the feedforward NNs.

Finally, the learned weight matrix W of the network is extracted after training and introduced as the word vector representations.

Mikolov et al. (2013b) show that by treating each phrase (sequence of words) as an individual token, they can similarly train the phrase embeddings as they train the word vectors.

Continuous Bag Of Words (CBOW):

Similar to skip-gram, the goal of training the network is to obtain vector representations for all words in a given vocabulary Σ of size V. However, as opposed to skip-gram, the task in CBOW is to predict the center word given context words as input to the network. Mikolov et al. (2013a) show the network architecture of this method as in Fig. 2.7, where w_t and each of its context word are represented with one-hot encodings of size V.

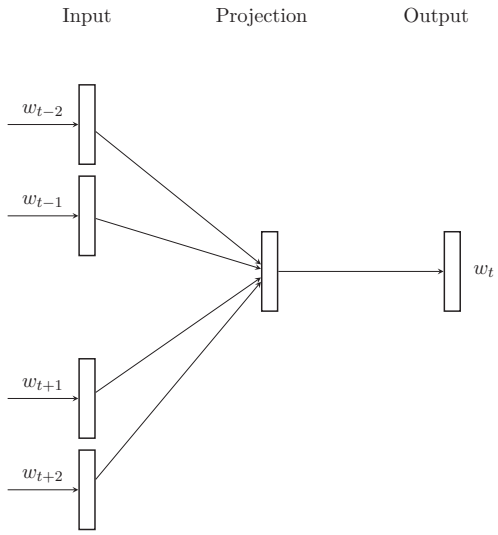


Figure 2.7: CBOW architecture to predict the center word w_t based on a set of context words. Illustration is taken from (Mikolov et al., 2013a). Copyright (2013) by CoRR.

Now, assume for each word $w \in \Sigma$ we extracted a set of context words (with window size k). The input to the network is one-hot encodings of k context words. The objective of the network is to predict the center word w_t based on a set of given contexts words $\{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$.

For training the network, each word w_t in vocabulary is paired with its contexts $C_t = \{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$. The objective is to train the model parameters θ so as to maximize the log probability of the center word co-occurring with the context words, or to minimize the negative of log probability, as defined in the following:

$$\min \mathcal{O} = -\log P(w_t | w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}; \theta).$$

After training the network, the weight matrix W' is extracted as the word vector representations.

The hierarchy of the studied numeric representation models in this section is illustrated in Fig. 2.8. The focus of this thesis is on distributed representation models. We aim to

improve on this topic using an alternative model to VSMs, and propose both task-specific and task-agnostic representation models. We are not concerned with the symbolic approaches and do not study these models in this thesis.

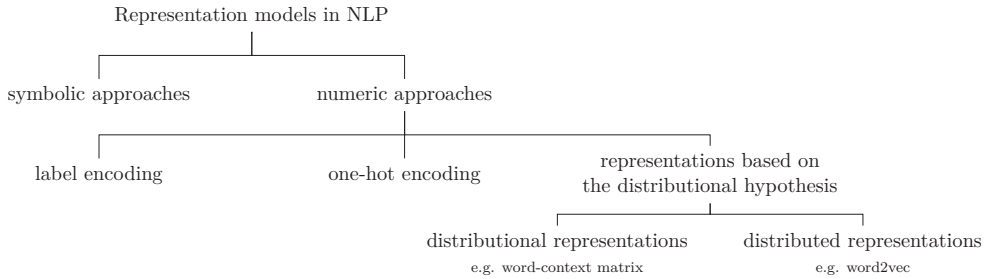


Figure 2.8: Hierarchy of representation models in NLP. Symbolic approaches are outside the scope of this thesis.

2.2 Compositionality in Distributional Semantic Models

Compositionality is a natural property of human language. According to Frege's *principle of compositionality*, "The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined" (Partee, 2004, p. 153). Therefore, the meaning of a complex expression in a natural language is determined by its structure and the meanings of its constituents. (Cresswell, 1976; Halvorsen and Ladusaw, 1979).

Based on this principle, we understand the meaning of *basketball player* from the meaning of its constituent words: *basketball* and *player*, and thus, it is a compositional phrase. Opposite to compositional expressions are the cases when we use phrases in other non-literal ways, and therefore, the meaning of a complex expression is not dependent on the meaning of its parts. For instance, *throw in the towel*, which means *to give up*, is not directly derived from composing the meaning of its constituents, and therefore, it is not compositional. Moreover, some phrases can be understood as either compositional or non-compositional depending on their context.

The principle of compositionality in computational linguistics can be formalized as follows (Rudolph and Giesbrecht, 2010): given a vocabulary Σ of a natural language, some semantic space $\mathbb{S}$ (its elements will be called "meanings"), and a mapping function $[\![\,\cdot\,]\!] : \Sigma \to \mathbb{S}$, a semantic composition operation $\bowtie$: $\mathbb{S}^* \to \mathbb{S}$, which maps sequences of meanings to meanings is defined such that the meaning of a sequence of words $s = \sigma_1\sigma_2\ldots\sigma_k$ can be obtained by applying $\bowtie$ to the sequence $[\![\sigma_1]\!][\![\sigma_2]\!]\ldots[\![\sigma_k]\!]$. This situation, displayed in Fig. 2.9, qualifies the semantic mapping $[\![\,\cdot\,]\!]$ as a homomorphism between the two algebraic structures $(\Sigma^*, \cdot)$ and $(\mathbb{S}, \bowtie)$.

The study of semantic compositionality is central in computational linguistics, as several downstream NLP tasks, such as statistical machine translation (Weller et al., 2014), word sense disambiguation (Akkaya et al., 2012), and sentiment analysis (Yessenalina and Cardie, 2011), require recognizing compositionality on phrase or sentence levels.

DSMs are concerned with the meaning representation of single words. They show little consideration on how the meaning of words combine in a phrase or sentence and

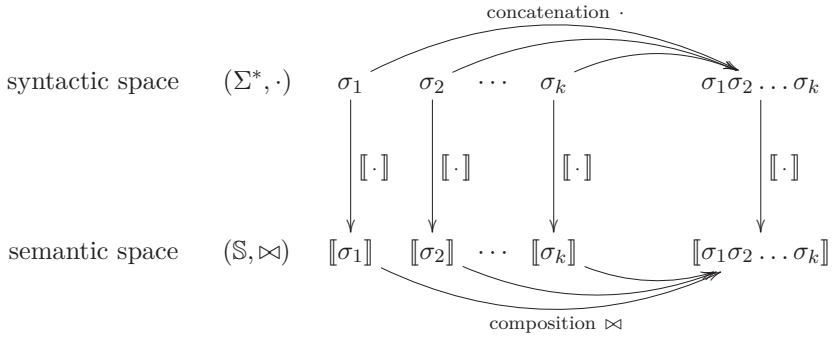


Figure 2.9: Semantic mapping as homomorphism, illustration is taken from (Rudolph and Giesbrecht, 2010) . Copyright (2010) by ACL.

how meaning might be influenced by way of position of a word in a sentence. Therefore, different semantic composition methods have been proposed to obtain meaning above the word-level. DSMs that are equipped with semantic composition methods are called Compositional Distributional Semantic Models (CDSMs). Generally, any semantic composition method that is applied to representation models produce compositional representation models. In these models, different composition methods are defined to compose the word representations in vector space to model representations of larger linguistic units. Salton and McGill (1986) introduced vector addition in VSMs as a composition method, which is the most common method. Given two words w_i and w_j and their associated d-dimensional semantic vector representations $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$, respectively, vector addition is defined as follows:

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}) = \mathbf{u} + \mathbf{v},$$

where $\mathbf{p} \in \mathbb{R}^d$ is the resulting compositional representation of the phrase $w_i w_j$ and f is called the composition function.

Despite its simplicity, the additive method is not a suitable method of composition because vector addition is commutative. Therefore, it is not sensitive to word order in the sentence, which is a natural property of human language. For instance, if $w_i =$*green* and $w_j =$ *light*, the representation of *green light* and *light green* would be the same discarding the word order.

As an early attempt to provide more compelling compositional functions in linguistic structures, a context-sensitive composition method was proposed by Kintsch (2001), who used a more sophisticated composition function to model predicate–argument structures. Kintsch (2001) argued that the neighboring words "strengthen features of the predicate that are appropriate for the argument of the predication" (p. 178). For instance, the predicate *run* depends on the noun as its argument and has a different meaning in, e.g., *the horse runs* and *the ship runs before the wind*. Thus, different features are used for composition based on the neighboring words and, not all features of a predicate vector are combined with the features of the argument, but only those that are appropriate for the argument.

An alternative seminal work on composition methods was proposed by Widdows (2008). Widdows introduced a number of more advanced vector operations for semantic compositionality, such as tensor product with convolution operation to capture compositionality in vector space. Given two vectors $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$, the tensor product of two vectors is a matrix $Q \in \mathbb{R}^{d \times d}$ with $Q_{ij} = \mathbf{u}(i)\mathbf{v}(j)$. Since the number of dimensions increases by tensor product, the convolution operation was introduced to compress the tensor product operation to $\mathbb{R}^d$ space. Both operations can be summarized as $\mathbf{p}(i) = \sum_j \mathbf{u}(j)\ \mathbf{v}(i-j)$ for $1 \leq i, j \leq d$, where $\mathbf{p} \in \mathbb{R}^d$. Widdows showed the ability of the introduced compositional models to reflect the phrasal meaning on a simplified analogy task[1], where they outperform the additive model.

Mitchell and Lapata (2010) introduced dilation and element-wise vector multiplication as the composition methods and compared with weighted vector addition $\mathbf{p} = \alpha\mathbf{u} + \beta\mathbf{v}$ with $\alpha + \beta = 1$. Element-wise vector multiplication, defined as $\mathbf{p} = g(\mathbf{u}, \mathbf{v}) = \mathbf{u} \odot \mathbf{v}$, does not consider the word order. The dilation method decomposes $\mathbf{v}$ into a parallel and an orthogonal component to $\mathbf{u}$ and then stretches the parallel component to adjust $\mathbf{v}$ along $\mathbf{u}$:

$$\mathbf{p}(i) = \mathbf{v}(i) \sum_j \mathbf{u}(j)\mathbf{u}(j) + (\lambda - 1)\mathbf{u}(i) \sum_j \mathbf{u}(j)\mathbf{v}(j),$$

where λ is the dilation factor and $\mathbf{p}$ is the composed vector. Therefore, $\mathbf{u}$ affects relevant elements of vector $\mathbf{v}$ in the composition. An evaluation was done on a compositional semantic similarity task, which is about determining the similarity degree between pairs of phrases. They developed a similarity dataset of two-word sequences (bigrams) for this purpose. For each bigram in a pair (ab, cd), the word vectors are composed using the introduced composition methods to obtain the bigram vector, and then the two bigram vectors are compared with each other using the cosine of the angle between them to determine the similarity degree between the bigrams. They conclude that element-wise vector multiplication outperforms vector addition.

Baroni and Zamparelli (2010) proposed a method for the compositional representation of Adjective–Noun (AN) compounds. Adjectives are attributive (i.e., they are treated as functions over nouns and modify the meaning of nouns when combined with them). Therefore, adjectives are considered as linear functions mapping the d-dimensional noun vectors onto the same vector space. This function can be expressed as a $d \times d$-dimensional matrix M, which is multiplied with the d-dimensional noun vector $\mathbf{v}$, resulting in a compositional representation of the adjective–noun compound in the same vector space:

$$\mathbf{p} = M\mathbf{v}.$$

Therefore, adjectives are represented with matrices. In this method, a unique matrix M is assigned to each adjective and trained using linear regression. To create a training dataset, AN and noun vectors are obtained by creating a co-occurrence matrix (or a term–context matrix in which terms are either nouns or ANs) from a large text corpus based on the distributional semantics (Salton et al., 1975; Deerwester et al., 1990). To train the adjective matrices using the training dataset, an adjective matrix M is

[1] Analogy task refers to determining the proportional analogy between two pairs of words, for example, *Berlin* is to *Germany* as *Rome* is to *Italy*.

multiplied with a noun vector $\mathbf{v}$, which is obtained from the co-occurrence matrix. Then, the i-th row of the matrix is the coefficients of the linear regression that predict the value of the i-th dimension of the corresponding AN vector $\mathbf{p}$.

Some approaches to the distributional representation of words in VSMs have also been extended to CDSMs. Turney (2012) proposed a dual-space model for semantic compositionality. He created two vector-space models from the word–context co-occurrence matrix, one from the noun as the context of the words (called domain space) and the other from the verb as the context of the words (called function space). Therefore, the dual-space model consists of a domain space for determining the similarity in topic or subject, and a function space for computing the similarity in role or relationship. He evaluated his dual-space on the task of similarity of compositions for pairs of bigram–unigram (ab, c). He computed the domain similarity of words a and c (i.e., computing the cosine of the word vectors $\mathbf{v}_a$ and $\mathbf{v}_c$ extracted from the domain space):

$$sim_d(a, c) = \cos(\mathbf{v}_a, \mathbf{v}_c) = \frac{\mathbf{v}_a . \mathbf{v}_c}{\| \mathbf{v}_a \| \| \mathbf{v}_c \|},$$

where $\| \cdot \|$ is the Euclidean norm of a vector $\mathbf{v}$ of size d as follows:

$$\| \mathbf{v} \| = \sqrt{\mathbf{v}(1)^2 + \cdots + \mathbf{v}(d)^2}.$$

Similarly, domain similarity of words b and c, $sim_d(b, c)$, and function similarity of b and c, $sim_f(b, c)$ (i.e., computing the cosine of the word vectors extracted from the function space) are computed. Then geometric mean of the three similarities are computed:

$$\begin{aligned} geo(x_1, x_2, x_3) &= (\prod_{i=1}^{3} x_i)^{\frac{1}{3}} \\ sim_1(ab, c) &= geo(sim_d(a, c), sim_d(b, c), sim_f(b, c)) \\ sim(ab, c) &= \begin{cases} sim_1(ab, c) & \text{if } a \neq c \text{ and } b \neq c, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

In general, the goal is to have a high combined similarities using the geometric mean value when the component similarities are high. This way, he introduced the dual-space model as a CDSM.

2.3 Compositional Matrix-Space Models

Several models have been proposed to model compositionality and address the commutativity issue of vector operations. Among these models, Rudolph and Giesbrecht (2010) proposed the Compositional Matrix-Space Models (CMSMs) as an alternative to VSMs. In these models, the semantic space consists of real-valued quadratic matrices $\mathbb{S} = \mathbb{R}^{m \times m}$, and therefore, the meaning of words is represented by matrices $[\![\cdot]\!] : \Sigma \rightarrow \mathbb{R}^{m \times m}$. Standard matrix multiplication serves as the composition operation $\bowtie : (\mathbb{R}^{m \times m})^\star \rightarrow \mathbb{R}^{m \times m}$, which, as opposed to many operations in VSMs, is order-sensitive. Rudolph and Giesbrecht (2010) studied different aspects of CMSMs and provided arguments showing that

CMSMs are intuitive and natural. In the following, we review the main aspects and properties of CMSMs proposed by Rudolph and Giesbrecht (2010), which represent an important premise for this thesis.

Common composition methods in natural language representation models are both associative and commutative. Therefore, they ignore the order of words within the sentence in a given natural language. For instance, the semantic representation of "Sarah likes Chocolate" and "Chocolate likes Sarah" would be the same. Therefore, the non-commutativity property of the composition method is crucial in compositional representation models to differentiate between meaning representations of a set of words with different orders. Matrix multiplication is non-commutative, and therefore, CMSMs are introduced as more adequate for modeling semantic compositionality in natural language (Rudolph and Giesbrecht, 2010).

Neurological and cognitive plausibility of CMSMs have also been studied by Rudolph and Giesbrecht (2010) from an abstract and simplified perspective. Assume that the mental state of a person at a particular time is encoded as a numeric vector $\mathbf{v}_1$. Then receiving an external signal (e.g., perceiving a new word ω_1) causes the change of the current mental state $\mathbf{v}_1$ to a new state $\mathbf{v}_2$. Therefore, the external signal can be formulated as a transition function applied to $\mathbf{v}_1$ to map the current mental state of a human to a new state $\mathbf{v}_2$. The transition can be defined as a matrix M_{ω_1} that changes the mental states linearly: $\mathbf{v}_2 = \mathbf{v}_1 M_{\omega_1}$. An arbitrary sequence of external signals (e.g., $\omega_1\omega_2\ldots\omega_k$) transforms the current mental state to a new state using a sequence of associated transition functions $\mathbf{v}_{k+1} = (\ldots((\mathbf{v}_1 M_{\omega_1})M_{\omega_2})\ldots M_{\omega_k})$. Therefore, $M_{\omega_1}M_{\omega_2}\ldots M_{\omega_k}$ represents the state transitions triggered by a sequence of external signals $\omega_1\omega_2\ldots\omega_k$. As an example, Fig. 2.10 illustrates the transitions triggered by $\omega_1\omega_2$. This way, the mental state transformations can be simulated in matrix space as the function space where matrix multiplication executes the transformations after receiving the external signals.

Pribram et al. (1960) and Baddeley and Hitch (1974) proposed the concept of *working memory*, a limited cognitive system that supports the processes of human thought and stores information. This working memory of human can be represented by the mental state vector, which is transformed by sequential external inputs to human memory. Therefore, the working memory can also be explained by CMSMs, where operations in working memory, such as storing, deleting, copying, can be realized by matrices (Rudolph and Giesbrecht, 2010), and matrix multiplication preserves the ordered changes of successive input information in human memory. For instance, the $m \times m$-dimensional matrix $M_{\text{copy(k,l)}}$ defined as

$$M_{\text{copy(k,l)}}(i,j) = \begin{cases} 1 \text{ if } i = j \neq l \text{ or } i = k,\ j = l, \\ 0 \text{ otherwise.} \end{cases}$$

can be applied to a m-dimensional vector $\mathbf{v}$ to copy the value of kth element of the vector to its lth position.

Another advantage of CMSMs exhibited by Rudolph and Giesbrecht (2010) is that several matrix models can be combined into one model in an appropriate way. Given a sequence of words $s = \sigma_1\ldots\sigma_k$ in a natural language with associated matrix representations $[\![\sigma_1]\!], \ldots, [\![\sigma_k]\!] \in \mathbb{R}^{m\times m}$ according to one model and matrix representations

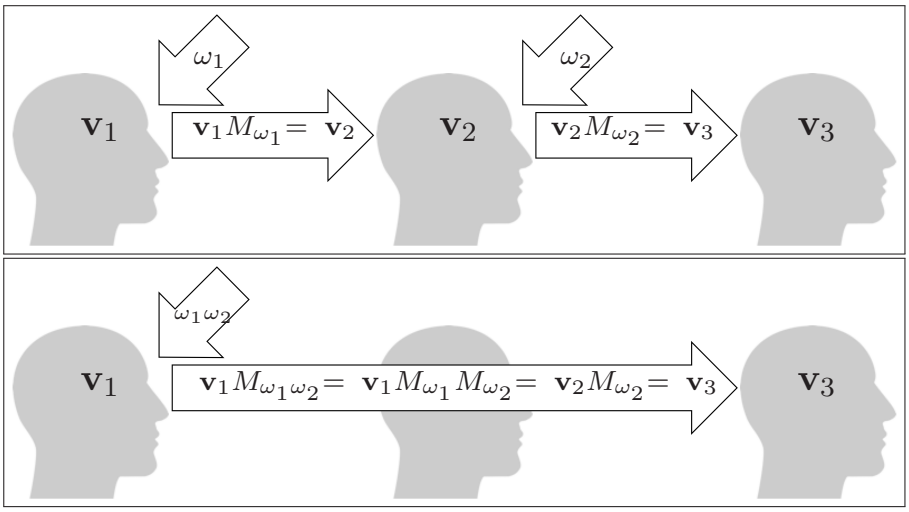


Figure 2.10: Cognitive state transformations of a human from state $\mathbf{v}_1$ to $\mathbf{v}_3$ using matrices M_{ω_1} and M_{ω_2} as the state transition functions after receiving external signals ω_1 and ω_2.

$(\!|\sigma_1|\!), \ldots, (\!|\sigma_k|\!) \in \mathbb{R}^{n\times n}$ according to another model in matrix space, the two models can be combined into one model by assigning to σ_i a $(m+n)\times(m+n)$-dimensional matrix:

$$\{\!|\sigma_i|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_i]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_i|\!) & \\ 0 & & 0 & & & \end{array}\right).$$

The matrix multiplication of newly created matrices results in the following matrix:

$$\{\!|\sigma_1|\!\}\ldots\{\!|\sigma_k|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_1]\!]\ldots[\![\sigma_k]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_1|\!)\ldots(\!|\sigma_k|\!) & \\ 0 & & 0 & & & \end{array}\right),$$

which captures the semantic compositions in the two models simultaneously.

Many words can be interpreted in multiple ways depending on their context. Ambiguity in natural language happens when a model cannot determine which meaning of a word is used in a given context. Therefore, associative operations are not able to disambiguate word meanings. For instance, "The man saw the girl with the telescope." can be interpreted in different ways. The first is that the man saw a girl holding a telescope,

while another possible interpretation refers to the man utilizing a telescope to spot the girl. Matrix multiplication is associative, which is generally unable to disambiguate the different meanings of a sentence. However, Rudolph and Giesbrecht (2010) argue that natural language is inherently stream-like and sequential. Thus, associativity alone seems more justifiable, and the different meanings of sentences can be resolved by contextual cues.

2.3.1 Compositional Matrix-Space Models and Compositional VSMs

It has been theoretically shown that CMSMs are capable to simulate well known composition operations in VSMs and therefore, they subsume compositional VSMs. To show this, let $\psi_{\bowtie} : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ and $\chi_{\bowtie} : \mathbb{R}^{m\times m} \to \mathbb{R}^n$ be the mapping functions between the vector and matrix representations, and a vector $\mathbf{v}_\sigma \in \mathbb{R}^n$ assigned to each word σ of vocabulary Σ in a natural language.

Consider vector addition as the composition operation. Given a sequence of words $w = \sigma_1 \dots \sigma_k$, $\mathbf{v}_w = \sum_{i=1}^{k} \mathbf{v}_{\sigma_i}$. Now, define the function $\psi_+ : R^n \to R^{(n+1)\times(n+1)}$ to map the vector $\mathbf{v}_\sigma$ of word σ to the corresponding matrix representation in the following way:

$$M_\sigma = \psi_+(\mathbf{v}_\sigma) = \left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline & \mathbf{v}_\sigma & & 1 \end{array}\right).$$

Multiplying the resulting matrices yields:

$$\psi_+(\mathbf{v}_w) = \psi_+(\mathbf{v}_{\sigma_1}) \dots \psi_+(\mathbf{v}_{\sigma_k})$$

$$M_w = \psi_+(\mathbf{v}_w) = \left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline & \mathbf{v}_w & & 1 \end{array}\right).$$

Now define $\chi_+ : \mathbb{R}^{(n+1)\times(n+1)} \to \mathbb{R}^n$ to extract the lowest row omitting the last element, which results in:

$$\chi_+(M_w) = \mathbf{v}_w = \Sigma_{i=1}^{k} \mathbf{v}_{\sigma_i}.$$

Element-wise vector multiplication in VSMs, is defined as $\mathbf{v}_w = \mathbf{v}_{\sigma_1} \odot \cdots \odot \mathbf{v}_{\sigma_k}$ where $\mathbf{v}_w(j) = \mathbf{v}_{\sigma_1}(j) \cdot \mathbf{v}_{\sigma_2}(j) \dots \mathbf{v}_{\sigma_k}(j)$ for $1 \le j \le n$, given a sequence of words $w = \sigma_1 \dots \sigma_k$. This operation can be also simulated by CMSMs. This time, let $\psi_\odot : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ encode the vector $\mathbf{v}_\sigma$ to diagonal matrix with the vector elements as its diagonal elements:

$$M_\sigma = \psi_\odot(\mathbf{v}_\sigma) = \begin{pmatrix} \mathbf{v}_\sigma(1) & 0 & \cdots & 0 \\ 0 & \mathbf{v}_\sigma(2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{v}_\sigma(n) \end{pmatrix}.$$

Then, multiplying the matrices corresponding to words σ_i results in:

$$M_w = \psi_{\odot}(\mathbf{v}_w) = \begin{pmatrix} \mathbf{v}_w(1) & 0 & \cdots & 0 \\ 0 & \mathbf{v}_w(2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{v}_w(n) \end{pmatrix}.$$

Now, define $\chi_{\odot} : \mathbb{R}^{n \times n} \to \mathbb{R}^n$ to extract the main diagonal elements of the output matrix, which is:

$$\chi_{\odot}(M_w) = \mathbf{v}_w = \mathbf{v}_{\sigma_1} \odot \cdots \odot \mathbf{v}_{\sigma_k}.$$

Similarly, circular convolution operation[1], which has been introduced by Plate (1995) as a composition operation in VSMs, can be simulated via CMSMs. Circular convolution is interpreted as a compressed outer product of vectors. Given two words σ_1 and σ_2 with associated n-dimensional vectors $\mathbf{u}$ and $\mathbf{v}$, respectively, circular convolution is defined as a tensor product of the two vectors which results in a matrix Q of dimension $n \times n$ where $Q(i,j) = \mathbf{u}(i)\mathbf{v}(j)$, and then a convolution operation is applied to map the matrix to the n-dimensional vector $\mathbf{v}_w$ corresponding to $w = \sigma_1\sigma_2$ where:[2]

$$\mathbf{v}_w(i) = \sum_{j=0}^{n-1} \mathbf{u}(j)\mathbf{v}(i-j) \quad \text{for } \ 0 \le i \le n-1.$$

Fig. 2.11 illustrates the computation of circular convolution operation as a compressed outer product of two vectors. To simulate this operation by CMSMs, let $\psi_{\circledast}(\mathbf{v}_\sigma)$ be the

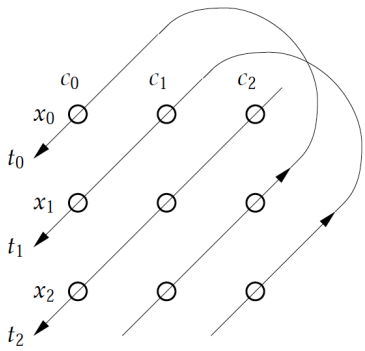


Figure 2.11: Circular convolution operation on two 3-dimensional vectors $\mathbf{c}$ and $\mathbf{x}$. $\mathbf{t}(i) = \sum_{j=0}^{2} \mathbf{c}(j)\mathbf{x}(i-j) \quad \text{for } \ 0 \le i \le 2$. To avoid confusion, indices start from 0. Illustration is taken from (Plate, 1995). Copyright (1995) by IEEE Press.

$n \times n$ matrix M_σ associated to word σ where the first row of the matrix is the vector $\mathbf{v}_\sigma$:

[1]This composition operation is distinct from the convolution operation in neural networks. Convolution operation in NNs applies a filter to an input data to summarize the features in the input.

[2]For simplicity and to avoid confusion, vector indices in the equation start from 0 instead of 1.

$$M_\sigma = \psi_{\circledast}(\mathbf{v}_\sigma) = \begin{pmatrix} \mathbf{v}(1) & \mathbf{v}(2) & \mathbf{v}(3) \\ \mathbf{v}(3) & \mathbf{v}(1) & \mathbf{v}(2) \\ \mathbf{v}(2) & \mathbf{v}(3) & \mathbf{v}(1) \end{pmatrix}.$$

Then multiplying matrices of two words σ_1 and σ_2, results in:

$$M_w = \psi_{\circledast}(\mathbf{v}_w) = \begin{pmatrix} \mathbf{v}_w(1) & \mathbf{v}_w(2) & \mathbf{v}_w(3) \\ \mathbf{v}_w(3) & \mathbf{v}_w(1) & \mathbf{v}_w(2) \\ \mathbf{v}_w(2) & \mathbf{v}_w(3) & \mathbf{v}_w(1) \end{pmatrix}.$$

If we define $\chi_{\circledast}(M_w)$ to extract the first row of the resulting matrix, it outputs $\mathbf{v}_w$.

2.3.2 Compositional Matrix-Space Models and Regular Languages

Rudolph and Giesbrecht (2010) showed that symbolic approaches to language (i.e., discrete grammar formalisms) can be embedded in CMSMs. This suggests the CMSMs are compatible with both discrete (e.g., symbolic approaches) and continuous (e.g., numeric approaches) settings.

First, they showed how CMSMs determine whether a sequence of symbols belongs to a given regular language (i.e., given a sequence of symbols determine if it is accepted by a given finite state automaton).

Definition 2.1 (Finite State Automata). *A finite automaton is defined as* $\mathcal{A} = (Q, \Sigma, \delta, Q_I, Q_\mathrm{F})$ *where* $Q = \{q_1, \ldots, q_m\}$ *is a finite set of states,* Σ *is a finite set of input symbols,* $\delta \subseteq Q \times \Sigma \times Q$ *is the transition function from one state to another labeled by a symbol in* Σ*, and* Q_I *and* Q_F *are the sets of initial and final states, respectively. The language accepted by* $\mathcal{A}$ *is the set of strings (i.e., sequences of symbols)* $w \in \Sigma^*$ *accepted by* $\mathcal{A}$*. If we let zero, one, or more transitions from a state on the same symbol, the automaton is called a nondeterministic finite automaton. This time,* δ *is a map from* $Q \times \Sigma$ *to the power set of* Q*,* 2^Q*. (Hopcroft and Ullman, 1979, p. 17-20).* $\diamondsuit$

Eilenberg (1974) showed that to each symbol $\sigma \in \Sigma$ a transition matrix $[\![\sigma]\!] = M_\sigma$ of size $m \times m$ (where m is the number of states) can be assigned. If we assign to every symbol σ a matrix with:

$$M_\sigma(i,j) = \begin{cases} 1 \text{ if } (q_i, \sigma, q_j) \in \delta, \\ 0 \text{ otherwise.} \end{cases}$$

for $1 \leq i, j \leq m$, the matrix M_σ encodes all state transitions labeled by the input symbol σ. Likewise, for a sequence $w = \sigma_1 \ldots \sigma_k \in \Sigma^\star$, the matrix $M_w := [\![\sigma_1]\!] \ldots [\![\sigma_k]\!]$ encodes all state transitions labeled w. This matrix determines whether w belongs to a given regular language, that is if it is accepted by a given finite automaton $\mathcal{A}$.

It has been shown that by selecting an appropriate assignment $[\![\ \cdot\ ]\!]$ for a CMSM, and defining two vectors $\mathbf{v}_I$ and $\mathbf{v}_F$ as follows:

$$\mathbf{v}_\mathrm{I}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{I}, \\ 0 \text{ otherwise,} \end{cases} \qquad \mathbf{v}_\mathrm{F}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{F}, \\ 0 \text{ otherwise,} \end{cases}$$

for $1 \leq i, j \leq m$, w is accepted by the automaton $\mathcal{A}$ exactly if $\mathbf{v}_I M_w \mathbf{v}_F \geq 1$ (Rudolph and Giesbrecht, 2010). Of course, one can also define the two vectors $\mathbf{v}_I$ and $\mathbf{v}_F$ differently and a threshold value r to compare $\mathbf{v}_I M_w \mathbf{v}_F$ against the threshold value. Based on this idea, Rudolph and Giesbrecht (2010, p. 912) define the notion of *matrix grammars* as follows :

Definition 2.2 (Matrix Grammars). *Let Σ be an alphabet. A* matrix grammar *$\mathcal{M}$ of degree n is defined as the pair $\langle [\![\cdot]\!], AC\rangle$ where $[\![\cdot]\!]$ is a mapping from Σ to $n \times n$ matrices and $AC = \{\langle \boldsymbol{\alpha}_1, \boldsymbol{\beta}_1, r_1\rangle, \ldots, \langle \boldsymbol{\alpha}_\ell, \boldsymbol{\beta}_\ell, r_\ell\rangle\}$ with $\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\alpha}_\ell, \boldsymbol{\beta}_\ell \in \mathbb{R}^n$ and $r_1, \ldots, r_\ell \in \mathbb{R}$ is a finite set of* acceptance conditions*. The language generated by $\mathcal{M}$ (denoted by $L(\mathcal{M})$) contains a sequence of symbols $\sigma_1 \ldots \sigma_k \in \Sigma^*$ exactly if $\boldsymbol{\alpha}_i^\top [\![\sigma_1]\!] \ldots [\![\sigma_k]\!] \boldsymbol{\beta}_i \geq r_i$ for all $i \in \{1, \ldots, l\}$. We call a language L* matricible *if $L = L(\mathcal{M})$ for some matrix grammar $\mathcal{M}$.* ♢

Based on the above definition, regular languages are matricible by appropriately encoding to CMSMs.

Rudolph and Giesbrecht (2010) also studied other formal languages such as non-regular languages (e.g., $L(\mathcal{A}) = \{w | w = w^R\}$) and non-context free languages (e.g., $L(\mathcal{A}) = \{a^m b^m c^m |$
$m > 0\}$) and showed that some are matricible by appropriate encoding. Moreover, they showed that the intersection of two matricible langauges is also a matricible language. However, some formal languages still need to be further investigated. For instance, they conjectured that not all context-free languages are matricible as they have not been able to show that, for example, the language of all well-formed parenthesis expressions is matricible. Some other questions are also open, such as whether matricible languages are closed under concatenation and require more investigations.

2.3.3 Compositional Matrix-Space Models and Weighted Finite Automata

In this subsection, inspired by the definition of Weighted Finite Automata (WFA) (Schützenberger, 1961; Eilenberg, 1974; Berstel and Reutenauer, 1988) and their applications in NLP (Knight and May, 2009), we show a close correspondence between CMSMs and WFA. As discussed in the introduction chapter, this subsection is based on my following paper: Asaadi and Rudolph (2016). The motivation is to investigate how both CMSMs and WFA suit in natural language processing tasks. We first introduce a *semiring* (Eilenberg, 1974, p. 122-123) and WFA. The definition of WFA is taken from (Knight and May, 2009, p. 122) as it has a comprehensive overview of WFA.

Definition 2.3 (Semiring). *A semiring is a set $\mathbb{S}$ equipped with two binary operations: addition $\oplus$ and multiplication $\odot$, and two constant neutral elements 0 and 1 such that:*

- *$\langle \mathbb{S}, \oplus, 0\rangle$ is a commutative monoid.*
- *$\langle \mathbb{S}, \odot, 1\rangle$ is a monoid.*
- *the distributive property $(a \oplus b) \odot c = a \odot c \oplus b \odot c$ and $c \odot (a \oplus b) = c \odot a \oplus c \odot b$ hold for every $a, b, c \in \mathbb{S}$.*
- *$a \odot 0 = 0 \odot a = 0$ for every $a \in \mathbb{S}$.* ♢

Definition 2.4 (Weighted Finite Automata). *Weighted finite automata generalize classical automata in that transitions and states carry weights. These weights can be considered as the error of the transitions or amount of resources needed to execute the transitions. Let Σ be a finite alphabet. A weighted automaton $\mathcal{A}$ is a tuple of $(Q_{\mathcal{A}}, \lambda, \alpha, \beta)$ and defined over a semiring $(\mathbb{S}, \oplus, \odot, 0, 1)$. $Q_{\mathcal{A}}$ is a finite set of states, $\lambda : \Sigma \to \mathbb{S}^{Q_{\mathcal{A}} \times Q_{\mathcal{A}}}$ is the transition weight function, and, $\alpha : Q_{\mathcal{A}} \to \mathbb{S}$ and $\beta : Q_{\mathcal{A}} \to \mathbb{S}$ are two functions assigning to every state its initial and final weight. Thereby, for each transition $e = (q, \sigma, q')$, $\lambda(\sigma)_{q,q'}$ denotes the weight of the label σ associated with the transition e between q and q', which are the source and target state of the transition, respectively. Moreover, A* path *$\mathcal{P}$ in $\mathcal{A}$ is a sequence of transitions labeled with $\sigma_1 \dots \sigma_k$, in more detail:*

$$\mathcal{P} := p_0 \xrightarrow{\sigma_1} p_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_k} p_k$$

with $p_i \in Q_{\mathcal{A}}$. The weight *of $\mathcal{P}$ is defined as the $\odot$-product of the weights of the starting state, its transitions, and final state:*

$$\omega(\mathcal{P}) = \alpha(p_0) \odot \lambda(\sigma_1)_{p_0,p_1} \odot \cdots \odot \lambda(\sigma_k)_{p_{k-1},p_k} \odot \beta(p_k).$$

Now, the weight of a sequence $\sigma_1 \dots \sigma_k \in \Sigma^\star$ is the cumulative weight of all paths labeled with the sequence $\sigma_1 \dots \sigma_k$, which is computed as the $\oplus$-sum of the weights of the corresponding paths:

$$f_{\mathcal{A}}(\sigma_1 \dots \sigma_k) = \bigoplus_{\mathcal{P} \in P_{\mathcal{A}}(\sigma_1 \dots \sigma_k)} \omega(\mathcal{P}), \tag{2.5}$$

where $P_{\mathcal{A}}(\sigma_1 \dots \sigma_k)$ denotes the (finite) set of paths in $\mathcal{A}$ labeled with $\sigma_1 \dots \sigma_k$. So, the function $f_{\mathcal{A}} : \Sigma^\star \to \mathbb{S}$ which maps every strings in $\Sigma^\star$ to $\mathbb{S}$ is called the behavior of the weighted automaton $\mathcal{A}$, also written as $\| \mathcal{A} \|$. ♢

In this thesis, we assume that the semantic space $\mathbb{S}$ is the set of the real numbers $\mathbb{R}$ with the usual multiplication and addition. Fig. 2.12 illustrates an example of WFA over $\Sigma = \{a, b\}$. Inside each state, there is a tuple of the name, initial and final weight of the state, respectively. As an example, for $w = ab$ we have: $f_{\mathcal{A}}(w) = 1 \times 1.5 \times 3 \times 0.8 + 1 \times 1.5 \times 2 \times 0.2 + 2 \times 4 \times 3 \times 0.5$.

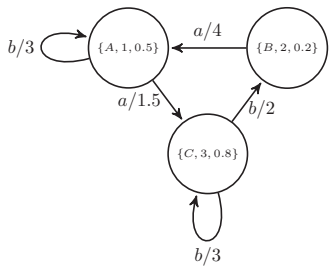


Figure 2.12: Example of WFA $\mathcal{A}$ with three states and the alphabet $\Sigma = \{a, b\}$.

Generally, in many NLP tasks, we need to estimate functions that map an arbitrary sequence of words (e.g., a sentence) in natural language to some semantic space. Using WFA, an extensive class of these functions can be defined, which assign sequences to

real numbers. Assigning word sequences of natural language to real numbers is popular in NLP tasks, e.g., assigning a real-valued sentiment score to a phrase in the sentiment analysis task.[1] To show that CMSMs are also capable of defining such functions to be used in NLP tasks, we first introduce the notion of a *graded matrix grammar* (Asaadi and Rudolph, 2016, p. 72), which constitutes a slight variation of matrix grammars (Rudolph and Giesbrecht, 2010). In graded matrix grammars, instead of the "yes or no" decision, if a sequence is part of a language, a real-valued score is assigned. Then, we show the correspondence between the CMSMs and WFA.

Definition 2.5 (Graded Matrix Grammars). *Let Σ be a vocabulary.*[2] *A graded matrix grammar $\mathcal{M}$ of degree n is defined as the tuple $\langle [\![\cdot]\!], \Sigma, \mathbf{u}, \mathbf{v}\rangle$ where $[\![\cdot]\!]$ is a function mapping words in Σ to $n \times n$ matrices of real numbers. Moreover, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. Then we map each sequence of words $\sigma_1 \dots \sigma_k \in \Sigma^\star$ to a real number (called the value of the sequence) using the target function $\varphi : \Sigma^\star \to \mathbb{R}$ defined by:*

$$\varphi(\sigma_1 \dots \sigma_k) = \mathbf{u}^\top [\![\sigma_1]\!] \dots [\![\sigma_k]\!] \mathbf{v}. \tag{2.6}$$

$\diamondsuit$

As discussed above, in WFA, for a behavior of weighted automaton f, a value $f(\sigma_1 \dots \sigma_k)$ is the sum of all possible paths labeled with $\sigma_1 \dots \sigma_k \in \Sigma^\star$. However, this computation can be described via matrices by the fact that a walk over a graph corresponds to a matrix multiplication (Eilenberg, 1974, p. 137, Cor. 6.2.). More precisely, for every $\sigma \in \Sigma$, let $E_\sigma \in \mathbb{R}^{Q_\mathcal{A} \times Q_\mathcal{A}}$ be the transition matrix of σ: $[E_\sigma]_{pq} = \sum_{e \in P_\mathcal{A}(p,\sigma,q)} \lambda(\sigma)_{p,q}$, where $P_\mathcal{A}(p, \sigma, q)$ is the set of all transitions labeled with σ from p to q. Also, the vectors $\mathbf{I} \in \mathbb{R}^{Q_\mathcal{A}}$ and $\mathbf{T} \in \mathbb{R}^{Q_\mathcal{A}}$ show the start and final weights of the states in $Q_\mathcal{A}$, respectively. Then, Equation 2.5 can be equally expressed as follows in terms of matrices with elements in $\mathbb{R}$:

$$f_\mathcal{A}(\sigma_1 \dots \sigma_k) = \mathbf{I}^\top E_{\sigma_1} \dots E_{\sigma_k} \mathbf{T}. \tag{2.7}$$

Hence, we see the correspondence between Equation 2.6 and 2.7. Fig. 2.13 shows this correspondence.

As mentioned above, many NLP tasks require mapping word sequences in natural language to real numbers. For instance, a phrase p="very good" can be assigned to a real value in $[-1, 1]$ as its sentiment score in the task of sentiment analysis. Consider each phrase p in a natural language with its target real-valued score r. If we extract the vocabulary of the language as a finite alphabet Σ in an automaton, then p would be a string in $\Sigma^\star$. The function $[\![\cdot]\!]$ in $\mathcal{M}$ applied over the words constructs $n \times n$ transition matrices of symbols in Σ in the automaton. Here, n can be the number of states of the automaton. So, estimating the function φ in graded matrix grammar corresponds to

[1] Sentiment analysis task refers to determining the attitudes towards a topic. Attitudes include evaluative judgment, such as positive (score 1) or negative (score -1). Details are explained in the next Subsection.

[2] Vocabulary is equivalent to the alphabet in finite automata. Words in vocabulary correspond to symbols in the alphabet, and a sequence of words corresponds to a sequence of symbols in the automaton, called string.

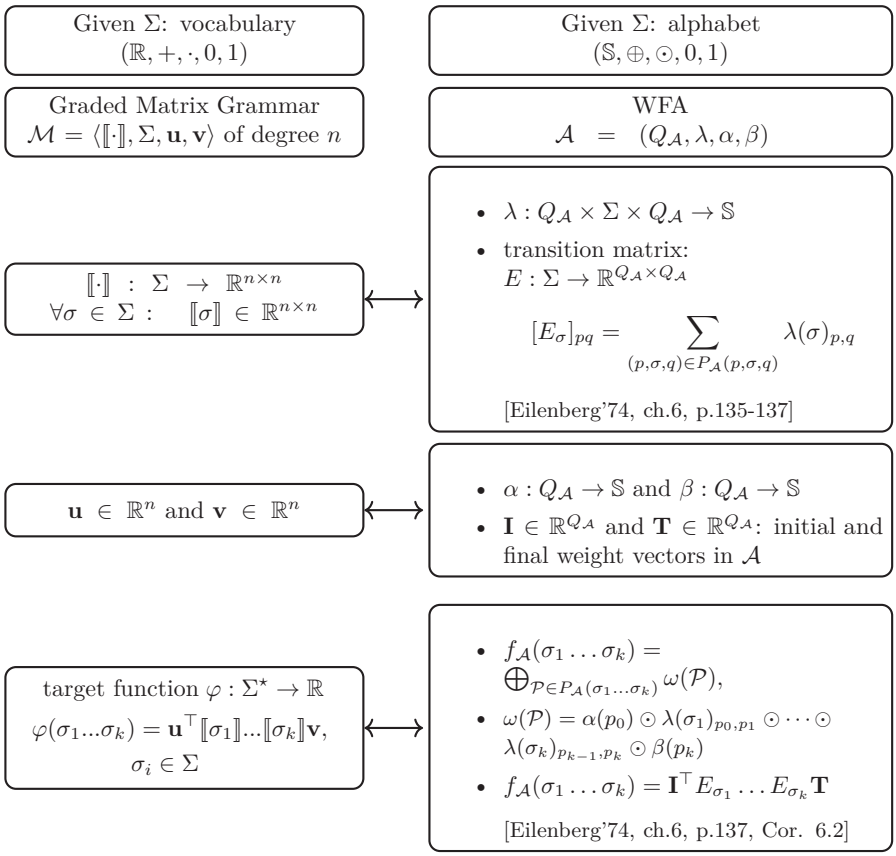


Figure 2.13: Correspondence between CMSM and WFA.

estimating the target function of the automaton $f_{\mathcal{A}}$, which computes exactly the score of a string translated from the phrase p in a natural language. That is, the representation of a string is obtained with multiplication of transition matrices of its symbols, which results in a new representation matrix for the string. Then, suitable predefined vectors $\mathbf{I}$ and $\mathbf{T}$ translate the resulting matrix to a real value, which denotes the score of the associated phrase p in the natural language.

Learning techniques are needed to learn WFA to perform in NLP tasks. The problem of learning WFA in NLP is to find an automaton that closely approximates a target function $f_{\mathcal{A}}$ using a finite set of pairs of strings and their target values $\{(x_1, y_1), \dots, (x_d, y_d)\}$ as the training dataset (Balle and Mohri, 2015). Given a number of states n, by learning WFA, one obtains an automaton that is a tuple $\mathcal{A} = \langle \mathbf{I}, \mathbf{T}, \{E_\sigma\}_{\sigma \in \Sigma} \rangle$, which approximates the target function $f_{\mathcal{A}}(x)$ and generalizes to predict the target values of previously unseen strings. Since WFA encode CMSMs and based on the close correspondence between them, learning a graded matrix grammar to determine the score of phrases in NLP tasks can be mapped to the problem of learning a weighted automaton. Conversely, if there is an efficient learning algorithm for learning CMSMs to best approximate a target function φ in graded matrix grammars, learning WFA can be mapped to the problem of learning CMSMs in NLP.

In fact, several methods for learning WFA have been described (Balle et al., 2014;

Balle and Mohri, 2015), for instance, based on the principles of expectation maximization (Dempster et al., 1977) and method of moments (Pearson, 1894). However, in the context of the NLP tasks investigated by us, the method of moments techniques performed very poorly in terms of scalability and accuracy. Hence, in this thesis, we reverted to more generic machine learning techniques for learning CMSMs in NLP.

2.3.4 Applications of Compositional Matrix-Space Models

CMSMs can be used as an alternative to VSMs in various NLP tasks. This means that the semantics of the linguistic units can be represented in matrix-space. These models are particularly suitable for tasks that require to compute categorical or scalar values as final labels for given input samples. For instance, computing the semantic similarity degree between a pair of phrases, determining if a phrase entails or contradicts another phrase in the textual entailment task, or detecting the sentiment class of a given phrase in sentiment analysis task can be investigated using CMSMs.

Sentiment analysis is one of the most popular tasks in NLP. The term *sentiment analysis* refers to different but related problems. In general, the task refers to determining the attitudes towards a topic. Attitudes include evaluative judgment (can also be called valence), such as positive or negative, or emotional states, such as happiness, sadness, anger, etc. (Mohammad, 2016). With the increasing importance of review websites for marketing, a lot of research has been done in sentiment analysis to determine the attitude of people about a special topic automatically. In this thesis, we use sentiment analysis to refer to determining the polarity (positive, negative, neutral) of a piece of text. For example, "a very good movie" indicates a very positive sentiment about the movie while "a bad movie" carries a negative polarity. Therefore, the task is to determine the polarity of a piece of text from multiple classes (negative, positive, neutral) and to determine intensities (e.g., weak, medium, extreme) for positive and negative polarities. Real-valued scores can represent the polarity and intensity in a continuous interval instead of categorical representations, which is also called fine-grained sentiment analysis.

Sentiment analysis can be applied to a single word or a sequence of words such as phrases and sentences. There are many aspects that must be considered in analyzing complex sequences. First, different parts-of-speech, such as adjectives and adverbs (e.g., negators and intensifiers), affect the total sentiment of the sequence differently. Second, a different order of words may result in a different sentiment score. Third, the compositionality of the constituents of a sequence determines the meaning of the whole sequence. For instance, a negator followed by an adjective changes the meaning of the pair negator–adjective phrase. Yessenalina and Cardie (2011) and Irsoy and Cardie (2015) showed an application of CMSMs in sentiment analysis and how they capture the above properties in this task. They proposed supervised machine learning techniques for learning CMSMs in sentiment analysis of short sequences. The proposed methods learn a matrix representation for each word that captures the compositionality properties of natural language. In this thesis, we investigate CMSMs in sentiment analysis as one of the applications of CMSMs in NLP to address the issues discussed in the previous works.

According to Mitchell and Lapata (2010), if a model cannot capture the compositionality of basic linguistic units, such as short phrases, it most likely will not be able to

capture the meaning composition of more complex linguistic units such as sentences. Multi-Word Expressions (MWEs) are short compounds with two or more words showing a range of semantic idiomaticity, which refers to non-compositional expressions, and statistical idiomaticity, which refers to collocations[1] like *social media*, and other types of idiomaticity, such as syntactic, pragmatic and lexical idiomaticity (Baldwin and Kim, 2010), which are outside the scope of this thesis. We are only interested in semantic idiomaticity (non-compositional expressions).

Detecting the (non-)compositionality of MWEs is especially important in meaning-related NLP tasks, such as phrase-based statistical machine translation (Kordoni and Simova, 2014; Enache et al., 2014; Weller et al., 2014) and word sense disambiguation (McCarthy et al., 2003; Finlayson and Kulkarni, 2011). Therefore, approaches for detecting the (non-)compositionality of compounds in NLP are needed.

One approach is to train CDSMs to capture the compositional compound representation from its components' representations. Recall that based on the principle of compositionality the meaning of a compositional expression can be determined by the meaning of its constituents. Thus, if the trained model cannot closely approximate the representation of a compound from its constituents' representations, it recognizes the compound as non-compositional. This way (non-)compositionality of MWEs can be detected (Farahmand et al., 2015). Gold standard evaluation datasets for the compositionality detection task have been introduced to evaluate the performance of the trained models using this approach. A gold standard dataset is a dataset that is accepted as a reliable and standard reference and the best available resource for evaluating methods and models. It is usually annotated and corrected by humans.

These evaluation datasets consist of pairs of compounds and their compositionality degrees. The degrees can, for example, take an integer value between 1 to 5 with 1 as fully compositional and 5 as fully non-compositional. Datasets are created with human judgments on the compositionality of compounds.

Therefore, the task of (non-)compositionality detection is one of the popular tasks in NLP in which different composition methods in CDSMs can be evaluated on their ability to capture the (non-)compositionality of MWEs. The best model that can identify the (non-)compositional MWEs will be employed in downstream NLP tasks.

In this thesis, as another application of CMSMs, we investigate these models in the compositionality detection task in NLP. We consider these two tasks, sentiment analysis and compositionality detection, for practical investigations of CMSMs since the compositionality properties of natural language play an important role in such tasks.

2.4 Examining Semantic Composition Methods

As mentioned in Section 2.1, influential approaches to meaning representations have involved DSMs in vector space, which produce word vectors that capture the patterns of co-occurrence in a given context (i.e., the distributional representations) (Baroni and Lenci, 2010). In this way, words with similar distributions in a given context tend to have close representations in vector space. In addition, different composition

[1]Collocation is a group of words (e.g., a two-word phrase) that co-occur more commonly in a given context.

methods have been introduced to achieve the representation of larger linguistic units (e.g., phrases) from word representations and capture semantic compositionality. Examining different semantic composition methods in capturing the semantic representation of larger linguistic units has been a main challenge in NLP, and different approaches have been proposed.

In the study of examining word representation models, a common approach is exploiting their ability to rank pairs of words in natural language by their closeness in meaning. Closeness is a measure of how close two term are in terms of their semantics; that is, two terms are considered to be semantically close if there is a sharing of some meaning (Mohammad, 2008). For instance, the two words *teacher* and *tutor* are closer in meaning than the two words *teacher* and *fish*. Sharing of meaning is defined based on the lexical–semantic relations (i.e., the semantic relationship between the lexical items; Cruse, 1986). He points out that the number of lexical–semantic relations is innumerable, however, certain relations, such as synonymy, hypernymy (hyponymy), meronymy (holonymy), and antonymy, received more attention as they are systematic and recur in a number of pairs of related terms. Morris and Hirst (2004) classify the lexical–semantic relations into classical and non-classical relations, as described below.

Classical Lexical–Semantic Relations:

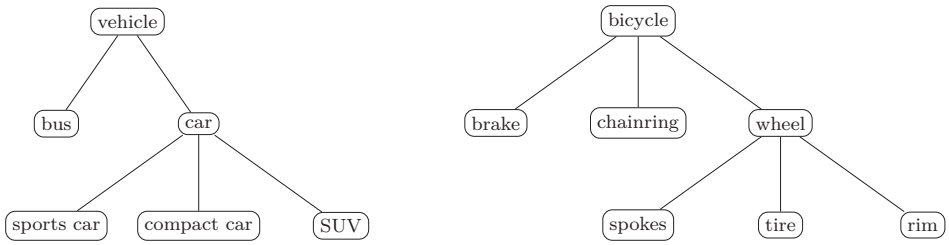
The classical category refers to the systematic relations that form predetermined structures and depend on shared properties (Morris and Hirst, 2004). They can be generally classified into hierarchical and non-hierarchical relations.

Hierarchical relations: These relations can be represented using graph structures.

- Hyponymy and Hypernymy (Taxonomy): Hyponymy is the well-known *is-a* relation that associates a term of a certain type to another term of a more general type. The more general term is called a *hypernym*, which subsumes the *hyponyms*. A chain of hyponyms defines a hierarchical taxonomy of elements: *Sports car* is a hyponym of *car* and a *car* is a kind of *vehicle* (Riemer, 2010). Transitivity is one of the main properties of taxonomy (Fig. 2.14(a)).
- Meronymy and Holonymy: Meronymy and holonymy relations are used to define the part–whole relationship between lexical items. A meronym denotes a part, and a holonym denotes the whole (Riemer, 2010). *spokes* is a part (or meronym) of *wheel*, and *wheel* is a meronym of *bicycle*. Conversely, *wheel* is a holonym of *spokes* (Fig. 2.14(b)).

Non-Hierarchical relations: These relations are not represented using graph structures. Instead, a ternary relationship is defined.

- Synonymy: Two terms are synonymous if they are substitutable in sentences in any context without changing the meaning of the sentences or their *truth conditions* (Jurafsky and Martin, 2014). Edmonds and Hirst (2002) argue that absolute synonymy is quite rare, and almost no two terms are interchangeable in all of their contexts; therefore, terms are *near synonymous*. A ternary relation $R(T1, T2, C)$ is defined as the terms $T1$ and $T2$ are synonyms in the context C. In the rest of the thesis, we use the term synonymy to refer to near synonymy.



(a) Example of hyerpnymy/hyponymy lexical–semantic relations.

(b) Example of meronymy/holonymy lexical–semantic relations.

Figure 2.14: Examples of hierarchical lexical–semantic relations.

- Antonymy: This term refers to opposition or a relationship of incompatibility between two terms in relation to some given aspects of contrast, for instance, *hot–cold* in terms of temperature and *long–short* in terms of length (Riemer, 2010). A ternary relation $R(T1, T2, C)$ is defined as the terms $T1$ and $T2$ are antonyms in relation to the aspect C.

Non-Classical Lexical–Semantic Relations:

Non-classical relations "do not depend on the shared properties as required of classical relations" (Morris and Hirst, 2004, p. 2). There are implicit connections between the terms (Zesch and Gurevych, 2010), and they cannot be classified into hierarchical and non-hierarchical relations. They do not form any structure, such as a graph structure. A few types observed by Morris and Hirst (2004, p. 4) are as follows:

- Commonly co-occurring words (associated words; *coffee–cup, car–drive*);
- Problem solution or cause (*homeless–drunk*);
- Positive qualities (*brilliant–kind*); and
- Negative qualities (*homeless–alcoholic*).

Closeness in meaning, defined based on the lexical–semantic relations, can be of two kinds: *semantic similarity* and *semantic relatedness*. While these terms may be used interchangeably in some contexts, it is important to know their distinction. Semantically similar terms tend to share several properties. For example, *apples* and *bananas* are both edible, grow on trees, have seeds, and so on. In terms of lexical–semantic relations, two terms are considered to be semantically similar if there is a hypernymy, (co-)hyponymy, synonymy, or antonymy relationship between them as these relations share common properties (Budanitsky and Hirst, 2001; Mohammad, 2008; Agirre et al., 2009; Baroni and Lenci, 2010). For instance, *bananas* and *apples* are similar as they are co-hyponyms of *fruits*, as their hypernym, and share common properties; moreover, *auto* and *car* are similar as they are synonyms.

Semantically related terms may not have many shared properties, but have at least one classical or non-classical relation between them that allows them to be considered semantically close. For example, consider the meanings of *coffee* and *cup*. They are not similar as they have practically no common properties: *coffee* is a drink, while *cup* is an object with a specific shape. However, both are related as they are associated in the

world by commonly co-occurring in a shared event of *drinking a cup of coffee*. Similarly, the words *surgeon* and *scalpel* are not similar as *surgeon* is a profession and *scalpel* is an object, but they are related, as the former uses the latter for their work (Mohammad and Hirst, 2005; Budanitsky and Hirst, 2006; Mohammad, 2008; Harispe et al., 2015). Therefore, two terms are considered semantically related if there is any lexical–semantic relation between them, classical or non-classical.

According to Kolesnikova and Gelbukh (2015) and Mel'cuk (1996), the term *Lexical Function (LF)* is used to describe the lexical–semantic relations between lexical items in natural language in the mathematical sense. This is defined as a mapping from a lexical item w_0, called the LF argument, to a set of items $\{w_1, \ldots, w_n\}$ that have a particular relation (association) with w_0. This can be represented as follows:

$$LF(w_0) = \{w_i\},\ 1 \leq i \leq n.$$

For instance, synonymy relation for *car* will be described as $syn(car) = \{auto, automobile\}$. A lexical item w can have different types of lexical–semantic relations, expressed as follows: $\{LF_1(w), LF_2(w), \ldots, LF_m(w)\}$. If a lexical item w has different relations, defined as relatedness:

$$\mathcal{R}(w) = \{LF_1(w), LF_2(w), \ldots, LF_l(w)\},$$

and a set of different lexical relations, defined as similarity:

$$\mathcal{S}(w) = \{LF_1(w), LF_2(w), \ldots, LF_k(w)\},$$

then S is a subset of R: $S(w) \subset R(w)$ and $k < l$; that is, relatedness is a broader class than similarity is. For instance, $R(car) = \{\{vehicle\}, \{auto, atuomobile\}, \{road, drive\}\}$ and $S(car) = \{\{vehicle\}, \{auto, automobile\}\}$. Note that there is an implicit association between the two terms *road* and *drive* as they co-occur commonly in a shared event of *drive a car on the road*, which is considered a non-classical relation. In general, similar terms are also related but there are terms that are not similar but strongly related (associated) and co-occur in the shared events, such as terms with non-classical relations (e.g., *surgeon–scalpel*, *drive–car*).

For a lexical item w, an expression $LF(w)$ can be defined for each lexical relation, that is, $hypo(w)$, $hyper(w)$, $mero(w)$, $holo(w)$, $syn(w)$, $ant(w)$, $non\text{-}classical(w)$. Now, if a lexical item w_0 is in $\{hypo(w) \cup hyper(w) \cup mero(w) \cup holo(w) \cup syn(w) \cup ant(w) \cup non\text{-}classical(w)\}$, the two items w and w_0 are absolutely related, but they may not be similar. If the lexical item w_0 is in $\{hypo(w) \cup hyper(w) \cup syn(w) \cup ant(w)\}$, they are also considered similar items.

The distributional hypothesis states that "words that occur in the same contexts tend to have similar meanings" (Pantel, 2005, p. 126). This hypothesis mentions similarity and not relatedness, and the works based on the distributional hypothesis use the term semantic similarity rather than semantic relatedness. However, as Mohammad and Hirst (2005) and Mohammad (2008) point out, the distributional hypothesis is generic enough to contain both relatedness and similarity. Recall the co-occurring words example *surgeon–scalpel* that are considered a non-classical relation, and therefore, belong to the semantic relatedness. They commonly co-occur in the same contexts. Thus, based on the

distributional hypothesis, DSMs produce close vector representations for semantically related words (as well as similar terms).

Therefore, the quality of the semantic representation of words produced by representation models can be evaluated by computing their relatedness (i.e., the strength of their semantic association), as their closeness in meaning in vector space using some measures. The ability to assess semantic relatedness is central to the use and understanding of natural language (Hutchison, 2003; Huth et al., 2016). Therefore, the quantification of semantic relatedness in NLP is needed for examining word representation models. For this, gold standard evaluation datasets consisting of pairs of terms with semantic relatedness scores are created. A gold standard dataset is a dataset that is accepted as a reliable and standard reference and the best available resource for evaluating methods and models. It is usually annotated and corrected by human experts.

Existing gold standard evaluation datasets of semantic relatedness in English, such as that by Finkelstein et al. (2002), only focus on single words (unigrams). However, the concept of semantic relatedness applies to larger linguistic units produced by composition, such as phrases or sentences. Therefore, we argue that semantic relatedness can be used to evaluate methods of semantic composition. Previous studies focused on the evaluation of semantic composition using semantic similarity on phrase level (e.g., Turney (2012) and Mitchell and Lapata (2010)). In this thesis, however, we focus on semantic relatedness, not only because it is the broader class subsuming semantic similarity (Rubenstein and Goodenough, 1965; Resnik, 1995; Budanitsky and Hirst, 2006) but also because many psychological and neuro linguistic studies have demonstrated the importance of semantic relatedness. These studies show that the human brain stores information in a thematic manner (based on relatedness) rather than based on similarity (Hutchison, 2003; Huth et al., 2016). Moreover, Hill et al. (2015) suggest that relatedness judgments have broader use in studies of human semantic cognition. Another limitation of similarity is that it can be only defined between terms categorized as the same Part-Of-Speech (POS). In contrast, two terms can be related even if they represent different parts of speech (Zesch and Gurevych, 2010). For instance *surgeon* and the verb *operate* belong to different parts of speech, but they are related as they commonly co-occur and have many common co-occurring words, such as *scalpel*, *surgery*, and *patient*.[1]

Thus, to investigate semantic composition methods using closeness in meaning, gold standard semantic relatedness evaluation datasets must be developed, which is one focus of this thesis.[2]

Relatedness measurements may vary across different fields (Deza and Deza, 2014). In computational linguistics, the relatedness measure can be formalized using a semantic distance measure (Harispe et al., 2015), such as distributional distance, and the relatedness score can be obtained by inversing the distance values (Budanitsky and Hirst, 2006). The study of different semantic relatedness measures is out of the scope of this thesis, and the interested reader is referred to Johnson (1995), Budanitsky and Hirst

[1] We are aware that some words may have several meanings depending on their context. In this thesis, whenever we compare two words, we consider their meanings in the same context.

[2] Antonymy is a broad concept in lexical–semantic relations. Some studies, such as that by Budanitsky and Hirst (2001), do not consider antonymy in the semantic similarity category. In this thesis, we are not concerned with the study of antonymy and exclude it in the development of our semantic relatedness dataset.

(2006) and Zesch and Gurevych (2010) for further information. In this thesis, we use the cosine value between the two term representations to refer to their relatedness (closeness) degree. The cosine of the two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is computed as:

$$\cos(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1.\mathbf{v}_2}{\parallel \mathbf{v}_1 \parallel \parallel \mathbf{v}_2 \parallel},$$

where $\parallel \cdot \parallel$ computes the Euclidean norm of a vector $\mathbf{v}$ of size d as follows:

$$\parallel \mathbf{v} \parallel = \sqrt{\mathbf{v}(1)^2 + \cdots + \mathbf{v}(d)^2}.$$

The cosine value ranges between -1 and 1, and higher cosine values indicate higher closeness between term meaning representations.

Chapter 3

Task-Specific Learning Approach to CMSMs

Compositional representation models were introduced to capture the representation of compositional expressions, such as phrases (sequences of words), for downstream NLP tasks. Among different models, we discussed advantageous theoretical properties of CMSMs in Chapter 2, demonstrating their principled suitability and expressivity in NLP tasks. The model has been shown to cover numeric (e.g., VSMs) and symbolic approaches. Moreover, the plausibility of CMSMs in the light of structural and cognitive considerations has been supported. In view of these advantageous properties, CMSMs seem to be a suitable candidate for a diverse range of NLP tasks. These established beneficial properties motivate a practical investigation of CMSMs in NLP applications, which is the focus of this chapter.

For practical applicability, machine learning methods are needed to train word matrix representations from available training datasets for different NLP tasks. Thus, similar to word vectors, each word matrix is supposed to contain semantic information about the word. With a few notable exceptions (Yessenalina and Cardie, 2011; Irsoy and Cardie, 2015), learning CMSMs has remained largely unexplored and this topic is discussed in detail in this chapter.

Depending on the NLP application, the information of interest encoded in the matrices needs to be utilized in different forms. In Section 3.1, we discuss various possible ways (forms) of utilizing the semantic information encoded in the matrix representation of words and phrases. We consider whether the required information for the NLP application should be in the form of a vector, a scalar, or just a Boolean value. In Chapter 2.3.4, we introduced two applications of CMSMs in NLP: sentiment analysis and compositionality detection. As explained before, we use sentiment analysis to refer to determining the polarity (positive, negative, neutral) of a piece of text. Compositionality detection is also concerned with detecting if a given compound (two-word sequence) is compositional or non-compositional. In this chapter, we aim at investigating the performance of CMSMs on meaning composition in these two NLP applications. We consider these two tasks for practical investigations since compositionality properties of natural language play an important role in such tasks. Therefore, we address the problem of learning CMSMs using machine learning techniques for sentiment analysis and compositionality detection[1] tasks in Sections 3.2 and 3.3, respectively, and review the related work. In the sentiment analysis, we look into a learning technique for compositional phrase scoring models; that is, phrases are "scored" by scalar values. In the latter task, we address the scenario aimed at simulating a compositional vector representation for words and phrases by means of CMSMs. In each section, we evaluate the performance of the learned models empirically against other compositional models from the literature. By means of these investigations, we show the following results:

- There are scalable methods for learning CMSMs; and
- In terms of model quality, the learned models are competitive with other baseline models and state-of-the-art CMSM-based models, while requiring significantly fewer training parameters.

[1]This task is also called *compositionality prediction* task.

3.1 Utilizing Matrix Representations in CMSMs

As discussed in Chapter 2, Rudolph and Giesbrecht (2010) have argued in favor of using quadratic matrices as representatives for the meaning of words and – by means of composition – phrases (i.e., word sequences). The matrix representation of a phrase thus obtained then arguably carries semantic information encoded in a certain way. This way of representation necessitates a "decoding step", where we utilize the information of interest encoded in the matrix representations in different forms: a vector, a scalar, or just a Boolean value.

Vectors

Vectors can represent various syntactic and semantic information of words and phrases, and are widely used in many NLP tasks. The information in matrix representations in CMSMs can be utilized in a vector shape allowing for their comparison and integration with other NLP vector-space approaches. There are numerous options to obtain vectors from matrices, among them the different functions $\chi : \mathbb{R}^{m\times m} \to \mathbb{R}^n$, introduced in Chapter 2.3.1.

One alternative option can be derived from the neurological motivation laid out in Section 2.3, where we assume that mental states correspond to vectors and word matrices encode mental state transformations. If we assume that the perception of a phrase will always start from a fixed initial mental state $\boldsymbol{\alpha} \in R^m$, then the mental state after perceiving $s = \sigma_1 \dots \sigma_k$ will be $\boldsymbol{\alpha}^\top M_{\sigma_1} \dots M_{\sigma_k} = \boldsymbol{\alpha}^\top M_s$. Consequently, one simple but plausible vector mapping operation would be given by the function $\chi_{\boldsymbol{\alpha}}$ where the vector $\mathbf{v}$ associated with a matrix M is

$$\mathbf{v} = \chi_{\boldsymbol{\alpha}}(M) = \boldsymbol{\alpha}^\top M.$$

Scalars

Scalars (i.e., real values) may also represent semantic information in NLP tasks, such as semantic similarity degree in similarity tasks or sentiment score in sentiment analysis. Also, the information in scalar form requires less storage than matrices or vectors. To map a matrix $M \in \mathbb{R}^{m\times m}$ to a scalar value, we may employ any m^2-ary function, which takes as input all entries of M and delivers a scalar value.

There are plenty of options for such a function. In this thesis, we focus on the class of functions brought about by two mapping vectors $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $\boldsymbol{\beta} \in \mathbb{R}^m$, mapping a matrix M to the scalar value r as follows:

$$r = \boldsymbol{\alpha}^\top M \boldsymbol{\beta}.$$

Again, we can motivate this choice along the lines of neurological plausibility described in Chapter 2.3. If, as argued, $\boldsymbol{\alpha}$ represents an "initial mental state" then, for a sequence s, the vector $\mathbf{v}_s = \boldsymbol{\alpha}^\top M_s \in \mathbb{R}^m$ represents the mental state after receiving the sequence s. Then $r_s = \boldsymbol{\alpha}^\top M_s \boldsymbol{\beta} = \mathbf{v}_s \boldsymbol{\beta}$ is the scalar obtained from a linear combination of the entries of $\mathbf{v}_s$, that is $r_s = b_1 \cdot \mathbf{v}(1) + \ldots + b_m \cdot \mathbf{v}(m)$, where $\boldsymbol{\beta} = (b_1 \quad \cdots \quad b_m)$.

Clearly, choosing appropriate "mapping vectors" $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ dependents on the NLP task and the problem to be solved. However, it turns out that with a proper choice of the token-to-matrix mapping, we can restrict $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to a very specific form. To this end, let

$$\boldsymbol{\alpha} = \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\beta} = \mathbf{e}_m = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

which only moderately restricts the expressivity of our model as made formally precise in the following theorem (Asaadi and Rudolph, 2017, p. 180).

Theorem 3.1. *Given matrices $M_1, \ldots, M_\ell \in \mathbb{R}^{m \times m}$ and vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^m$, there are matrices $\hat{M}_1, \ldots, \hat{M}_\ell \in \mathbb{R}^{(m+1)\times(m+1)}$ such that for every sequence $i_1 \ldots i_k$ of numbers from $\{1, \ldots, \ell\}$ holds*

$$\boldsymbol{\alpha}^\top M_{i_1} \ldots M_{i_k} \boldsymbol{\beta} = \mathbf{e}_1^\top \hat{M}_{i_1} \ldots \hat{M}_{i_k} \mathbf{e}_{m+1}$$

$\diamondsuit$

Proof. If $\boldsymbol{\alpha}$ is the zero vector, all scores will be zero, so we can let all $\hat{W}_h$ be the $(m+1) \times (m+1)$ zero matrix.

Otherwise, let W be an arbitrary $m \times m$ matrix of full rank, whose first row is $\boldsymbol{\alpha}$ (i.e., $\mathbf{e}_1^\top W = \boldsymbol{\alpha}^\top$). Now, let

$$\hat{M}_h := \begin{pmatrix} W M_h W^{-1} & M M_h \boldsymbol{\beta} \\ 0 \;\cdots\; 0 & 0 \end{pmatrix}$$

for every $h \in \{1, \ldots, \ell\}$. Then, we obtain

$$\hat{M}_g \hat{M}_h = \begin{pmatrix} W M_g M_h W^{-1} & W M_g M_h \boldsymbol{\beta} \\ 0 \;\cdots\; 0 & 0 \end{pmatrix}$$

for every $g, h \in \{1, \ldots, \ell\}$. This leads to

$$\begin{aligned} & \mathbf{e}_1^\top \hat{M}_{i_1} \ldots \hat{M}_{i_k} \mathbf{e}_{m+1} \\ = \; & \mathbf{e}_1^\top W M_{i_1} \ldots M_{i_k} \boldsymbol{\beta} \\ = \; & \boldsymbol{\alpha}^\top M_{i_1} \ldots M_{i_k} \boldsymbol{\beta}. \end{aligned}$$

q.e.d.

In words, this theorem guarantees that for every CMSM-based scoring model with arbitrary vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ there is another such model (with dimensionality increased by one), where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are distinct unit vectors. This theorem justifies our choice mentioned above.

Boolean Values

Matrix representations can also be mapped to boolean values. Any function $\zeta : \mathbb{R}^{m \times m} \to \{\text{true}, \text{false}\}$ can be seen as a binary classifier which accepts or rejects a sequence of

tokens as being part of the formal language L_ζ defined by

$$L = \{\sigma_1 \dots \sigma_k \mid \zeta([\![\sigma_1]\!] \dots [\![\sigma_k]\!]) = \text{true}\}.$$

One option for defining such a function ζ is to first map to a scalar (for instance using the mapping discussed before) and then compare that scalar against a given threshold value as defined in *matrix grammars* in Chapter 2.3.2, Definition 2.2.

3.2 Learning Approach to CMSMs in Sentiment Analysis

We propose a supervised learning approach to CMSMs in the task of sentiment analysis based on linear regression, which determines the real-valued sentiment score of short phrases. Training CMSMs using machine learning algorithms yields a low-dimensional real-valued quadratic matrix for each word. Since we consider the task of sentiment analysis, word representations must be trained to contain sentiment-related and compositional information.

Gradient descent (Berstsekas, 1999; Cauchy, 1847) is an iterative optimization algorithm, which is applied to linear and nonlinear problems in machine learning. In gradient descent, the goal is to find the local minimum/maximum (i.e., local optimum) of a loss function (also called objective function) by taking steps proportional to the negative/positive gradient of the function at the current point toward the local optimum. The loss function computes the error between the predicted and target values in the training set.

There are several variants of gradient descent optimization methods. One basic distinction made is that of batch as opposed to stochastic learning: given a set of training examples, in batch gradient descent, at each iteration parameter updates are done after seeing all training examples, while in stochastic gradient descent parameters are updated after seeing each training example.

We found gradient descent to be a suitable optimization method for learning CMSMs in sentiment analysis. We consider two variants of CMSM learning in predicting the scores of phrases in the sentiment analysis task.

3.2.1 Gradient Descent-based Matrix-Space Models

This variant of learning algorithm for CMSMs in sentiment analysis, called Gradient descent-based Matrix Space Models (GMSM), employs the standard gradient descent optimization technique to train the word matrices. As illustrated in Fig. 3.1(a), the input to the learning algorithm is a training dataset containing pairs (s_i, ω_i) of phrases s_i with associated real values ω_i for $i \in \{1, \dots, N\}$, where N is the size of the training dataset. $s_i = \sigma_1 \dots \sigma_{k_i}$ is a phrase consisting of a sequence of k_i words and ω_i is a scalar value representing s_i's associated target score. The learning algorithm optimizes a loss function by training a set of parameters of a model using the training dataset. In our algorithm, the model parameters to train are the word matrices $[\![\sigma_j]\!] \in \mathbb{R}^{m \times m}$ for each word σ_j and the mapping vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^m$. Recall that in sentiment analysis we map the phrases to scalars the same way as described in Section 3.1. As justified by Theorem 3.1, we fix the mapping vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to unit vectors $\mathbf{e}_1$ and $\mathbf{e}_m$, respectively, which

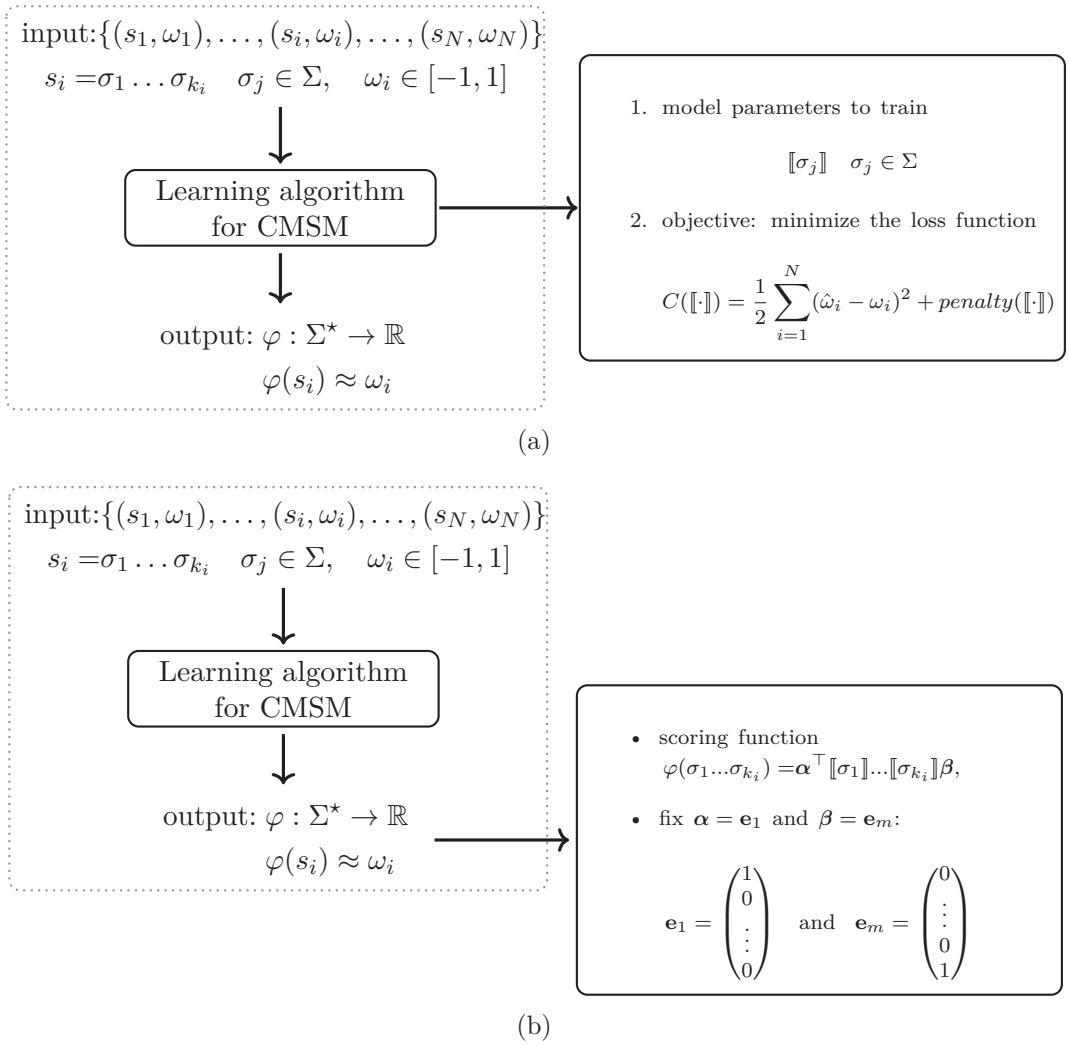


Figure 3.1: Supervised learning procedure for phrase scoring in sentiment analysis. The input to the learning algorithm is a training set of size N. Σ is the vocabulary extracted from training set.

reduces the number of model parameters for training. We extract all the words from phrases in the training dataset as our vocabulary Σ, creating for each a quadratic matrix of size $m \times m$. This provides us with the initial values of our model parameters.

We consider the batch gradient descent optimization method for training. The objective of the learning algorithm is to train the model parameters in order to minimize the loss function defined as the Summed Squared Error (SSE):

$$E([\![\cdot]\!]) = \frac{1}{2}\sum_{i=1}^{N}(\hat{\omega}_i - \omega_i)^2 \tag{3.1}$$

in which $\hat{\omega}_i$ is the score of phrase $s_i = \sigma_1 \ldots \sigma_{k_i}$ predicted by the model and ω_i is its target score from the training dataset. N is the size of the training set. $\hat{\omega}_i$ is computed

as follows:

$$\hat{\omega}_i = \mathbf{e}_1^\top [\![\sigma_1]\!][\![\sigma_2]\!] \dots [\![\sigma_{k_i}]\!]\mathbf{e}_m, \tag{3.2}$$

which is the product of word matrices $[\![\sigma_j]\!]$ in the same order as they appear in the phrase being mapped to a scalar using the mapping vectors.

To prevent from over-fitting[1] and ill-conditioned matrices in learning, a penalty term is added to the loss function. Therefore, our final loss function is defined as follows:

$$C([\![\cdot]\!]) = E([\![\cdot]\!]) + \mathit{penalty}([\![\cdot]\!]). \tag{3.3}$$

The penalty introduces additional information during the training. $L2$ regularization can be considered as the penalty term, defined as follows:

$$\| M_\sigma \|_2^2 = \lambda((\sum_{l,k} M_\sigma(l,k)^2)^{1/2})^2,$$

which is the squared 2-norm of a word matrix $M_\sigma = [\![\sigma]\!]$. $\lambda > 0$ is the regularization parameter that shows the strength of the regularization effect. In our learning algorithm, using the L2 regularization, the penalty term becomes as:

$$\mathit{penalty}([\![\cdot]\!]) = \frac{\lambda}{2} \sum_\sigma \| [\![\sigma]\!] \|_2^2,$$

where:

$$\| [\![\sigma]\!] \|_2^2 = \sum_{l,k} M_\sigma(l,k)^2.$$

We use L2 regularization because it is differentiable with respect to matrices. In gradient descent, at each iteration of training, the loss function is computed and the parameter values are updated towards the local optimum of the loss function. Here, the parameters to be updated are the word matrices. Therefore, we update each word matrix M_σ as follows:

$$M_\sigma = M_\sigma - \eta \cdot (\frac{\partial C([\![\cdot]\!])}{\partial M_\sigma}) = M_\sigma - \eta \cdot (\frac{\partial E([\![\cdot]\!])}{\partial M_\sigma} + \lambda M_\sigma), \tag{3.4}$$

where η is the step size towards the local minimum of the loss function, called learning rate. $\frac{\partial C([\![\cdot]\!])}{\partial M_\sigma}$ is the partial derivative of the loss function with respect to the word matrix M_σ. $\frac{\partial E([\![\cdot]\!])}{\partial M_\sigma}$ is computed as follows:

$$\frac{\partial E([\![\cdot]\!])}{\partial M_\sigma} = \sum_{i=1}^{N} \frac{\partial E([\![\cdot]\!])}{\partial \hat{\omega}_i} \times \frac{\partial \hat{\omega}_i}{\partial M_\sigma}, \tag{3.5}$$

and the derivative of the penalty term with respect to M_σ is λM_σ.

According to Petersen and Pedersen (2012, p. 10) the derivative of the predicted score $\hat{\omega}_i$ for a phrase $s_i = \sigma_1 \dots \sigma_{k_i}$ with respect to the j-th word matrix $M_{\sigma_j} = [\![\sigma_j]\!]$ is

[1] Over-fitting in machine learning happens when the model fits a dataset perfectly and is unable to generalize to other datasets.

computed as follows:

$$\frac{\partial \hat{\omega}_i}{\partial M_{\sigma_j}} = \frac{\partial(\boldsymbol{\alpha}^\top M_{\sigma_1} \dots M_{\sigma_j} \dots M_{\sigma_{k_i}} \boldsymbol{\beta})}{\partial M_{\sigma_j}} = (\boldsymbol{\alpha}^\top M_{\sigma_1} \dots M_{\sigma_{j-1}})^\top (M_{\sigma_{j+1}} \dots M_{\sigma_{k_i}} \boldsymbol{\beta})^\top. \tag{3.6}$$

If a word σ_j occurs several times in the phrase, then the partial derivative of the phrase with respect to M_{σ_j} is the sum of partial derivatives with respect to each occurrence of M_{σ_j}.

Iterative training of the word matrices continues until there is no improvement in decreasing the error (i.e., the loss function converges to its local minimum). We discuss it in detail in the experiments section.

The output of a supervised learning algorithm is an approximation to a target scoring function φ^* using the trained parameters. In our algorithm, as illustrated in Fig. 3.1(b), after training the word matrices, the output is a scoring function φ (an approximation to φ^*) that closely approximates the scalar value ω_i for each phrase $s_i = \sigma_1 \dots \sigma_{k_i}$ using the following computations:

$$\hat{\omega}_i = \varphi(\sigma_1 \dots \sigma_{k_i}) = \mathbf{e}_1^\top M_{s_i} \mathbf{e}_m \tag{3.7}$$

$$M_{s_i} = M_{\sigma_1} M_{\sigma_2} \dots M_{\sigma_{k_i}} = [\![\sigma_1]\!][\![\sigma_2]\!] \dots [\![\sigma_{k_i}]\!], \tag{3.8}$$

where $\hat{\omega}_i$ is an approximation to ω_i. The scoring function φ^* is defined in Definition 2.5 in Chapter 2. Note that the matrix representation of a phrase $s_i = \sigma_1 \dots \sigma_{k_i}$ is the matrix product of the trained word matrices in the corresponding order.

3.2.2 Gradual Gradient Descent-based Matrix-Space Models

The novelty of this variant of learning algorithm for CMSMs, which is called Gradual Gradient descent-based Matrix Space Models (Grad-GMSM), consists in a two-step learning procedure, where the result of the first step is used as initialization for the second step. In this approach, we (1) perform an "informed initialization" exploiting available scoring information for single words (unigrams), (2) apply a first learning step only on parts of the matrices using scored one- and two-word sequences from our training set (unigrams and bigrams), and (3) use the matrices obtained in the last step as initialization for training the full matrices on the whole training set.

Informed Initialization:

In this step, we first take all the words in the training dataset as our vocabulary, creating quadratic matrices of size $m \times m$ with entries from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. Then, we consider the unigrams $s_i = \sigma$ with associated score ω_i in the training set. We exploit the fact that for any matrix M, computing $\mathbf{e}_1^\top M \mathbf{e}_m$ extracts exactly the

entry of the first row, last column of M:

$$\hat{\omega}_i = \mathbf{e}_1^\top M \mathbf{e}_m = \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}^\top \begin{pmatrix} x_{1,1} \cdots x_{1,m} \\ \vdots \ \ddots \ \vdots \\ x_{m,1} \cdots x_{m,m} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} = x_{1,m}.$$

Hence, to minimize the loss function, we update this entry in every matrix M_σ that corresponds to a unigram $s_i = \sigma$ of a scored unigram (s_i, ω_i) in our training set by this value, in other words, we let

$$M_\sigma = \begin{pmatrix} \cdot & \cdots & \omega_i \\ \vdots & \ddots & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix}. \tag{3.9}$$

This way, we have initialized the word-to-matrix mapping such that it leads to perfect scores on all unigrams mentioned in the training set.

First Learning Step:

After initialization, we consider bigrams (two-word sequences). The predicted score $\hat{\omega}_i$ of a bigram $s_i = \sigma_1\sigma_2$ is computed as follows:

$$\begin{aligned} \hat{\omega}_i = \mathbf{e}_1^\top M_{\sigma_1} M_{\sigma_2} \mathbf{e}_m &= \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}^\top \begin{pmatrix} x_{1,1} \cdots x_{1,m} \\ \vdots \ \ddots \ \vdots \\ x_{m,1} \cdots x_{m,m} \end{pmatrix} \begin{pmatrix} y_{1,1} \cdots y_{1,m} \\ \vdots \ \ddots \ \vdots \\ y_{m,1} \cdots y_{m,m} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{1,m} \end{pmatrix}^\top \begin{pmatrix} y_{1,m} \\ \vdots \\ y_{m,m} \end{pmatrix} = \textstyle\sum_{j=1}^m x_{1,j} y_{j,m}. \end{aligned}$$

We observe that for bigrams, multiplying the first row of the first matrix (row vector) with the last column of the second matrix (column vector) yields the score of the bigrams. Hence, as far as the scoring of unigrams and bigrams is concerned, only the corresponding row and column vectors are relevant – thanks to our specific choice of the vectors $\boldsymbol{\alpha} = \mathbf{e}_1$ and $\boldsymbol{\beta} = \mathbf{e}_m$.

This observation justifies the next learning step: we use the unigrams and bigrams in the training set to learn optimal values for the relevant matrix entries only. We apply the batch gradient descent optimization method on the training set to minimize the loss function defined as the SSE with the penalty term:

$$C([\![\cdot]\!]) = E([\![\cdot]\!]) + \mathit{penalty}([\![\cdot]\!]) = \frac{1}{2}\sum_{i=1}^{D}(\hat{\omega}_i - \omega_i)^2 + \frac{\lambda}{2}\sum_{\sigma} \parallel [\![\sigma]\!] \parallel_2^2,$$

where D is the size of the subset of training set used in this step (i.e., all scored unigrams and bigrams). $\parallel [\![\sigma]\!] \parallel_2^2$ is the squared 2-norm of the word matrix σ, and λ is the regularization parameter. Word matrices are trained and updated in the same way as explained in Section 3.2.1.

Second Learning Step:

Using the entries obtained in the previous learning step for training part of the word matrices, we finally repeat the optimization process using the whole training set and optimizing all the matrix entries simultaneously in the same way as described in Section 3.2.1. The loss function is defined the same as in Equation 3.3, and matrices are updated in the same way as in Equation 3.4. The training procedure continues until there is no improvement in decreasing the error (i.e., the loss function converges to its local minimum).

3.2.3 Experiments and Discussion

As explained before, the sentiment analysis task is to determine the polarity (negative, neutral, positive) and intensity (weak, medium, extreme) of a piece of text, that is, to rate the sentiment of a single word or a sequence of words (a phrase). Real-valued scores can express the polarity and intensity in a continuous interval, also called fine-grained sentiment analysis.

In the following, we show the results of training CMSMs to capture the sentiment scores of compositional phrases using our proposed learning approach. We conduct several experiments with two different datasets and compare the results with prior works on learning CMSMs and VSMs for sentiment analysis.

Datasets:

We use the following datasets for our experiments:

- *MPQA (Multi-Perspective Question Answering) opinion corpus*[1]: This dataset contains newswire documents annotated with phrase-level polarity and intensity. We extracted the annotated verb and noun phrases from the documents, obtaining $9,501$ phrases. In line with Yessenalina and Cardie (2011) and Irsoy and Cardie (2015), we removed phrases with low intensity. The levels of polarities and intensities, their translation to numerical values, and their occurrence frequency in the dataset are as per Table 3.1.

Polarity	**Intensity**	**Score**	**Frequency**
negative	high, extreme	-1.0	1581
negative	medium	-0.5	1940
neutral	medium, high, extreme	0.0	4475
positive	medium	0.5	1151
positive	high, extreme	1.0	354

Table 3.1: Polarity and intensity categories in the MPQA dataset, their translation to sentiment scores, and their occurrence frequency.

- *SCL-OPP (Sentiment Composition Lexicon with Opposing Polarity Phrases)*[2]: This dataset consists of 602 unigrams, 311 bigrams, and 265 trigrams (three-word

[1] http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/
[2] http://www.saifmohammad.com/WebPages/SCL.html

sequences) that have been taken from a corpus of tweets, and annotated with real-valued sentiment scores in the interval $[-1, +1]$ with human judgments (Kiritchenko and Mohammad, 2016b). Each phrase in the dataset contains at least one negative word and one positive word. For instance, the phrase "happy tears" carries a positive sentiment score, while the unigrams "tear" and "happy" carry negative and positive sentiment scores, respectively. The dataset contains different noun and verb phrases. The frequency of polarities are as per Table 3.2.

Polarity	**Frequency**
negative	647
neutral	12
positive	519

Table 3.2: Phrase polarities and their occurrence frequencies.

Experiments on MPQA:

The purpose of this experiment is to evaluate the performance of CMSMs in predicting the sentiment value of phrases with different lengths. In this experiment, we compare the results of our proposed approach with two previous methods for learning CMSMs introduced by Yessenalina and Cardie (2011) and Irsoy and Cardie (2015). We also study the impact of different types of matrix initialization as well as the number of dimensions on the CMSM performance.

K-fold cross-validation (Mosteller and Tukey, 1968) is an approach that randomly splits the whole dataset into K groups (folds) of approximately equal size. The $K-1$ folds are used for training the model and the remaining fold is kept for testing the model after training is done. This way, we train a model K times. At each time, a distinct fold is considered as the test set, and the average performance of the trained model in K times is the overall performance of the model. In our experiments, we apply a ten-fold cross-validation process on the training dataset as follows: eight folds are used as the training set, one fold as the validation set, and one fold as the test set. We apply the ten-fold cross-validation to both learning steps. The number of iterations in the first and second learning step of Grad-GMSM is set to $T = 400$ each. The same number of iterations is set in GMSM. In each step, we train the model parameters on the training set and at each iteration of training, we evaluate the performance of the current model on validation set using the Mean Mean Absolute Error (NLP) defined as follows:

$$L = \frac{1}{n} \sum_{i=1}^{n} |\hat{\omega}_i - \omega_i|,$$

where n is the size of the validation set. Note that validation set is not used for training the model parameters. Finally, a model with the minimum MAE on validation set is taken as the final model (i.e., the parameter values that result in minimum MAE on validation set are considered as the optimal values). Since we have two learning steps, the optimal parameter values from the first step are considered as the initial values for

the second learning step. We then record the MAE of the final model for the test set after training in the second step. The learning rates η of the first and second learning steps of Grad-GMSM were adapted experimentally and set to 0.001 and 0.01, respectively. The learning rate of GMSM is also set to $\eta = 0.01$.

Word matrices are initialized in two ways: (1) with a normal distribution $\mathcal{N}(0, 0.01)$ (2) with an identity matrix plus noise drawn from a normal distribution $\mathcal{N}(0, 0.01)$ as it is also suggested in previous works (Socher et al., 2012; Maillard and Clark, 2015). The results of the Grad-GMSM approach with both initializations and the GMSM approach with the former initialization are reported. We call our Grad-GMSM with the latter initialization Grad-GMSM+IdentityInit.

The dimension of matrices is set to $m = 3$ in order to be able to compare our approach with previous methods for learning CMSMs introduced by Yessenalina and Cardie (2011), called Matrix-space OLogReg+BowInit, and Irsoy and Cardie (2015), called multiplicative RNN (mRNN). Yessenalina and Cardie (2011) proposed the first supervised learning technique for CMSMs in sentiment analysis after Rudolph and Giesbrecht (2010) introduced the model. Also inspired by CMSMs, Irsoy and Cardie (2015) proposed a Multiplicative RNN to train the CMSMs for sentiment analysis.

Yessenalina and Cardie (2011) propose a model to predict an ordinal sentiment score (e.g., label 0 for highly negative sentiment, 1 for medium negative, 2 for neutral, and so on) for a given phrase. The model learns an interval for each sentiment label. Therefore, the model parameters to optimize are the word matrices as well as a set of threshold values, which indicate the intervals for sentiment labels. The word matrices are initialized in two ways: random initialization using the normal distribution, and BOWs initialization. In the latter case, first a Bag-of-Words Ordered Logistic Regression (BOW-OLogReg) model is trained on the same dataset in which each word in the BOWs model learns a scalar weight using ordered logistic regression. Then, a specific element of matrices is initialized with the learned weights from BOW-OLogReg. They apply Ordered Logistic Regression (OLogReg) to train word matrices and optimize the threshold values by maximizing the probability of predicting the sentiment interval of given phrases in the dataset or minimizing the negative log of the probability. To avoid ill-conditioned matrices, they add a projection step to matrices after each training iteration by shrinking all singular values of matrices close to one. The trained model with random initialization is called Matrix-space OLogReg+RandInit and the one with BOW initialization is called Matrix-space OLogReg+BowInit. They show outperformance of the latter model compared with the random initialization of the matrix-space model.

In multiplicative RNN proposed by Irsoy and Cardie (2015), a multiplicative interaction between the input vector and the previous hidden layer in a RNN is introduced using a shared third-order tensor $\mathbf{T} \in \mathbb{R}^{d \times m \times d}$. At each time step, the input word vector $\mathbf{v} \in \mathbb{R}^m$ is multiplied with the weight tensor $\mathbf{T}$, which results in a matrix M of size $d \times d$. Then the resulting matrix is multiplied with the previous hidden layer $\mathbf{h}_{t-1}$ to finally obtain the current hidden layer at time step t. Therefore, if the current hidden layer of a RNN is defined as follows:

$$\mathbf{h}_t = g(\mathbf{v}_t U + \mathbf{h}_{t-1} W + \mathbf{b}),$$

then the Multiplicative RNN computes the current hidden layer as follows:

$$\mathbf{h}_t = g(\mathbf{v}_t U + \mathbf{h}_{t-1} W + \mathbf{v}_t^\top \mathbf{T} \mathbf{h}_{t-1} + \mathbf{b}),$$

where in both equations U and W are the shared weight matrices for input-to-hidden and hidden-to-hidden layers, respectively, and $\mathbf{b}$ is the bias of the network. g is a nonlinear activation function, such as *tanh* function. $\mathbf{v}_t$ is the specific input word at time t, while $\mathbf{h}_t$ is the result of the current hidden layer. This means that the multiplicative relation between the input and the previous hidden layer is added to the current hidden layer computation. Thus, they incorporate multiplicative interaction in matrix-space models to RNNs using the term $\mathbf{v}_t^\top \mathbf{T} \mathbf{h}_{t-1}$. They use pre-trained word vectors of dimension $m = 300$ from word2vec (Mikolov et al., 2013b) as the input to their network.

Table 3.3 compares the result of our approach with the explained Yessenalina and Cardie (2011)'s approach and Irsoy and Cardie (2015)'s approach in matrix-space. As we observe, Grad-GMSM+IdentityInit obtains a significantly lower MAE than previously proposed methods and our Grad-GMSM and GMSM approaches. Moreover, the GMSM-based models train lower number of parameters as they do not optimize the threshold values for sentiment classes intervals.

Method	**Ranking loss**
Matrix-space OLogReg+RandInit (Yessenalina and Cardie, 2011)	0.7417
Matrix-space OLogReg+BowInit (Yessenalina and Cardie, 2011)	0.6375
Multiplicative RNN (Irsoy and Cardie, 2015)	0.5147
GMSM	0.3645 ± 0.007
Grad-GMSM	0.3429 ± 0.013
Grad-GMSM + IdentityInit	$\mathbf{0.3086 \pm 0.009}$

Table 3.3: Performance of different methods for learning CMSMs on MPQA dataset in sentiment analysis. The average MAE in a ten-fold cross-validation is reported.

By comparing Grad-GMSM+IdentityInit with Grad-GMSM we also observe faster convergence of the former over the latter, since the lowest MAE of Grad-GMSM+IdentityInit is obtained after 114.55 iterations on average. In Grad-GMSM, the lowest MAE happens on average after 161.85 iterations. GMSM cannot converge to its best model in T iterations, and we reported the results after T iterations.

Table 3.4 shows the sentiment scores of some example phrases obtained by our method and Matrix-space OLogReg+BowInit method (Yessenalina and Cardie, 2011). As shown in the table, the two methods' results coincide regarding the order of basic phrases: the score of *"very good"* is greater than *"good"* (and both are positive) and the score of *"very bad"* is less than *"bad"* (and both are negative). Also, *"not good"* is characterized as negative by both methods. On the other hand, there are significant differences between the two methods. For example, our method characterizes the phrase *"not bad"* as mildly positive while Matrix-space OLogReg+BowInit associates a negative score to it, the same discrepancy occurs for *"not very bad"*. Intuitively, we tend to agree more with our

method's verdict on these phrases.

Phrase	Grad-GMSM +IdentityInit	Matrix-space OLogReg+BowInit
good	0.64	2.81
very good	0.84	3.53
not good	−0.43	−0.16
not very good	−0.23	0.66
bad	−0.69	−1.67
very bad	−0.81	−2.01
not bad	0.32	−0.54
not very bad	0.21	−1.36

Table 3.4: Sample phrases with average sentiment scores obtained from different methods.

In general, our findings confirm those of Yessenalina and Cardie (2011): *"very"* seems to intensify the sentiment score of the subsequent phrase, while the *"not"* operator not only flips the sentiment of the phrase syntactically following it but also gradually dampens the sentiment of the phrases. In contrast, the scores of phrases starting with *"not very"* defy the assumption that the described effects of these operators can be combined in a straightforward way. Adverbs and negators play an important role in determining the sentiment scores of phrases.

Fig. 3.2 provides a more comprehensive selection of phrases and their associated scores by our method Grad-GMSM+IdentityInit. We obtained the range of sentiment scores by taking the minimum and maximum values predicted in the ten-fold cross-validation. Average values are computed by taking the average of predicted scores in the ten-fold cross-validation. We obtained an average of $\varphi(\textit{very very good}) = 0.98$, which is greater than *"very good"*, and $\varphi(\textit{very very bad}) = -0.95$ less than *"very bad"*. Therefore, we can also consider *"very very"* as an intensifier operator. Moreover, we observe that the average score of $\varphi(\textit{not really good}) = -0.34$ is not equal to the average score of $\varphi(\textit{really not good}) = -0.58$, which demonstrates that the matrix-based compositionality operation shows sensitivity to word orders, arguably reflecting the meaning of phrases better than any commutative operation could.

Although the sentiment scores of the training dataset consist of only the values of Table 3.1, the training of the model is done in a way that sentiment scores for phrases with more extreme intensity might yield real values greater than 1 or lower than −1, since we do not constrain the sentiment scores to $[-1, +1]$. Moreover, in our experiments we observed that, as opposed to Matrix-space OLogReg+BowInit, no extra precautions were needed to avoid ill-conditioned matrices or abrupt changes in the scores while training.

To observe the effect of a higher number of dimensions on our method, we repeated the experiments for Grad-GMSM+IdentityInit with $m = 50$, and observed a MAE of $e = 0.3092$ (i.e., virtually the same as for $m = 3$), and almost similar values for the number of iterations $T = 122$. The results confirmed the observation of Yessenalina and Cardie (2011), that increasing the number of dimensions does not significantly improve

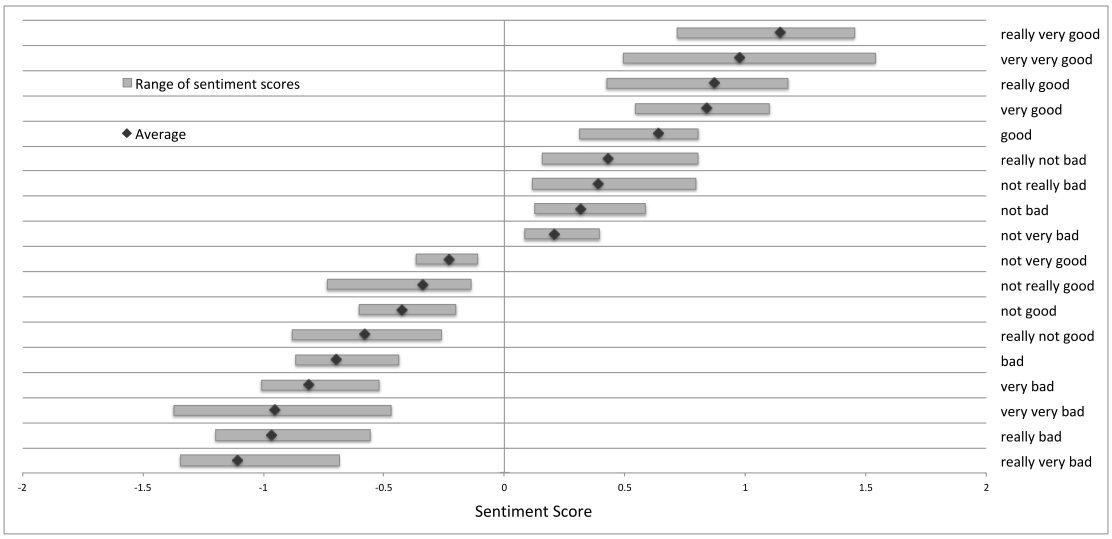


Figure 3.2: Sentiment scores for sample phrases obtained by our method (trained on MPQA dataset). Range of sentiment scores are obtained by taking the minimum and maximum values of the predicted scores in the ten-fold cross-validation. Average values are computed by taking the average of predicted scores in the ten-fold cross-validation.

the prediction quality of the obtained model.

Experiments on SCL-OPP:

The purpose of this experiment is to investigate CMSMs in predicting the sentiment composition of phrases that contain words with opposing polarities. The sentiment value of words (unigrams) is given for training the CMSM. In the first part of the experiments, we compare our results with the results of a supervised learning technique in the vector space taken from the work by Kiritchenko and Mohammad (2016b). In the second part, we explore different choices of dimensionality in learning CMSMs.

For the first experiment, we apply a ten-fold cross-validation process on the training dataset as follows: eight folds are used as training set, one fold as validation set and one fold as test set. We set the number of iterations to $T = 500$. We train the model parameters on the training set and at each iteration of training, we evaluate the performance of the current model on validation set using the MAE defined as follows:

$$L = \frac{1}{n} \sum_{i=1}^{n} |\hat{\omega}_i - \omega_i|,$$

where n is the size of the validation set. Finally, a model with the minimum MAE on validation set is taken as the final model; that is, the parameter values that result in minimum MAE on validation set are considered as the optimal values. We then record the Pearson correlation coefficient r for the test set, which measures the linear correlation between the predicted and the target sentiment value of phrases. The Pearson coefficient

value ranges from -1 to 1 with higher values showing more correlation between the predicted and target values and lower values showing they are more inversely correlated. The Pearson value r of pairs of predicted and target values $(\omega_i, \hat{\omega}_i)$ is computed as follows:

$$r = \frac{\sum_i (\omega_i - \bar{\omega})(\hat{\omega}_i - \bar{\hat{\omega}})}{\sqrt{\sum_i (\omega_i - \bar{\omega})^2}\sqrt{\sum_i (\hat{\omega}_i - \bar{\hat{\omega}})^2}} \quad \text{for} \quad 1 \leq i \leq n,$$

where n is the size of the test set, and $\bar{\omega}$ and $\bar{\hat{\omega}}$ are the mean values of the target and predicted values of the test set. We repeat the ten-fold cross-validation ten times and average the results.

The dataset contains unigrams, bigrams, and trigrams. Kiritchenko and Mohammad (2016b) report the results of training bigrams and trigrams separately using VSMs. To compare with Kiritchenko and Mohammad (2016b), we report the results for training only trigrams in the dataset since training bigrams does not exploit matrix properties in our method. Therefore, we only need to train the full matrices using the second learning step in the Grad-GMSM method.

Word matrices are initialized with an identity matrix plus noise drawn from a normal distribution $\mathcal{N}(0, 0.01)$. Then, we use the sentiment value of unigrams to initialize the corresponding element in the word matrices as our "informed initialization" explained in Section 3.2.2.

We set the dimension of matrices to $m = 200$ in line with Kiritchenko and Mohammad (2016b) as well as $m = 5$. The learning rate η in the second learning step of Grad-GMSM is set to 0.017 and 0.001 for dimensions 200 and 5, respectively, which are chosen as the optimal values via empirical experiments. We call the model GMSM-IdentityInit as the second learning step of Grad-GMSM is similar to GMSM.

The results of our method are compared with the results obtained by Kiritchenko and Mohammad (2016b), who study different patterns of sentiment composition in phrases. They analyze the efficacy of baseline and a supervised learning method in VSMs on the SCL-OPP dataset and the effect of different features used for training. Table 3.5 shows the results of different methods for training the trigrams. As baselines, Kiritchenko and Mohammad (2016b) consider the sentiment score of the last unigram of the phrase (Row 1), POS rule[1] (Row 2), and the sentiment score of the most polar unigram of the phrase (Row 3) to predict the overall sentiment score of the phrase. As a supervised method, they apply Radial Basis Function kernel-based Support-Vector Regression (RBF-SVR). In RBF-SVR the following set of features are evaluated on predicting real-valued sentiment scores: all unigrams (uni), sentiment score of unigrams (sent. score), POS tags (POS), concatenation of unigram embeddings (emb(conc)) to obtain phrase embedding, average of unigram embeddings (emb(avg)), and maximal vector of the unigram embeddings (emb(max)). The word embeddings are obtained from word2vec (Mikolov et al., 2013a). Row 8 considers the following features, which give the best results: uni, sent. score, POS, and emb(conc). Comparing the results of Row 7 and 8 show that concatenation of unigram embeddings as the composition operation outperforms average of unigram

[1]The POS rule assigns to the phrase the sentiment score of the words in the following priority order: adjectives, verbs, the most polar word. Therefore, if the phrase has adjectives, assign the score of the last adjective to the phrase, otherwise, if it has verbs, assign the sentiment score of the last verb. If none of them exist in the phrase, assign the score of the most polar word to the phrase.

Row	Method	Pearson r
	Baselines	
1	Baseline last unigram	0.376
2	Baseline POS rule	0.515
3	Baseline most polar unigram	0.551
	Supervised methods	
4	RBF-SVR(POS, sent. score)	0.578
5	RBF-SVR(POS, sent. score, uni)	0.711
6	RBF-SVR(POS, emb(conc), uni)	0.744
7	RBF-SVR(POS, sent. score, emb(avg), emb(max))	0.710
8	RBF-SVR(POS, sent. score, uni, emb(conc))	**0.753**
9	GMSM + IdentityInit (m=5)	0.734
10	GMSM + IdentityInit (m=200)	**0.737**

Table 3.5: Performance comparison of different methods for learning CMSMs on SCL-OPP trigram phrases in sentiment analysis. Pearson value r is used as the performance measure. The average of ten repeated runs is reported for supervised learning methods.

embeddings (emb(ave)) and maximal embeddings (emb(max)). They analyze the results for bigrams and trigrams separately; however, we only report the results on trigrams in Table 3.5.

Our approach does not use information extracted from other resources, such as pre-trained word embedding, nor POS tagging (i.e., we perform a light-weight training). As it is shown in Table 3.5, we observe an outperformance of our model on trigram phrases (Row10) over baseline methods and emb(ave) as the composition operation (Row 7). We also obtained similar results with lower dimensions (Row 9) which still outperforms the described methods. This introduces an advantage of CMSMs over VSMs.

Now, we combine bigrams and trigrams as our training set and apply our regular training procedure using Grad-GMSM on all phrases. We consider it important that the learned model generalizes well to phrases of variable length. Hence we find the training of one model per phrase length not conducive. Rather, we argue that training CMSM can and should be done independent of the length of phrases, by ultimately using the combination of different length phrases for training and testing, given the sentiment value of unigrams. Ten-fold cross-validation is used as before. Note that unigrams are only included for initialization of the training step, and we excluded them from the validation and test sets. This time we repeated the experiments on the Grad-GMSM+IdentityInit model with values of m (i.e., different numbers of dimensions). The number of iterations is set to $T = 500$. The learning rate η is set to 0.01 and 0.001 for the first and second learning steps, respectively. For each dimensionality, we take the average of five runs of ten-fold cross-validation. As shown in Table 3.6, the results improve only marginally when increasing m over several orders of magnitude. By increasing the dimensionality, the number of parameters to train grows, which might lead the model to get stuck in local optima. Also, the average number of required iterations remains essentially the

same, except for $m = 1$, which does not exploit the matrix properties. We see that – as opposed to VSMs – good performance can be achieved already with a very low number of dimensions.

Number of dimensions	**Ranking loss**	**Pearson** r	**Total number of iterations**
1	0.389	0.463	283.48
2	0.300	0.702	179.75
3	0.293	0.716	130.13
5	0.289	0.722	153.60
10	0.292	0.724	150.17
50	0.293	0.721	151.35
100	0.291	0.722	153.30
200	0.289	0.724	157.15
300	0.292	0.722	160.36

Table 3.6: Performance comparison for different dimensions in the SCL-OPP dataset using the Grad-GMSM+IdentityInit method.

3.2.4 Related Work

There is a lot of research interest in compositional sentiment analysis in NLP. Mohammad (2016) studies the task of sentiment analysis comprehensively. In this thesis, as explained in Chapter 2.3.4, we use sentiment analysis to refer to determining the polarity and intensity of a piece of text, which can also be a real-valued score. Yessenalina and Cardie (2011) focus on learning sentiment scores of short sequences based on supervised machine learning techniques. They apply ordered logistic regression method on compositional matrix-space models to acquire a matrix representation of words. The learning parameters in their method include the word matrices as well as a set of thresholds which indicate the intervals for sentiment classes since they convert the sentiment classes to ordinal labels. They argue that the learning problem for CMSMs is not a convex problem, so it must be trained carefully and specific attention has to be devoted to a good initialization to avoid getting stuck in local optima. We address this issue in our proposed learning method for CMSMs. Moreover, our learning method does not need to constrain the sentiment scores to certain intervals, and therefore, the number of parameters to learn is reduced to only word matrices.

Recent approaches have focused on training different types of neural networks for sentiment analysis, such as the work by Socher et al. (2013, 2012). Socher et al. (2012) propose a Recursive Neural Network in which the vector representations of phrases are trained using a tree structure. Each word and phrase is represented by a vector $\mathbf{v}$ and a matrix M. When two words in the tree are composed, the matrix of one is multiplied with the vector of the other word. Therefore, the composition function is parameterized by the words that participate in it. In their work, they predict the fine-grained sentiment scores of short phrases using the trained Recursive Neural Network. A problem with this model is that the number of parameters becomes very large as it needs to store a matrix

and a vector for each word and phrase in the tree together with the fully labeled parse tree. This means that the dataset must be preprocessed to generate the parse trees. In contrast, compositional matrix-space models do not rely on parse trees, and therefore, preprocessing of the dataset is not required. Each word is represented only with matrices where the compositional function is the standard matrix multiplication, which replaces the recursive computations with a sequential computation. Socher et al. (2013) address the issue of the high number of parameters by introducing a Recursive Neural Tensor Network in which a global tensor-based composition function is defined. In this model, a tensor layer is added to their standard Recursive Neural Network, where the vectors of two words are multiplied with a shared third-order tensor in this layer and then passed to the standard layer of the network. The output is a composed vector of the words, which is then composed with the next word in the same way. The model is evaluated on both fine-grained and binary (only positive and negative) sentiment classification of phrases and sentences.

Irsoy and Cardie (2015) propose a multiplicative RNN as a compositional model with multiplicative operation inspired by CMSMs, and evaluate the model on fine-grained sentiment analysis. They show that their proposed architecture is more generic than CMSMs and outperforms additive neural networks in sentiment analysis. They use pre-trained word vectors of dimension 300 from word2vec (Mikolov et al., 2013b) as the input to the network, and explore different sizes of the shared third-order tensor. The results on the task of sentiment analysis are compared with the work by Yessenalina and Cardie (2011). We also compare the results of our method for learning CMSMs with this approach using the same task and show that our method performs better while using fewer dimensions.

Kiritchenko and Mohammad (2016b) create a dataset of unigrams, bigrams, and trigrams, which contains specific phrases with at least one negative and one positive word. They analyze the performance of a Support-Vector Regression in vector space considering different features. We show that our trained model can predict the sentiment score of such phrases with a lower number of dimensions. In the work by Kiritchenko and Mohammad (2016c), they create a sentiment composition lexicon for phrases containing negators and adverbs with their associated sentiment scores and study the effect of modifiers on the overall sentiment of phrases. Adverbs and negators play an important role in determining the sentiment scores of phrases.

There are several deep neural network models on the task of compositional sentiment analysis, such as Hong and Fang (2015) who apply Long Short-Term Memory (LSTMs) and deep recursive neural networks, and Wang et al. (2016) who combine CNNs and RNNs leading to a significant improvement in sentiment analysis of short phrases. These techniques do not focus on training word representations that can be readily composed and, thus, are not comparable directly to our proposed method.

3.3 Learning Approach to CMSMs in Compositionality Detection

Multi-Word Expressions (MWEs) are short compounds with two or more words showing a range of semantic compositionality. The semantics of a compositional MWE, such

as *graduate student*, is a function of the semantics of its components and can be understood from the meaning of its components. Therefore, compositional compounds are decomposable. The semantics of a non-compositional compound cannot be understood from the semantics of its parts, such as *couch potato* in which the meaning of the compound cannot be obtained from the meaning of *couch* and *potato* (Baldwin and Kim, 2010).

Detecting the compositionality of MWEs is especially important in meaning-related NLP applications, such as phrase-based statistical machine translation (Kordoni and Simova, 2014; Enache et al., 2014; Weller et al., 2014) and word sense disambiguation (McCarthy et al., 2003; Finlayson and Kulkarni, 2011). Therefore, approaches for detecting the (non-)compositionality of compounds in NLP are needed.

One approach is to train compositional representation models to capture the compositional compound representation from its components' representations. In such methods, target representation of compositional compounds must be provided from standard resources for training and testing. Thus, if the trained model cannot closely approximate the target representation of a compound, it recognizes the compound as non-compositional. This way (non-)compositionality of MWEs can be detected (Farahmand et al., 2015). Gold standard evaluation datasets for the compositionality detection task have been introduced to evaluate the performance of the trained models using this approach. These datasets consist of pairs of compounds and their compositionality degrees. The degrees can, for example, take an integer value between 1 to 5 with 1 as fully compositional and 5 as fully non-compositional. They can also take binary values, for example, 0 as non-compositional and 1 as compositional. Datasets are created with human judgments on the compositionality of compounds.

In the compositionality detection task, different compositional models can be evaluated on their ability to distinguish between the compositional and non-compositional MWEs.

In this section, we first propose a learning technique for CMSMs that is trained to produce compositional representations of compounds from their components' representations. Then, using the available gold standard evaluation datasets, we conduct experiments to investigate CMSMs on detecting compositionality of MWEs and compare with popular compositional models in vector space. Since the evaluation datasets are labeled with compositionality degrees, to predict these values, we compare the compound representation produced by the models with the target representation, and based on the closeness (proximity) of the two representations in the semantic space, the models compute a degree of compositionality. Closeness or proximity of two representations can be computed in two ways: Squared Euclidean (SE) norm and the cosine value. We conduct experiments for both measures. Details of computations are explained in Section 3.3.2 on experiments.

3.3.1 Learning Approach

Fig. 3.3 shows the learning procedure for CMSMs. The task is to learn a model that can compose the word representations to obtain the compositional compound representation.

The training dataset for the supervised learning algorithm is a set of pairs $(s_i, \mathbf{v}_i)$, where s_i is a compound and $\mathbf{v}_i$ is the vector associated with it. The length of the compounds is limited to two-word sequences (i.e., bigrams), as they are the most basic

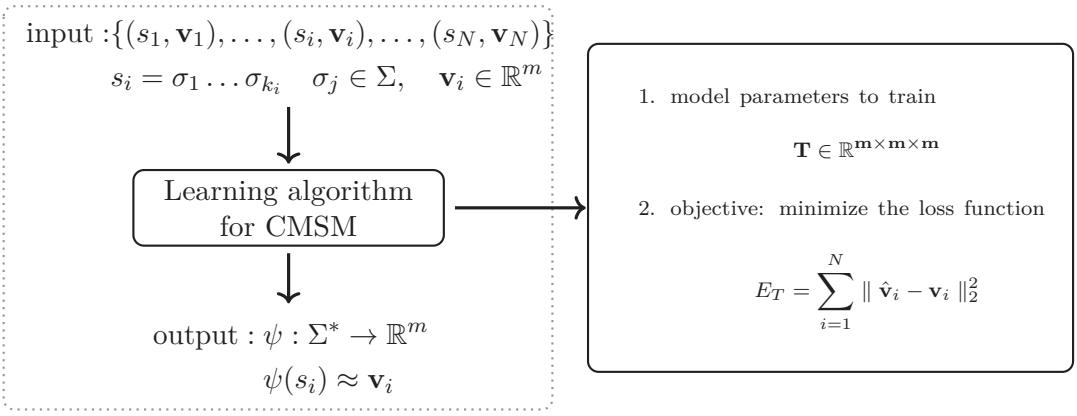


Figure 3.3: Supervised learning procedure for phrase scoring in the compositionality detection task. $\{s_i, \mathbf{v}_i\}_{i=1}^N$ is the training set of size N. Σ is the vocabulary extracted from training set.

compositional structures and also to respect the evaluation dataset standards. The compounds in the training dataset are frequent two-word sequences with a frequency threshold of 50 extracted from the English Wikipedia dump 2018[1] used as our corpus. After extracting the frequent compounds, the compound vectors and their constituent word vectors are obtained by training the vector embeddings of all words and compounds using word2vec (Mikolov et al., 2013a) and fastText (Bojanowski et al., 2017) on the corpus, separately. In these models, short phrases can also be considered as individual tokens, and the model is trained to extract a vector representation for those phrases as well as for words. It has been shown that these models capture the semantics of short phrases as well as words (Mikolov et al., 2013b). In the end, we have two training datasets; one training dataset consists of compounds with their vector representations extracted from the word2vec embeddings, and the other which includes the compounds with vector representations from the fastText embeddings. Now, we use the created training dataset for our supervised learning algorithm to train our model. We report the results of the experiments with the two training datasets separately.

In the learning algorithm, our model parameter is not the word matrices, but instead, we train a shared tensor $\mathbf{T} \in \mathbb{R}^{m \times m \times m}$, which then produces the word matrices. We explain why we use a tensor $\mathbf{T}$ as our model parameter to train. We exploit the fact that for every word σ a vector $\mathbf{v}_\sigma$ is readily available (from word2vec or fastText embeddings) and, as many investigations in distributional semantics have shown, the semantic closeness between two words σ and σ' correlates with smaller distances between their vectors $\mathbf{v}_\sigma$ and $\mathbf{v}'_\sigma$. We want to preserve that information by making sure that closeness (proximity) of $\mathbf{v}_\sigma$ and $\mathbf{v}'_\sigma$ entails proximity of their associated matrices $[\![\sigma]\!]$ and $[\![\sigma']\!]$. To this end, we let

$$[\![\sigma]\!] = \mathbf{v}_\sigma \mathbf{T}, \tag{3.10}$$

[1] https://dumps.wikimedia.org/

where $\mathbf{T} \in \mathbb{R}^{m \times m \times m}$ is a third-order tensor and $\mathbf{v}_\sigma \mathbf{T}$ yields the matrix M_σ with

$$M_\sigma(i,j) = \sum_{k=1}^{m} \mathbf{v}_\sigma(k)\mathbf{T}(k,i,j).$$

This way, instead of training the word matrices directly, we only train the shared tensor $\mathbf{T}$ as the model parameter. Besides having the above-mentioned effect, the usage of a shared tensor significantly reduces the number of model parameters to be trained. Moreover, as the size of the vocabulary increases, the number of parameters to train remains fixed. Using a shared tensor in this way is inspired by Irsoy and Cardie (2015).

The objective of the learning algorithm is to train $\mathbf{T}$ to minimize the loss function E_T, defined in Equation 3.11, during the training procedure.

$$E_T = \sum_{i=1}^{N} ||\hat{\mathbf{v}}_i - \mathbf{v}_i||_2^2, \tag{3.11}$$

where $\| \cdot \|_2^2$ is the square of the 2-norm (Euclidean norm) of the vector difference, and N is the size of the training dataset. $\| \cdot \|_2$ computes the Euclidean norm of a vector $\mathbf{v}$ of size m as follows:

$$\| \mathbf{v} \|_2 = \sqrt{\mathbf{v}(1)^2 + \cdots + \mathbf{v}(m)^2}.$$

The vector $\hat{\mathbf{v}}_i$ is the approximated vector representation for s_i computed by our model as follows:

$$\hat{\mathbf{v}}_i = \boldsymbol{\alpha}^\top [\![\sigma_1]\!] \ldots [\![\sigma_{k_i}]\!] \tag{3.12}$$

in which word matrices are obtained from Equation 3.10. Note that $[\![\sigma_1]\!] \ldots [\![\sigma_k]\!]$ is the compositional matrix representation of the compound, but since the training dataset is only available in vector space, we use a global mapping vector $\boldsymbol{\alpha}$ to map the final matrix to a vector representation. $\mathbf{T}$ must be trained to produce suitable word matrices which consequently are mapped to the vector representation of the corresponding compound using Equation 3.12. Gradient descent optimizer is used to train the tensor $\mathbf{T}$ as a regression task.

The output is to learn a composition function ψ which predicts the vector $\hat{\mathbf{v}}_i$ for a compound $s_i = \sigma_1 \ldots \sigma_{k_i}$ through the multiplication of its word matrices $[\![\sigma_j]\!] \in \mathbb{R}^{m \times m}$, obtained from the trained tensor $\mathbf{T}$, and the projection of the resulting matrix to the vector space $\mathbb{R}^m$ using the global mapping vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ as follows:

$$\hat{\mathbf{v}}_i = \psi(s_i) = \boldsymbol{\alpha}^\top [\![\sigma_1]\!] \ldots [\![\sigma_{k_i}]\!].$$

Finally, the CMSM learns to compose the word matrix representations and predicts the vector representation of the compound by mapping the final compound matrix to the vector space.

The model detects non-compositionality with the assumption that most of the training compounds extracted from the corpus are compositional and a composition function ψ can be learned for them. This entails that the compounds for which a composition function ψ cannot be learned with a low error are non-compositional.

After learning CMSM as a compositional model, we evaluate the performance of the

model on the compositionality detection task using a procedure, illustrated in Fig. 3.4. The evaluation dataset consists of compounds s_i and their corresponding compositionality degrees r_i. For evaluation purposes, we apply the learned function ψ to the compounds in the dataset to predict the vector representations of the compounds. The predicted vector $\hat{\mathbf{v}}_i$ is compared with the target vector representation $\mathbf{v}_i$ of the compound by computing the SE norm between the two representations, as shown in Equation 3.13 (or the cosine value, as shown in Equation 3.14):

$$SE(\hat{\mathbf{v}}_i, \mathbf{v}_i) = \| \hat{\mathbf{v}}_i - \mathbf{v}_i \|_2^2, \tag{3.13}$$

$$\cos(\mathbf{v}_i, \hat{\mathbf{v}}_i) = \frac{\hat{\mathbf{v}}_i . \mathbf{v}_i}{\| \hat{\mathbf{v}}_i \| \| \mathbf{v}_i \|}, \tag{3.14}$$

where $\| \cdot \|_2$ and $\| \cdot \|$ compute the Euclidean norm of a vector $\mathbf{v}$ of size m as follows:

$$\| \mathbf{v} \|_2 = \| \mathbf{v} \| = \sqrt{\mathbf{v}(1)^2 + \cdots + \mathbf{v}(m)^2}.$$

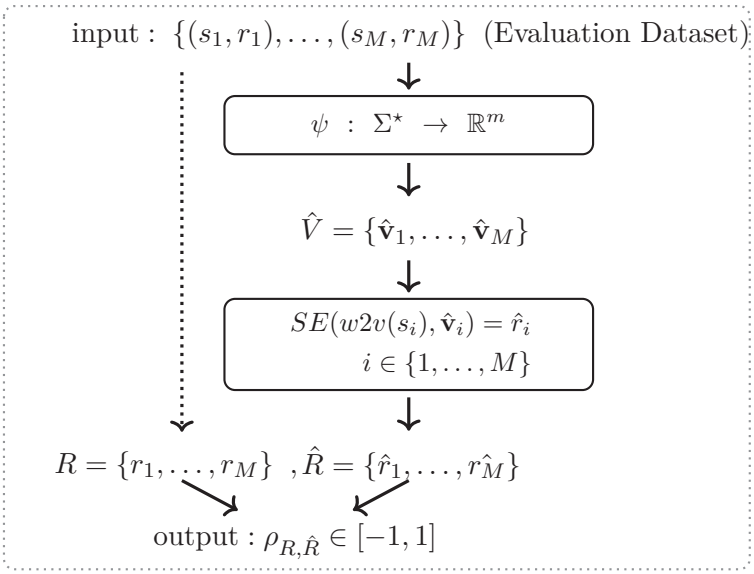


Figure 3.4: Evaluation procedure of the learned model on the compositionality detection task using the gold standard evaluation dataset of size M. Pearson correlation value between the predicted and the target compositionality degree of compounds is computed. $w2v$ produces the target representation of the compound from the word2vec and fastText methods. Note that evaluation results are reported with both SE and the cosine value.

Note that the target representations $\mathbf{v}_i$ of compounds s_i in the evaluation dataset are obtained from the trained word2vec and fastText embeddings on the English Wikipedia dump corpus.

The obtained values from SE (or the cosine value) are compared with the target compositionality degrees of the compounds in the dataset. If the SE value is high, the

model recognizes the compound as less compositional or non-compositional. If the cosine value is high, the model recognizes the compound as more compositional. To investigate the performance of the model on compositionality detection, the Pearson correlation between the target and obtained values of all compounds are computed. A higher Pearson value indicates better performance of the model on detecting compositionality.

3.3.2 Experiments and Discussion

We evaluate the performance of several compositional models on predicting the degree of MWEs' compositionality and compare them with the performance of CMSMs.

Compositional Models:

Each model defines a composition function f over the constituent word vectors to predict the compound vector. Given two words w_i and w_j with associated vectors $\mathbf{v}_i \in \mathbb{R}^m$ and $\mathbf{v}_j \in \mathbb{R}^m$, we evaluate the following models:

- *Weighted additive model*: In this model, the predicted compound vector representation $\mathbf{v}_{ij}$ is the weighted sum of the constituent word vectors (Mitchell and Lapata, 2008; Reddy et al., 2011).

$$\hat{\mathbf{v}}_{ij} = f(\mathbf{v}_i, \mathbf{v}_j) = \lambda_1 \mathbf{v}_i + \lambda_2 \mathbf{v}_j \quad \lambda_1 + \lambda_2 = 1,$$

  where λ_1 and λ_2 are the weight coefficients.

- *Multiplicative model*: In this model, the predicted compound vector representation $\mathbf{v}_{ij}$ is the element-wise product of the constituent word vectors (Mitchell and Lapata, 2008; Reddy et al., 2011).

$$\hat{\mathbf{v}}_{ij} = f(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i \odot \mathbf{v}_j.$$

- *Polynomial regression model*: In this model, to predict the compound representation $\mathbf{v}_{ij}$, the constituent word vectors are stacked together $[\mathbf{v}_i, \mathbf{v}_j]$ and a polynomial function ψ is applied to them (Yazdani et al., 2015):

$$\hat{\mathbf{v}}_{ij} = f(\mathbf{v}_i, \mathbf{v}_j) = \psi([\mathbf{v}_i, \mathbf{v}_j])\theta,$$

  where θ is the weight matrix to be trained, and ψ is the quadratic transformation in our experiment, defined as follows, which is applied to the stacked vectors:

$$\psi(x_1, \ldots, x_{2m}) = (x_1^2, \ldots, x_{2m}^2, x_1x_2, \ldots, x_{2m-1}x_{2m}, x_1, \ldots, x_{2m}).$$

  Therefore, it models a nonlinear relationship between the compound representation and the constituent words' representations, and then θ is trained using linear regression.

- *Feedforward NN*: In this model, the constituent word vectors are stacked together as the input vector, and the input and output weight matrices are trained in order to predict the vector representation of the compound (Yazdani et al., 2015).

$$\hat{\mathbf{v}}_{ij} = f(\mathbf{v}_i, \mathbf{v}_j) = \sigma([\mathbf{v}_i, \mathbf{v}_j]W)V,$$

where W and V are the input-to-hidden and hidden-to-output layer weight matrices to be trained and σ is a nonlinear function, such as the *sigmoid* function. The size of the hidden layer h in the network is set to 300.

- *RNN*: In this model, as explained in Chapter 2.1.2, the input word vectors are fed into the network sequentially. The hidden state at time step t is computed as follows:

$$\mathbf{h}_t = g(\mathbf{v}_t U + \mathbf{h}_{t-1} W + \mathbf{b}),$$

  where g is an activation function, such as *tanh*, to introduce nonlinearity. The hidden state $\mathbf{h}_{t-1}$ from previous time step is combined with the current input $\mathbf{v}_t$ and a bias $\mathbf{b}$. The new hidden state $\mathbf{h}_t$ that we computed will then be fed back into the RNN cell together with the next input and this process continues until the last input feeds into the network.

  Inputs are the word vectors of the compounds in a sequence. The size of the hidden layer is set to 300. We only require the output of the last time step T in the sequence, and therefore we pass the last hidden layer $\mathbf{h}_T$ through a linear layer to generate the predicted compound vector representation:

$$\hat{\mathbf{v}}_{ij} = \mathbf{h}_T V + \mathbf{c},$$

  where V is the shared weight matrix of the linear layer.

- *CMSM*: This model has been introduced in Section 3.3.1.

For all models tested, the predicted compound vectors are compared with the target vector representation of the compounds through the closeness (proximity) measurements. Recall that the constituent word vectors and the target compound vectors in the training and evaluation datasets are obtained by training the vector embeddings of all words and compounds using word2vec (Mikolov et al., 2013b) and fastText (Bojanowski et al., 2017) on English Wikipedia dump 2018[1] used as our corpus. It has been shown that these models capture the semantics of short compositional phrases as well as words (Mikolov et al., 2013a). We report the results of word2vec and fastText separately.

Datasets:

Training Datasets: For some models (CMSM, Polynomial regression model, feed-forward NN, and RNN), we require to fit the composition function f using supervised learning to capture the compositional representation of the compounds. Therefore, as described in Section 3.3.1, we create a training dataset from frequent two-word compounds extracted from our corpus, Wikipedia dump 2018. For this purpose, we first extract frequent two-word sequences with a frequency threshold of 50 from our corpus and then we train the word and compound representations using word2vec and fastText. Recall that we create two training datasets: one dataset consists of compounds with associated target representations obtained from the fastText method, and the other includes two-word compounds with associated target representations obtained from word2vec. We limit our experiments to bigrams as they are the most basic compositional

[1] https://dumps.wikimedia.org/

structures and to respect the evaluation datasets standard. We assume the majority of compounds are compositional and train the compositional models on each training dataset separately. We split each training dataset to the training and development sets for experimental purposes.

Evaluation Datasets: Finally, we use two recent gold standard evaluation datasets which reflect the compositionality judgments of MWEs to evaluate all compositional models:

- *Farahmand15*[1] (Farahmand et al., 2015) provides $1,042$ English noun–noun compounds (bigrams) extracted from Wikipedia which were annotated with a non-compositionality degree between 0 (fully compositional) to 1 (fully non-compositional) using crowdsourcing. Each compound was annotated by four annotators for binary non-compositionality judgments, and the average of annotations was considered as the final score of the compound which is a value from $\{0, 0.25, 0.5, 0.75, 1\}$.
- *Reddy++*[2] (Ramisch et al., 2016; Reddy et al., 2011) provides 180 English noun–noun and adjective–noun compounds (bigrams) with real-valued compositionality degree ranging from 1 (fully non-compositional) to 5 (fully compositional) obtained from crowdsourcing and averaged over around ten to twenty annotators per compound.

The vector representation of bigrams in the evaluation datasets are obtained from word2vec and fastText for examining the learned compositional models.

Experimental Setting and Results:

To train the models with word2vec, the size of the training and development sets are $7,692$ and 854 compounds, respectively, and fixed for all models. The size of the training and development set for training the models with fastText embedding are $11,566$ and $1,156$ compounds, respectively, and fixed for all models. The batch size for training is set to $B = 10$. The learning rate is adapted experimentally for each model. We apply early stopping by computing the MSE value of the development set to prevent overfitting. If the absolute difference of development loss in two consecutive iterations is less than the threshold value $\epsilon = 10^{-5}$, we stop the training. The tensor $\mathbf{T}$ in the CMSM is initialized with a random Gaussian distribution $\mathcal{N}(0, 0.01)$. The size of all vectors is set to 300 in both experiments with word2vec and fastText.

Once the models are trained using each training dataset, we evaluate their performance on detecting the compositionality degrees of compounds in the evaluation datasets, Farahmand15 and Reddy++.

In the evaluation phase with word2vec embeddings, some compounds of the evaluation datasets do not exist in our trained embeddings on Wikipedia dump. Therefore, we remove those compounds from the evaluation datasets and consider 800 compounds from the Farahmand15 and 148 compounds from the Reddy++ dataset. In the evaluation with fastText, all compounds of the Farahmand15 and the Reddy++ datasets exist in

[1]https://github.com/meghdadFar/en_ncs_noncompositional_conventionalized
[2]http://pageperso.lif.univ-mrs.fr/ carlos.ramisch/?page=downloads/compounds

Performance measures	Cosine		SE	
Model \ Dataset	Reddy++	Farahmand15	Reddy++	Farahmand15
Additive	0.631	0.398	0.621	**0.393**
Multiplicative	0.218	0.055	0.225	0.057
Polynomial regression	0.699 ± 0.008	$\mathbf{0.404 \pm 0.005}$	0.698 ± 0.008	0.394 ± 0.005
Feedforward NN	0.658 ± 0.027	0.395 ± 0.016	0.642 ± 0.029	0.382 ± 0.018
RNN	0.688 ± 0.011	0.394 ± 0.006	0.687 ± 0.010	0.382 ± 0.006
CMSM	$\mathbf{0.710 \pm 0.012}$	0.401 ± 0.005	$\mathbf{0.700 \pm 0.011}$	0.389 ± 0.004

Table 3.7: Average Pearson value (with standard deviation for the trained models) for compositionality judgment on the two evaluation datasets. Two measures cosine and the SE norm are used to measure the performance of compositional models. word2vec embedding is used for the experiments.

our trained fastText embedding on Wikipedia dump and therefore, we test the models on the whole dataset.

To measure the closeness (proximity) between the predicted compound representations from the models and the target representations of compounds, we compute the cosine value as well as the SE norm between the two representations. We expect a low cosine value for non-compositional compounds as the composition functions cannot capture their representations. We also compute the SE between the predicted and the target vector representation of the compound being sensitive to small errors. We expect a high error value for non-compositional compounds as the composition functions cannot capture their representations (Yazdani et al., 2015). Then, we compute the Pearson correlation value r between the computed values and the compositionality judgments from the evaluation datasets. r ranges from -1 to 1 with higher values showing more correlation between the predicted and target values.

We report the average results over fifteen runs of training and evaluation. Table 3.7 and 3.8 show the average Pearson correlation coefficient r between the predicted and target degrees of each evaluation dataset for different supervised and unsupervised compositional models. Table 3.7 shows the results of the word2vec embeddings and Table 3.8 shows the results of the fastText embeddings. The performance measures of the models are shown in two ways as described before. First, if a method captures the compositional representation of the compounds, the cosine between the predicted and target representations has a higher value, otherwise, the cosine is a low value. Therefore, the "Cosine" column in both tables shows the result of Pearson correlation value between the computed cosine of the representations and the target degrees in the evaluation datasets which are normalized between -1 (non-compositional) and 1 (compositional) compounds. Second, if a method captures the compositional representation of the compounds, following Yazdani et al. (2015), the SE value between the predicted and target representation of a compositional compound must be low and close to 0, otherwise it is a high value. Therefore, the "SE" column in the tables shows the result of the correlation between the SE of the representations and target degrees in the evaluation datasets, which are normalized to 0 (fully compositional) and 1 (fully non-compositional). The tables demonstrate that the two measures provide very similar results.

Performance measures	**Cosine**		**SE**	
Model \ **Dataset**	**Reddy++**	**Farahmand15**	**Reddy++**	**Farahmand15**
Additive	0.355	**0.527**	0.348	**0.523**
Multiplicative	0.091	0.021	0.104	0.028
Polynomial regression	0.583 ± 0.011	0.521 ± 0.003	0.576 ± 0.011	0.5132 ± 0.003
Feedforward NN	0.583 ± 0.009	0.493 ± 0.004	0.586 ± 0.010	0.482 ± 0.005
RNN	0.565 ± 0.005	0.505 ± 0.003	0.557 ± 0.005	0.495 ± 0.003
CMSM	$\mathbf{0.617 \pm 0.009}$	0.513 ± 0.004	$\mathbf{0.605 \pm 0.009}$	0.503 ± 0.004

Table 3.8: Average Pearson value (with standard deviation for the trained models) for compositionality judgment on the two evaluation datasets. Two measures Cosine similarity and the SE norm (SE) are used to evaluate the performance of compositional models. fastText embedding is used for the experiments.

We report the best results of the additive model obtained by adapting λ_1 and λ_2 (ranging from 0 to 1 with a step size of 0.1) in these models. As we observe in both tables, the multiplicative model is not powerful enough to model compositionality. These results are in line with the results reported by Yazdani et al. (2015). The CMSM is trained to capture the compositionality better than other models in the Reddy++ dataset in both tables, which means that CMSM gives a higher SE value and lower cosine to non-compositional compounds.

The number of training iterations for each supervised compositional model to reach its optimum performance is shown in Table 3.9. As it is shown in the table, the CMSM converges to its best model in less training iterations on average. The different iteration numbers in the two experiments are due to the different learning rates adapted to obtain the best models on the word embeddings. In the Farahmand15 dataset, the additive model shows a competitive result in comparison to CMSM, while in the Reddy++ dataset, the CMSM outperforms the additive model considerably. We conjecture that this is because the Reddy++ is a dataset with much more fine-grained values, and CMSM tends to be more accurate in predicting the nuanced values than other models.

Model	**Average iterations in word2vec**	**Average iterations in fastText**
Multiple Regression	114	221
Neural Network	320	258
RNN	98	126
CMSM	124	169

Table 3.9: Average number of training iterations for each supervised compositional model trained using word2vec and fastText.

Moreover, Reddy++ contains adjective–noun and noun–noun compounds as opposed to Farahmand15, which contains only noun compounds. Therefore, we conclude that CMSM can predict the compositionality of adjective–noun compounds better than the studied compositional models. Various parameters, such as the training dataset and

vector embeddings, impact the performance of the models. Therefore, in our experiments, we used the same training data and vector embeddings for all models to obtain a more reliable indication regarding the relative performance of the models.

Based on these results, we can conclude that a CMSM can be trained to capture the semantic compositionality of compounds more efficiently than baseline VSMs. Moreover, CMSM is sensitive to syntactic properties such as the word order of the compound which affects the representation of compounds. Furthermore, multiplicative interaction shows a better performance in matrix space than in vector space. The results suggest that matrix multiplication should be considered instead of additive models in vector space as the composition operation for compositionality prediction in meaning-related NLP tasks.

Finally, we study the time cost required for training CMSMs in the studied datasets (SCL–OPP and MPQA) and with two different dimensionality (5 and 200). Note that we report the time cost for ten-fold cross validation. Results in Table 3.10 show the advantage of smaller dimensionality of CMSMs in faster convergence.

Model	**Time (MPQA)**	**Time (SCL–OPP)**
CMSM (m=5)	13	4
CMSM (m=200)	270	90

Table 3.10: Time cost for training CMSMs with different dimensionality and datasets. Time is reported in minutes.

3.3.3 Related Work

Reddy et al. (2011) study the performance of compositional models on compositionality detection of multi-word compounds. For this purpose, they provide a dataset of noun compounds with fine-grained compositionality scores as well as literality scores for constituent words based on human judgments. They analyze both constituent-based models and composition-function-based models regarding compositionality detection of the proposed compounds. In constituent based models, they study the relations between the contribution of constituent words and the judgments on compound compositionality. They argue if a word is used literally in a compound, most probably it shares common co-occurrences with the corresponding compound. Therefore, they evaluate different composition functions applied on constituent words and compute their similarity with the literality scores of phrases. In composition-function-based models, they evaluate weighted additive and multiplicative composition functions on their proposed dataset, and investigate the similarity between the composed word vector representations and the target compound vector representation. The results show that in both models, additive composition outperforms other functions. Biemann and Giesbrecht (2011a) aim at extracting non-compositional phrases using automatic distributional models that can determine the compositionality degree of a phrase. This degree indicates how much the compositionality assumption holds for the given phrase. For this purpose, they create a dataset of English and German phrases which attracted several models ranging from statistical association measures and word space models (Biemann and Giesbrecht, 2011b).

Salehi et al. (2015) explore compositionality detection of MWEs using constituent-based and composition-function-based approaches on three different VSMs, consisting of count-based models, word2vec, and multi-sense skip-gram model. In a similar work, Farahmand et al. (2015) proposed a larger dataset of multi-word expressions annotated with non-compositionality judgments. Yazdani et al. (2015) explore different compositional models ranging from simple to complex models such as neural networks for non-compositionality detection of MWEs. Representation of words are obtained from word2vec (Mikolov et al., 2013a), and the models are trained using compounds extracted from the Wikipedia dump, assuming that most compounds are compositional. Therefore, the trained models are expected to give a relatively high error to non-compositional compounds. They improve the accuracy of the models using latent compositionality annotation and show that this method improves the performance of nonlinear models significantly. Their results show that a polynomial regression model with a quadratic degree outperforms other models.

Cordeiro et al. (2019) and Cordeiro et al. (2016) explore the performance of various representation models (GloVe (Pennington et al., 2014), word2vec and PPMI-based models) regarding predicting semantic compositionality of noun compounds over previously proposed datasets. Vector addition is considered as the composition operation, and the performance of word embeddings are investigated using different parameter settings for training them. They also study the impact of corpus preprocessing on capturing compositionality with DSMs. Recent deep learning techniques also focus on modeling the compositionality of more complex texts without considering the compositionality of the smaller parts such as the work by Wu and Chi (2017) which is out of the scope of our study. None of the mentioned works, however, has investigated the performance of CMSMs in compositionality detection of short phrases on MWE datasets.

3.4 Conclusion

In this chapter, we studied the behavior of CMSMs on different aspects (e.g., dimensionality) experimentally. According to experimental investigations, CMSMs are a promising framework to model task-specific semantic compositionality, such as sentiment analysis and compositionality detection of short sequences. Matrix product as the composition operation outperforms vector composition operations in the compositional sentiment analysis task. It performs competitively to compositional VSMs in the compositionality detection task. Adverbs and negators in natural language play an influential role in determining the sentiment scores of phrases. The results in the sentiment analysis task showed that multiplicative interaction in CMSMs captures the effect of adverbs and negators on the sentiment score when composed with a phrase. Moreover, CMSMs do not rely on parse trees, and therefore, preprocessing of the texts in datasets is not required. Each word is represented only with matrices where the compositional function is the standard matrix multiplication, which replaces the recursive computations in parse trees with a sequential computation. CMSMs also showed a better performance in capturing more fine-grained degrees in the compositionality detection task and tend to be more accurate in predicting the nuanced values than other models. They can also predict the compositionality of adjective–noun compounds better than the studied compositional

models.

We have seen strong evidence that CMSMs are capable of embedding relevant information in considerably fewer dimensions compared with VSMs, which gives a clear advantage in terms of computational cost, storage, and convergence in learning. We are aware that experiments have been only done on short length sequences, and further investigation is needed for examining CMSMs on longer sequences, such as sentences. Matrix multiplication on long sequences can cause the final matrix to contain extremely small values. Therefore, when updating word matrices in the gradient descent algorithm using Equation 3.4, the derivative of the loss function with respect to a word matrix M_σ is a function of Equation 3.6. Small values from the result of the derivative may not update the word matrix values adequately. Therefore, mechanisms are needed to avoid this issue when training CMSMs on long sequences. Moreover, when CMSMs are trained on long sequences in a specific task, such as, sentiment analysis, not all words contain task-specific information. A method is needed to pay attention and give more weights to those words that carry the relevant information, for instance, sentiment-carrying words in sentiment analysis.

Overall, this chapter demonstrates that CMSMs compose attractive theoretical features and practical behavior, which strongly suggest CMSMs as a suitable model for semantic compositionality in NLP downstream applications.

A central question related to CMSMs is how the linearity limits their applicability. In Chapter 2.3, we studied that Rudolph and Giesbrecht (2010) justified CMSMs by showing that matrix multiplication can realize mental state transitions triggered by the sequence of input tokens. Using linear mappings to represent those functions introduces limitations that need to be further investigated. An obvious issue is the word sense disambiguation problem that cannot be modeled via linear mappings. Therefore, we need to equip the CMSMs with simple nonlinear functions to introduce non-associativity to the CMSMs and resolve word sense disambiguation problems in natural language. Thus, a line of further research is to generalize CMSMs from the linear approach, very much in line with the current trend in deep learning.

Chapter 4

Evaluating Semantic Composition Methods

Several approaches for evaluating semantic composition methods on their ability to capture the semantic representation of compositional terms[1] have been proposed. A common evaluation approach to word representation models is through the models' ability to rank pairs of words by their closeness in meaning. Closeness is a measure of how close two terms are in terms of their semantics (Mohammad, 2008). Two terms are semantically close if they share some meaning. (Mohammad, 2008). For instance, the two words *teacher* and *tutor* are closer in meaning than the two words *teacher* and *fish*. Sharing of meaning is defined based on the lexical–semantic relations (i.e., the semantic relations between the lexical items; Cruse, 1986). As explained in Chapter 2.4, closeness in meaning can be of two kinds: semantic similarity and semantic relatedness. Both similarity and relatedness can be used for evaluating word representation models. In terms of lexical–semantic relations, two terms are considered to be semantically similar if there is a hypernymy, (co-)hyponymy, synonymy, or antonymy relationship between them as these relations share common properties (Budanitsky and Hirst, 2001; Mohammad, 2008; Agirre et al., 2009). Two terms are considered semantically related if there is any lexical–semantic relation between them, classical or non-classical. As discussed in Chapter 2.4, semantic relatedness is the broader class subsuming semantic similarity (Budanitsky and Hirst, 2006), and psychological and neuro linguistic studies have demonstrated the importance of semantic relatedness (Hutchison, 2003; Huth et al., 2016). These studies show that the human brain stores information in a thematic manner (based on relatedness) rather than based on similarity (Hutchison, 2003; Huth et al., 2016). Moreover, Hill et al. (2015) suggest that relatedness judgments have broader use in studies of human semantic cognition. Another limitation of similarity is that it can be only defined between terms categorized as the same POS. In contrast, two terms can be related even if they represent different parts of speech (Zesch and Gurevych, 2010). Previous studies have also shown that the ability to assess semantic relatedness is central to the use and understanding of natural language (Hutchison, 2003; Santus et al., 2015; Huth et al., 2016). Therefore, the quantification of semantic relatedness is needed for such evaluations in NLP. For this, semantic relatedness datasets annotated with human judgments are created.

Existing datasets of semantic relatedness, such as that by Finkelstein et al. (2002), only focus on pairs of unigrams (i.e., in this case, single words). However, the concept of semantic relatedness applies more generally to any unit of text, such as phrases and sentences (sequence of words). Bigrams (i.e., in this case, two-word sequences), are especially important since they are the smallest unit formed by composing words. Even though there is a large body of work on how to represent the meanings of sentences (Le and Mikolov, 2014; Kiros et al., 2015; Lin et al., 2017), there is relatively little work on how best to compose the meanings of two words to represent the meaning of a bigram. Thus, it would be useful to have large semantic relatedness datasets involving bigrams for evaluating semantic composition. Existing datasets also suffer from shortcomings due to the annotation techniques employed.

In this chapter, we introduce a new dataset, called Bigram Semantic Relatedness Dataset (BiRD), for evaluating the semantic composition methods. The dataset includes

[1]Note that the expressions *term* and *lexical item* are used interchangeably throughout this chapter. Lexical items can be words, phrases, or sentences.

3,345 English term pairs AB–X with a real-valued semantic relatedness score r between the two terms. The first term AB is a bigram (A represents the first word in the bigram and B represents the second word), and the other term X in the pair is either another bigram or a unigram. Section 4.1 describes the procedure to create the dataset, which is summarized as follows:

1. We first selected a set of target bigrams AB. For each AB, we created several pairs of the form AB–X, where X is a unigram or bigram. As X's, we chose terms from a diverse set of natural language resources and from various types of lexical–semantic relations. (Details in Section 4.1.1.)
2. We used Best–Worst Scaling (BWS) annotation technique (Louviere, 1991; Cohen, 2003; Louviere et al., 2015; Kiritchenko and Mohammad, 2017) to obtain semantic relatedness by prompting four pairs at a time and asking annotators to mark the pair that is most related and the pair that is least related. Kiritchenko and Mohammad (2017) showed through empirical experiments that BWS produces more reliable and more discriminating scores than those obtained using rating scales technique.
   Once the annotations were complete, we obtained real-valued scores of semantic relatedness for each pair using simple arithmetic on the counts of how often an item is chosen as best and worst. (Details in Section 4.1.2.)
3. To evaluate the quality of BiRD, we determined the consistency of the BWS annotations. A commonly used approach to determine consistency in dimensional annotations is to calculate Split-Half Reliability (SHR; Cronbach, 1951). We showed that our semantic relatedness annotations have a high reliability; that is, if the annotations were repeated, similar scores would be obtained. (Details in Section 4.1.3.)

After the creation of BiRD, we present analyses of the dataset in Section 4.2 to obtain insights into the relatedness scores of pairs from different types of lexical–semantic relations. In Section 4.3, we present benchmark experiments on using BiRD as a testbed to evaluate various semantic composition methods. Specifically, we conduct experiments to gain insights into research questions such as: Which common composition method captures the semantics of a bigram more accurately?; Which of the two terms in a bigram has greater influence on the semantics of the bigram?. Finally, Section 4.4 presents the related work and reviews the existing gold standard datasets on semantic relatedness and similarity.

We also developed interactive visualizations that allow for easy exploration of the dataset. The BiRD and visualizations of the data are made freely available online.[1]

4.1 BiRD: Bigram Semantic Relatedness Dataset

4.1.1 Term Pair Selection

Randomly selecting term pairs (AB–X) will result in most pairs being unrelated. This is sub-optimal in terms of the human annotation effort that is to follow. Further, since our

[1] http://saifmohammad.com/WebPages/BiRD.html

goal is to create a gold standard relatedness dataset, we wanted it to include term pairs across the whole range of semantic relatedness: from maximally unrelated to maximally related. Thus, a key challenge in term-pair selection is obtaining pairs with a wide range of semantic relatedness scores, without knowing their true semantic relatedness in advance. In addition, we wanted the dataset to satisfy the following criteria:

- For each target bigram AB, we wanted to include several pairs of the form AB–X, where X is a unigram or bigram.
  Motivation: Applications of semantic relatedness, such as real-word spelling correction (Hirst and Budanitsky, 2005) and textual entailment (Mirkin et al., 2006), often require judgments of the form '*is AB–X_1 more related or less related than AB–X_2*'.
- There should exist some pairs AB–X, such that X is BA and a common English bigram.
  Motivation: This is useful for testing the sensitivity of semantic composition models to word order.
- The unigrams and bigrams should be commonly used English terms.
  Motivation: Data annotation of common terms is expected to be more reliable. Also, common terms are more likely to occur in application datasets.
- There should exist pairs that are taxonomically related (i.e., semantically similar), for example, hypernyms, hyponyms; and there should exist pairs that are not taxonomically related but semantically related nonetheless.
  Motivation: This increases dataset diversity.
- We focus on noun phrases (adjective–noun and noun–noun bigrams).
  Motivation: Noun phrases are the most frequent phrases in English.

To pursue these criteria, we compiled a set of term pairs from three diverse resources (Wikipedia, WordNet, and a machine translation phrase table) as described below.

Wikipedia: We chose to collect our target bigrams from the English Wikipedia dump (2018).[1] The corpus was tagged with POS using the Natural Language ToolKit (NLTK).[2] For each of the adjective–noun and noun–noun bigrams AB in the corpus, we checked to see if the bigram BA (its transpose) also exists in the corpus. We refer to such pairs of bigrams as *transpose pairs*. Only those transpose bigrams (AB and BA) were selected that were both noun phrases and where both AB and BA occur in the corpus with frequencies greater than a pre-chosen threshold t (we chose $t = 30$). For a pair of transpose bigrams, the bigram with the higher frequency was chosen as AB, and the bigram with the lower frequency was chosen as the corresponding BA. The above process resulted in 4,095 transpose pairs (AB–BA).

WordNet: WordNet is a lexical English database from which the lexical–semantic relations between terms such as synonymy, hypernymy, etc. can be extracted (Fellbaum, 1998). WordNet represents word senses, the many different meanings that a single word can have. Therefore, there are synonym sets (called synset) indicating a distinct sense and containing synonymous words. A word with several senses (meanings) belongs to different synsets. Each synset is connected via lexical–semantic relations, such as hypernymy to other synsets. The different senses of a word are ordered based on their

[1] https://dumps.wikimedia.org/
[2] https://www.nltk.org/

frequency in a certain text. If there is no information on frequency, they are ordered randomly. Among the 4,095 ABs, 330 exist in WordNet version 3.0.[1] For each of these, we selected (when available) synonyms (at most five), a hypernym, a hyponym, a holonym, and a meronym from WordNet.

Translation Phrase Table: Word-aligned parallel corpora map words in a text of one language to those in a text of another language. Often this can lead to more than one word/phrase in one language being mapped to a common word/phrase in the other language. We refer to such terms as being *co-aligned*. Due to the nature of languages and the various forms that the same text can be translated to, co-aligned terms tend to include not just synonyms but also other semantically related terms, and sometimes even unrelated terms. Thus, we hypothesize that it is beneficial to include pairs of co-aligned terms in a semantic relatedness dataset as they pertain to varying degrees of semantic relatedness.

These co-aligned pairs can be extracted from Phrase Tables used in automatic machine translation systems. Phrase tables, generated from parallel corpora, indicate the probability of a word/phrase being translated to a word/phrase in a different language.

We used an English–French phrase table from the Portage Machine Translation Toolkit (Larkin et al., 2010) to determine additional pairs AB–X. French was chosen as it is close to English and there exist English–French parallel corpora of sufficient size. This phrase table was created using a large English–French parallel corpus harvested from the Canadian government web sites. The phrase pairs in source–target languages were filtered out using an approach based on the statistical significance of pair co-occurrence in the parallel documents as described in the work by (Johnson et al., 2007). The final phrase pairs were assigned with alignment frequency in the parallel documents. Fig. 4.1 shows an example of the phrase *software development* paired with French phrases in the second column and their corresponding frequencies in the third column. Note that the number of pairs are more than what is shown in the example table.

English term	**French translations**	**Frequency**
software development	développement de logiciels	232
	conception de logiciels	71
	élaboration de logiciels	66
	le développement de logiciels	65
	logiciels	33
	mise au point de logiciels	28
	...	
	production de logiciels	7
	réalisation de logiciels	3
	nouveau logiciel	1

Figure 4.1: Example of some French translations for bigram AB=*software development* with their frequencies in the phrase table.

To extract the co-aligned term pairs from the phrase table, we looked for each bigram AB (e.g., software development) in the phrase table. If AB is a translation of a term X' (e.g., développement de logiciels) in French with a frequency of at least $t = 30$, we collect

[1] https://wordnet.princeton.edu/download/current-version

French term	English translations	Frequency
développement de logiciels	software development	232
	software	13
	develop software	7
	developing software	5
	software engineering	1
	experimental development	1
conception de logiciels	software development	71
	software design	43
élaboration de logiciels	software development	66
	user interface	2
	development	2
le développement de logiciels	software development	65
	developing software	7
logiciels	software development	33
	computer programming	30
	computer applications	12
	computer program	5
	existing computer	2
	industrial machinery	2
	computer	87
	source	77
	product	69
	application	30
	program	11

Figure 4.2: Example of the collected English terms that are aligned to some of the AB=*software development*'s French translations with alignment frequencies in the phrase table.

other possible English translations of X' (see, for example, Fig. 4.2). Then, we sort all English translations based on their frequency. We repeat this process for all eligible X' phrases in Fig. 4.1, as can be seen in Fig. 4.2. Specifically, for each AB–X' entry in the phrase table (where X' is a French term) we keep the five most frequent English unigrams and the five most frequent English bigrams other than AB that are aligned to X' and pair them with AB to create co-aligned pairs (e.g., (software development–computer programming)). Note that all the extracted terms can be either a noun unigram or a nominal bigram. Moreover, unigrams and bigrams are distinct from words A and B.

Among the 4,095 ABs, 454 occurred in the phrase table. This resulted in 3,255 AB–X pairs in total, 1,897 where X is a unigram, and 1,358 where X is a bigram.

Finally, after the selection of term pairs from the three resources, we chose to filter the term pairs, keeping only those ABs that occurred in at least three unique pairs. Therefore, for a given AB, apart from the AB–BA entry, there should be at least two other entries of the form AB–X, generated using WordNet or the phrase table. We also manually examined the remaining entries and removed those with terms that are not known to many people. For instance, we removed the bigram "breeding stock" since its meaning is not known to many people. The final master term pair list consists of 3,345 AB–X pairs in total (1,718 where X is a unigram, and 1,627 where X is a bigram), corresponding to 410 ABs. Thus on average, each AB occurred in about eight

Source	# a–n	# n–n	# both
Wikipedia_transpose	80	330	410
WordNet_synonym	18	70	88
WordNet_is-a	49	220	269
WordNet_part-whole	7	30	37
PhraseTable_co-aligned	440	2,101	2,541
All	**594**	**2,751**	**3,345**

Table 4.1: Number of pairs from different sources. a–n denotes adjective–noun pairs and n–n denotes noun–noun pairs.

distinct pairs. This is yet another aspect that makes BiRD unique, as existing datasets were not designed to include terms in multiple pairs. Table 4.1 shows the number of adjective–noun pairs, the number of noun–noun pairs, and the total number of pairs from different resources in BiRD. We grouped the hypernym and hyponym pairs into a common class, which we refer to as the *is-a* pairs. Similarly, we group the meronym and holonym pairs into a common class, which we refer to as the *part-whole* pairs.

4.1.2 Annotating for Semantic Relatedness

We use the comparative annotation method BWS (Louviere, 1991; Cohen, 2003; Louviere et al., 2015) to obtain the annotations. Existing datasets suffer from shortcomings due to the annotation techniques employed. Except in the case of a few small but influential datasets, such as those by Miller and Charles (1991) and Rubenstein and Goodenough (1965), annotations were obtained using the rating scales technique. In this technique, annotators are asked to choose from categorical or discrete numerical values to rate the data. For instance, when annotating a pair of words for semantic relatedness, the annotator can be asked to choose among integer values from 1 to 5, with 1 representing that the two words are least semantically related or semantically unrelated, and 5 representing that the words are strongly semantically related. Rating scales suffer from significant known limitations, including: inconsistencies in annotations by different annotators, inconsistencies in annotations by the same annotator at different times, scale region bias (annotators often have a bias toward a portion of the scale, usually toward the middle), and problems associated with a fixed granularity (an annotator may want to assign 1.5 to an item instead of 1 or 2; Presser and Schuman, 1996).

BWS is an annotation technique that addresses the limitations of the rating scales technique by employing comparative annotations (Louviere, 1991; Cohen, 2003; Louviere et al., 2015; Kiritchenko and Mohammad, 2017). Annotators are given n items at a time (an n-tuple, where $n > 1$ and commonly $n = 4$). They are asked which item is the *best* (highest in terms of the property of interest) and which is the *worst* (lowest in terms of the property of interest). When working on 4-tuples, best–worst annotations are particularly efficient because each best and worst annotation will reveal the order of five of the six items (i.e., for a 4-tuple with items A, B, C, and D, if A is the best, and D is the worst, then A > B, A > C, A > D, B > D, and C > D). Therefore, the annotator is not concerned with rating the data using categorical or discrete numerical

values and a fixed granularity. Moreover, since the annotator annotates the best and worst responses, scale region bias issues are resolved. It has been analytically shown that annotating $2N$ 4-tuples produces highly reliable scores, where N is the number of items to be annotated (Louviere, 1991; Kiritchenko and Mohammad, 2016a, 2017; Mohammad, 2018a). Kiritchenko and Mohammad (2017) showed through empirical experiments that BWS produces more reliable and more discriminating scores than those obtained using rating scales.

From the list of $N = 3,345$ term pairs, we generated $2N = 6,690$ distinct 4-tuples (each 4-tuple is a set of four term pairs) such that each term pair appears in roughly equal distinct tuples, and no term pair appears more than once in a tuple. If 2N 4-tuples are generated from N pairs, and each pair is to occur in an equal number of tuples, then each pair will occur in eight tuples. The annotators were presented with a 4-tuple at a time and were asked to specify which of the four pairs is closest in meaning (or most related) and which is the least close (or least related). Detailed annotation instructions, with examples of appropriate and inappropriate responses, were provided. Notably, we made it clear that if terms in the pair have several meanings in different contexts, then the annotators should consider the meanings that are closest to each other or the meanings in the same context. We also asked the annotators to be mindful of word order (i.e., the meaning of a bigram AB may be different from the meaning of its transpose BA). An example is as follows:

Q1: Which pair is closest in meaning (or most related)?

- (building block, unit)
- (traffic light, intersection)
- (water quality, health)
- (fantasy world, system)

Answer: (building block, unit)

Q2: Which pair is least close in meaning (or least related)?

- (building block, unit)
- (traffic light, intersection)
- (water quality, health)
- (fantasy world, system)

Answer: (fantasy world, system)

The full questionnaire along with annotation instructions and several examples is provided in the Appendix 6.

We set up the annotation task on the crowdsourcing platform Figure Eight[1]. We did not collect personally identifiable information from the annotators. The compensation that the annotators would receive was clearly stated. We selected a pool of annotators

[1]https://www.figure-eight.com/

fluent in English and with a history of high-quality annotations. Annotators were told that they could annotate as many instances as they wished. Prior to the annotation, the planned procedure was approved by the National Research Council Canada's Research Ethics Board.

We annotated bout 2% of the data. These questions are referred to as gold questions. Figure Eight interspersed the gold questions with the other questions. If a crowd worker answered a gold question incorrectly, then they were immediately notified. This served as an additional way to guide the annotators. If an annotator's accuracy on the gold questions fell below 70%, then they were refused further annotation, and all of their annotations were discarded. This served as a mechanism to avoid malicious annotations.

In the task settings for Figure Eight, we specified that we needed annotations from eight people for each 4-tuple. Note that since each term pair occurs in eight different 4-tuples, it is involved in $8 \times 8 = 64$ best–worst judgments. In all, 57,482 pairs of best and worst responses were obtained from 427 annotators. Gold questions were annotated more than eight times.

Annotation Aggregation:

After the completion of BWS responses and annotations by annotators, the final semantic relatedness scores were calculated from the BWS responses using a simple counting procedure (Orme, 2009; Flynn and Marley, 2014): For each term pair, the semantic relatedness score is the proportion of times the term pair AB–X was chosen as the best minus the proportion of times the term pair was chosen as the worst:

$$r_{AB-X} = \frac{\#best_{AB-X}}{num} - \frac{\#worst_{AB-X}}{num},$$

where num is the total number of 4-tuple judgments (i.e., best–worst responses by annotators), in which the pair AB–X occurs. $\#best_{AB-X}$ and $\#worst_{AB-X}$ are the number of times AB–X was chosen as the best or worst response in the 4-tuple annotations. The scores range between -1 to 1. Then, they were linearly transformed to the interval: 0 (lowest semantic relatedness) to 1 (highest semantic relatedness). Now, BiRD consists of 3,345 English term pairs along with their real-valued scores for semantic relatedness.

4.1.3 Reliability of Data Annotations

A commonly used measure of quality in dimensional annotation tasks is the reproducibility of the final scores—the extent to which repeated independent manual annotations produce similar results. To assess this reproducibility, we calculate average SHR (Cronbach, 1951) as follows: The annotations for each 4-tuple are randomly split into two halves. One set is put in bin 1 and another set in bin 2. Next, two independent sets of semantic relatedness scores for all pairs are produced independently from the two bins, 1 and 2, respectively, using the counting procedure. Then the Pearson correlation between the two sets of scores is calculated. The correlation between the two sets of relatedness scores determines the quality of the annotations. High quality annotations have a higher Pearson correlation closer to 1. This process is repeated 100 times, and the correlations are averaged (Fig. 4.3). An SHR of $r = 0.9374$ indicates high reliability. Table 4.2 summarizes key annotation statistics.

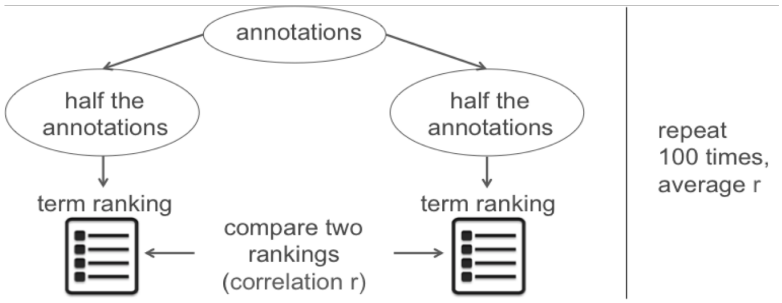


Figure 4.3: Split-half reliability approach to determine the consistency of dataset. Pearson value r is used to determine the reliability degree.

# Term Pairs	# Tuples	# Annotations per Tuple	# Annotations	# Annotators	SHR
3,345	6,690	8 (for most tuples), >8 (for some)	57,482	427	0.9374

Table 4.2: BiRD annotation statistics. SHR = Split-Half Reliability (as measured by Pearson correlation).

4.2 Studying Bigram Semantic Relatedness

Since very little prior work exists on the semantic relatedness of bigrams, several research questions remain unanswered, including:

- If both AB and BA are common English bigrams, then what is the average semantic relatedness between AB and BA?
- What is the range of semantic relatedness between a bigram and its hypernym or hyponym? What is the average semantic relatedness of such pairs? How do these averages and standard deviations vary with respect to the different lexical–semantic relations?
- What is the distribution of semantic relatedness values for co-aligned terms?

We now present analyses of the relatedness dataset to obtain insights into these questions. Fig. 4.4 shows example adjective–noun and noun–noun entries from BiRD. Observe that for the term *adult female*, the WordNet synonym and the transposed bigram (BA) are marked as being most related. Note that the WordNet-provided hyponym *amazon* is marked as less related (probably because that sense of *amazon* is rare). For the term *ageing population*, the most related term is *ageing society*—a co-aligned term in the phrase table. (Other co-aligned terms have lower relatedness scores.) The transpose bigram *population ageing* is also marked as highly related. WordNet does not provide a synonym for *ageing population*. The term *green light* is most related to *go-ahead* from WordNet synonym and *traffic light* from WordNet *is-a* relations. Its transposed bigram *light green* is marked as only slightly related. The term *science laboratory* is also mostly related to *lab* and *research lab* which are the synonyms from WordNet. BiRD can be examined for each individual relation and sorted by relatedness scores to determine other example pairs that seemingly should be closely related, but are not highly semantically related in the perception of the average English speaker. These include pairs such as

subject area–discipline (WordNet synonym) and *frying pan–spider* (WordNet hyponym). The AB–BA pairs with low relatedness, such as *law school–school law, home run–run home,* and *traffic light–light traffic* are especially useful in testing whether semantic composition methods generate suitably different representations for the terms in such pairs.

Table 4.3 shows the average semantic relatedness scores as well as standard deviations for the term pairs from various sources.

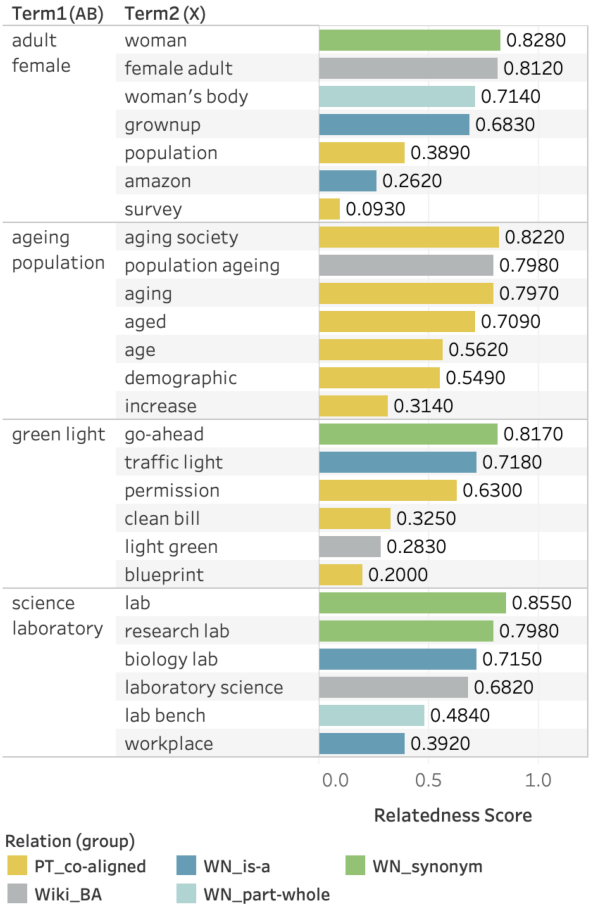


Figure 4.4: Example entries from BiRD.

Observe that, on average, the AB–BA pairs and the AB–WordNet synonym pairs are found to be the most related. On average, the AB–WordNet part-whole pairs and the AB–phrase table co-aligned pairs have the lowest semantic relatedness scores. The high average relatedness and low standard deviation (σ) for the transpose bigrams indicate that these pairs tend to be closely related to each other. The standard deviation is markedly higher for the other sources of term pairs. Manual examination of such pairs (especially those involving WordNet synonyms) revealed that this is often because one of the terms might be related to the other in a rare sense (such as in the *adult female–amazon* pair). The high standard deviations for hypernyms, hyponyms, meronyms, and

Source	avg. rel.	σ
Wikipedia_transpose	0.669	0.118
WordNet_synonym	0.640	0.194
WordNet_is-a	0.550	0.177
WordNet_part-whole	0.453	0.193
PhraseTable_co-aligned	0.463	0.189

Table 4.3: Average and standard deviation (σ) of relatedness scores for term pairs from various sources.

holonyms, indicate that pairs connected by these relations in the WordNet can still exhibit a wide range of semantic relatedness.

The standard deviations also indicate that 95% of the co-aligned pairs have semantic relatedness scores between 0.09 and 0.83 (a wide interval). Manual examination revealed that the lowest score pairs were unrelated, and the highest score terms were often synonymous. Thus co-aligned pairs from phrase tables are indeed a good source of term pairs for a semantic relatedness dataset since they include pairs with a wide variety of relatedness values.

Fig. 4.5 illustrates the trend of each type of relation for relatedness scores in the interval $[0, 1]$ (grouped into bins of size 0.05). The colors show all seven relation types. The numbers on each bar indicate the number of records for each relation type. Complete interactive visualizations of the data to explore more examples and further study on the relatedness from different resources are made freely available online.[1]

4.3 Evaluating Methods of Semantic Composition on BiRD

Language representation models have been proposed to represent word meaning in NLP. An area of active research is how these word vector representations can be composed to create representations for larger linguistic units of text such as phrases and sentences (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012; Tai et al., 2015). Even though there is a large body of work on how to represent the meanings of sentences (Le and Mikolov, 2014; Kiros et al., 2015; Lin et al., 2017), there is relatively little work on how best to compose the meanings of two words to represent the meaning of a bigram. One reason for this is a lack of gold standard evaluation resources. A common approach to evaluate semantic composition method is through their ability to rank pairs of terms by closeness (or relatedness) in meaning (Pennington et al., 2014; Levy and Goldberg, 2014; Faruqui and Dyer, 2014). BiRD allows for the evaluation of semantic composition methods through their ability to rank pairs involving bigrams, by semantic relatedness.

Here, we present benchmark experiments on commonly used semantic composition methods by measuring their ability to rank the term pairs in BiRD by relatedness scores. The underlying assumption is that the more accurately a method of semantic composition can determine the representation of a bigram, the more accurately it can determine the

[1] http://saifmohammad.com/WebPages/BiRD.html

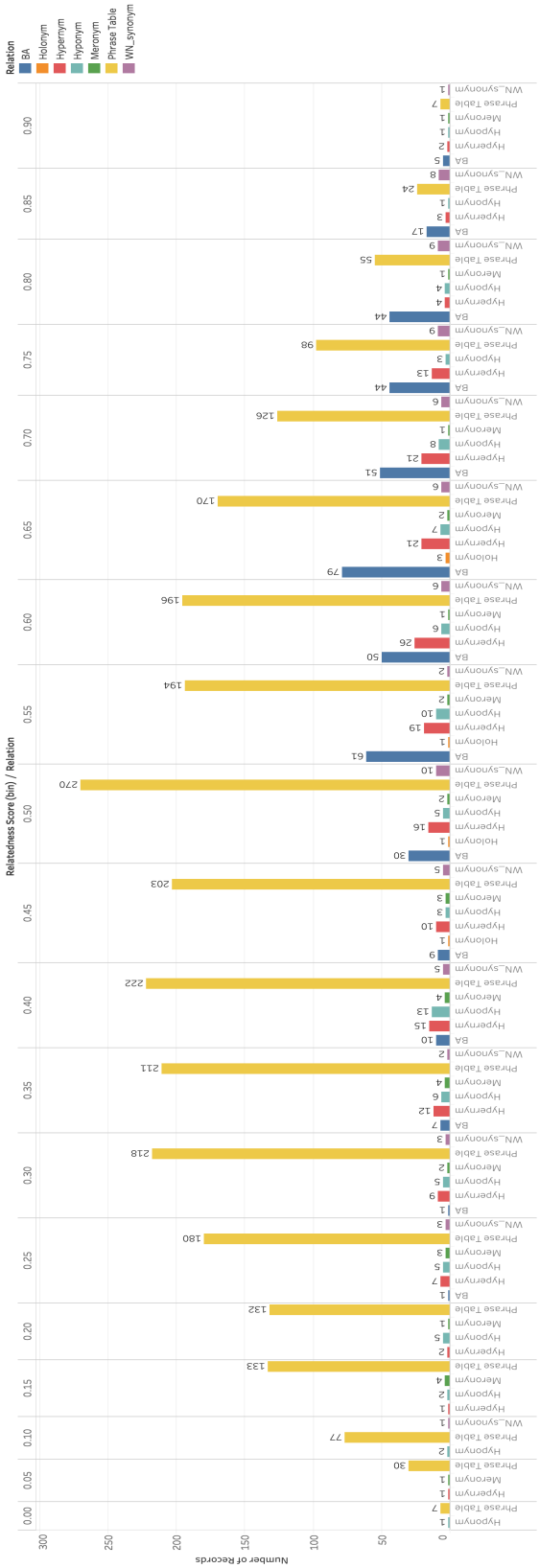


Figure 4.5: Sum of number of records for each relation broke down by relatedness scores along $[0, 1]$ (grouped into bins of size 0.05). The bars are labeled by the number of records for each relation type.

relatedness of that bigram with other terms.

We focus on unsupervised approaches[1] as we wanted to identify how well basic composition operations perform. The applicability of BiRD is broader though, and it can, for instance, be used for the following purposes: (1) evaluating a large number of proposed supervised methods of semantic composition, such as LSTM-based methods (Zhu et al., 2016); and (2) evaluating the large number of measures of semantic relatedness, such the measures introduced by Budanitsky and Hirst (2006); and (3) studying the mechanisms underpinning semantic composition.

We test four natural language representation models in vector space to obtain word representations: ELMo (Peters et al., 2018), fastText (Bojanowski et al., 2017), GloVe (Pennington et al., 2014), and a traditional model based on matrix factorization of a word–context co-occurrence matrix (Turney et al., 2011). We test four mathematical composition operations on representation models: (1) vector addition, (2) element-wise vector multiplication, (3) tensor product with circular convolution (Widdows, 2008), and (4) dilation (Mitchell and Lapata, 2010). In adjective–noun and noun–noun bigrams, the second word usually plays the role of a head noun, and the first word is a modifier. We test the performance of two baseline methods that do not employ vector composition: one that represents a bigram with the vector for the first word and one that represents a bigram with the vector for the second word.

Word Representations:

We use ELMo representations pre-trained on the 1 Billion Word Benchmark corpus,[2] GloVe word embeddings pre-trained on the 840B-token CommonCrawl corpus,[3] and fastText word embeddings pre-trained on Common Crawl and Wikipedia using CBOW.[4] For the traditional model, we use the exact word–context co-occurrence matrix described in Turney et al. (2011) and explained in Chapter 2.1.1. They created the matrix from a corpus of 5×10^{10} tokens gathered from university websites. The rows correspond to terms (single words from WordNet) and columns correspond to contexts (single words from WordNet appearing to the left or the right of the term in phrases or sentences on the websites). Each element of the matrix is the Positive Pointwise Mutual Information (PPMI) score (Church and Hanks, 1990) between the word and the context. PMI is a measure of association between words in a given text. The PMI score between two words x and y is computed given the probability of each word $p(x)$ and $p(y)$ and the joint probability of the words $p(x, y)$ in the given text as follows:

$$PMI_{x,y} = \log \frac{p(x, y)}{p(x)p(y)}.$$

The word probabilities are estimated by counting the number of occurrences of words x and y and normalizing by N, the size of the corpus (i.e., the total number of words in the corpus). $p(x, y)$ is estimated by counting the number of co-occurrence of the words (i.e., number of times y follows x), and normalizing by N. PMI can take negative and positive

[1] In these approaches semantic composition is obtained without training a specific model for compositionality.

[2] https://allennlp.org/elmo

[3] https://nlp.stanford.edu/projects/glove/

[4] https://fasttext.cc/docs/en/crawl-vectors.html

values, where a value of zero indicates the statistical independence between occurrences of two words, and higher values show more association between them. PMI can be seen as representing global information of a word in relation to the entire corpus. A variation of PMI is Positive PMI (PPMI) in which the negative PMI values are mapped to zero. This way, PPMI gives a high value to two words when there is an interesting semantic association between them and otherwise is zero indicating that the co-occurrence of the two words is not informative (Turney et al., 2011).

The extracted co-occurrence matrix is decomposed to $U_d \Sigma_d V_d^\top$ via truncated Singular Value Decomposition (SVD). SVD is a factorization technique that decomposes a word–context matrix $M \in \mathbb{R}^{m \times n}$ into the product of three matrices $U \Sigma V^\top$. Σ with size $r \times r$ is a diagonal matrix of singular values expressing the importance of each dimension where r is the rank of M. U of size $m \times r$ and V of size $n \times r$ are orthonormal matrices. If M is a high-dimensional sparse matrix, we can select the top d singular values from Σ where $d < r$ and let U_d and V_d be the matrices that are created by selecting the corresponding columns from U and V, meaning that we selected the d most important dimensions from the original matrix. Hence, $\hat{M} = U_d \Sigma_d V_d^\top$ is the truncated SVD which is a low-dimensional matrix that best approximates the original matrix M, meaning that it minimizes the Frobenius norm $||M - \hat{M}||_F$ (or Euclidean norm) over all matrices $\hat{M}$ wit rank d, which captures most important information from the original matrix.

Word vectors are obtained from the matrix $U_d \Sigma_d^p$, where rows correspond to the d-dimensional word vectors and p is the weight factor for singular values in Σ_d. p adjusts the weights of the dimensions and can vary between -1 to 1. When p is -1 it shows that smaller singular values have more weight, and when it is 1 it means that we give more weight to dimensions with higher singular values. We set parameter p to 0.5, the dimensionality of word vectors in GloVe, fastText and Word–Context Matrix to $d = 300$, and ELMo to $d = 1024$.

Unsupervised Compositional Models:

For a bigram $w_1 w_2$, let $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$ denote the vectors for words w_1 and w_2, respectively. Each of the methods below applies a different composition function f on the word vectors $\mathbf{u}$ and $\mathbf{v}$ to obtain the vector representation p for the bigram $w_1 w_2$: $\mathbf{p} = f(\mathbf{u}, \mathbf{v})$:

- *Addition* (Salton and McGill, 1986): add the two word vectors ($\mathbf{p} = \mathbf{u} + \mathbf{v}$).
- *Multiplication* (Mitchell and Lapata, 2010): element-wise multiplication of the two vectors ($\mathbf{p} = \mathbf{u} \odot \mathbf{v}$, where $\mathbf{p}_i = \mathbf{u}_i \cdot \mathbf{v}_i$).
- *Tensor product with convolution* (Widdows, 2008): outer product of two vectors resulting in matrix Q ($\mathbf{q}(i,j) = \mathbf{u}(i)\mathbf{v}(j)$). Then, circular convolution is applied to map Q to vector $\mathbf{p}$. This is equivalent to:

$$\mathbf{p}(i) = \sum_j \mathbf{u}(j) \cdot \mathbf{v}(i-j) \quad \text{for } 1 \leq i, j \leq 1.$$

- *Dilation* (Mitchell and Lapata, 2010): decompose $\mathbf{v}$ to parallel and orthogonal

components to $\mathbf{u}$, and then stretch the parallel component along $\mathbf{u}$:

$$\mathbf{p}(i) = \mathbf{v}(i)\sum_j \mathbf{u}(j)\mathbf{u}(j) + (\lambda - 1)\mathbf{u}(i)\sum_j \mathbf{u}(j)\mathbf{v}(j)$$

for $1 \leq i \leq d$, where λ is the dilation factor. We set $\lambda = 2$.

For the two baseline experiments that do not employ vector composition, we consider *head only*: $\mathbf{p} = \mathbf{v}$ and *modifier only*: $\mathbf{p} = \mathbf{u}$.

Semantic Relatedness:

The relatedness score for a term pair AB–X in the BiRD is computed by taking the cosine between the vectors representing AB and X, where X can be a unigram or a bigram. Given $\mathbf{u}$ and $\mathbf{v}$ as the vectors for AB and X, respectively, the cosine is computed as follows:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}.\mathbf{v}}{\parallel \mathbf{u} \parallel \parallel \mathbf{v} \parallel},$$

where $\parallel \cdot \parallel$ computes the Euclidean norm of a vector $\mathbf{v}$ of size d as follows:

$$\parallel \mathbf{v} \parallel = \sqrt{\mathbf{v}(1)^2 + \cdots + \mathbf{v}(d)^2}.$$

Evaluation:

As evaluation metric, we use the Pearson correlation ρ of the relatedness scores predicted by a method with the gold relatedness scores in BiRD, which measures the linear relationship between the two values. The Pearson coefficient value ranges from -1 to 1 with higher values showing more correlation between the predicted and gold values and lower values showing they are more inversely correlated. Some words in BiRD do not occur in some of the corpora used to create the word vectors. Thus, we conduct experiments on a subset of BiRD (3,159 pairs) for which word vectors exist for all models under consideration. To determine if the differences between the correlation scores are statistically significant, we perform Steiger's Z significance test (Steiger, 1980).[1] The procedure for evaluation is summarized in Fig. 4.6. As it is shown, first word vectors from a representation model are obtained, and then the compositional representation of the bigrams are computed using a semantic composition function. The cosine values between the two term representations in pairs are computed, and finally, they are compared with the gold relatedness scores in BiRD using the Pearson correlation.

Results:

Table 4.4 shows the results. Observe that among the methods of semantic composition, the addition model performs best (for all four ways of representing word vectors). The scores are statistically significantly higher than those of the second best (dilation). The element-wise vector multiplication and tensor product with convolution perform poorly (even worse than the baseline methods). Among the four models of word vector representations, the best results of the addition model are obtained using ELMo and fastText models performing competitively. Overall, the results show that ELMo and

[1]Statistical significance refers to the claim that the obtained results are not likely to occur by chance.

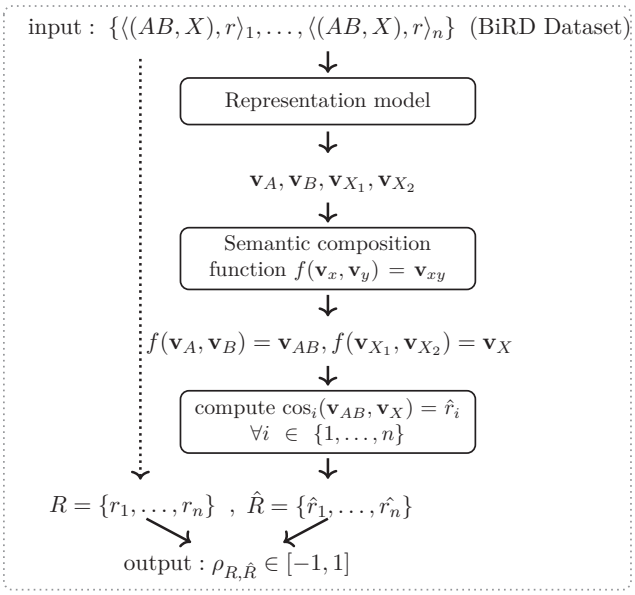


Figure 4.6: Illustration of the evaluation procedure for semantic composition methods using BiRD. Note that here we assume that X is a bigram. If it is a unigram, then $\mathbf{v}_X$ is computed in the first step. $\rho_{R,\hat{R}}$ is the Pearson correlation score.

fastText with addition as the composition operation capture the meaning composition better than other representation models and composition operations. These results differ substantially from the observations by Mitchell and Lapata (2010). In particular, in their work the multiplication model showed the best results, markedly outperforming the addition model. Our results are consistent with the findings of Turney (2012), where the addition model also performed better than the multiplication model. It should be noted though that unlike BiRD, which has scores for semantic relatedness, the Mitchell and Lapata (2010) and Turney (2012) datasets have scores for semantic similarity.

Method	**ELMo**	**GloVe**	**fastText**	**Matrix Factor.**
Baselines				
head only	0.487	0.342	0.403	0.339
modifier only	0.460	0.438	0.495	0.425
Composition methods				
addition	**0.603**	**0.564**	**0.601**	**0.582**
multiplication	0.203	0.182	0.328	0.244
tensor product	0.380	0.374	0.382	0.451
dilation	0.589	0.523	0.569	0.496

Table 4.4: Pearson correlations of model predictions with BiRD relatedness scores. Highest scores are in bold.

Surprisingly, the baseline model that uses the vector for the modifier word obtains better results than the one that uses the vector for the head noun. The difference is

statistically significant. To better understand this, we compute relatedness correlations using the *weighted addition* of the two word vectors ($\mathbf{p} = \alpha\mathbf{u} + (1-\alpha)\mathbf{v}$), where α is a parameter that we vary between 0 and 1, in steps of 0.1. Fig. 4.7 shows the results. Observe that giving more weight (but not too much weight) to the modifier word than the head word is beneficial. $\alpha = 0.7$ and $\alpha = 0.8$ produce the highest correlations. These results raise further questions as to under what conditions the role of the modifier is particularly prominent, and why.

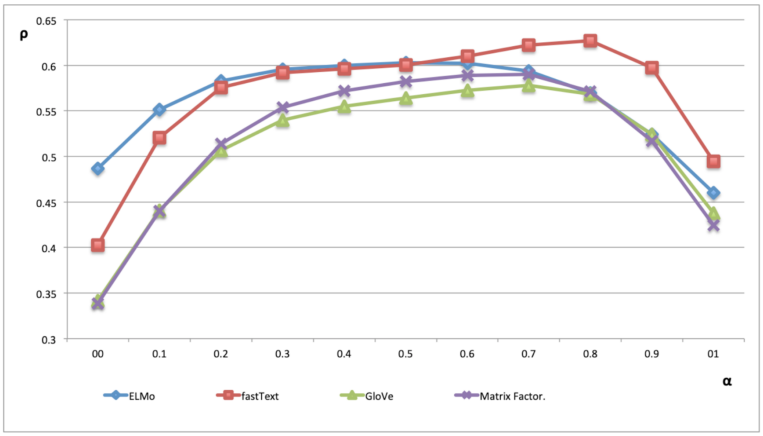


Figure 4.7: Pearson correlation coefficient (ρ) of the model predictions using weighted addition with BiRD relatedness scores. α varies from 0 to 1 in steps of 0.1. $\alpha = 0.7$ and $\alpha = 0.8$ produce the highest scores.

4.4 Related Work

In this section we review the related work in semantic similarity and semantic relatedness datasets for evaluation of CDSMs, on word and phrase levels.

Evaluation of CDSMs:

As discussed before, a common approach to examining CDSMs in capturing meaning composition is through their ability to rank pair of terms (e.g. phrases) by their closeness in meaning. Closeness of meaning can be of two kinds: semantic similarity and semantic relatedness. Both semantic relatedness and similarity can be used to evaluate CDSMs and several datasets have been proposed for examining CDSMs. However, we argued semantic relatedness is the broader class subsuming semantic similarity. Moreover, many psychological and neuro linguistic studies have demonstrated the importance of semantic relatedness. Therefore, we propose semantic relatedness as a more suitable approach for examining CDSMs in capturing meaning composition.

There exist other approaches for examining CDSMs. The lexical substitution approach evaluates the meaning composition at the level of individual words (McCarthy, 2002; McCarthy and Navigli, 2007). In this approach, the task is to find a meaning-preserving replacement for a target word in the context of a sentence. For instance, the word *match*

in the sentence "We spent the afternoon at a football match." can be replaced with *game* without changing the meaning of the sentence. This approach addresses the word sense disambiguation issue by providing the context of the target word in which it occurs. McCarthy and Navigli (2009) and Buljan et al. (2018) proposed lexical substitution datasets for evaluating CDSMs. Datasets contain sentences and a target word in each sentence for substitution evaluation. Each target word has two correct substitutes and two confounders, and CDSMs are supposed to rank the correct substitutes higher than the confounders.

Word-Pair Semantic Similarity and Relatedness Datasets:

Several semantic similarity and relatedness datasets involving unigram pairs (word pairs) exist. Rubenstein and Goodenough (1965) and Miller and Charles (1991) provided influential but small English word-pair datasets with fine-grained semantic similarity scores. More recent larger datasets including hundreds of pairs annotated with human judgments were provided by Finkelstein et al. (2002) (for relatedness) and Hill et al. (2015) (for similarity). Word-pair relatedness datasets exist in some other languages as well, such as the one by Gurevych (2006) in German, Panchenko et al. (2016) in Russian, and Santus et al. (2015) in both English and Mandarin Chinese. Huang et al. (2012) introduce a similarity dataset of pairs of words. The dataset is created with human judgments on similarity ranking of two words in sentential context, which addresses the issue of word sense disambiguation for polysemous words (i.e., words with several meanings in different contexts). However, none of the relatedness datasets include items that are bigrams.

Bigram Semantic Similarity Datasets:

Mitchell and Lapata (2010) created a semantic similarity dataset for 324 bigram pairs using crowdsourcing. The terms include adjective–noun, noun–noun, and verb–object bigrams. Using the rating scales technique, annotators were asked to choose an integer between one and seven, indicating a coarse semantic similarity rating. Turney (2012) compiled a coarse-grained dataset of 2,180 bigram–unigram synonym pairs from WordNet synsets (synonym sets) that were marked as having a similarity of 1. The bigrams are either noun–noun or adjective–noun phrases. Other pairs were created taking bigrams and random words that do not exist in the same synsets. These were marked as having a similarity of 0. He thus created a dataset of synonyms and non-synonyms. As opposed to Mitchell and Lapata (2010), this dataset was not annotated with human judgments. The dataset was used in a compositional similarity task, which was then formulated as 2,180 multiple-choice questions (seven-choice questions). In each question, the stem is the bigram of a pair and choices are unigrams in which only the unigram of that pair is the correct answer. Distractors include head word, modifier word, hypernym or synonym of the head word, hypernym or synonym of the modifier word, and two other random nouns. In contrast to these datasets, BiRD has fine-grained relatedness scores.

There exist datasets on the semantic similarity between sentences and between documents for evaluation of sentence- and document-based representation models (Marelli et al., 2014; Agirre et al., 2014; Cera et al., 2017). Those are outside the scope of this thesis as we consider composition of words to obtain phrases.

Other Natural Language Datasets Created Using BWS:

BWS has been used for creating diverse datasets for NLP tasks, such as relational similarity (Jurgens et al., 2012), word sense disambiguation (Jurgens, 2013), word-sentiment intensity (Kiritchenko and Mohammad, 2016a), word-emotion intensity (Mohammad, 2018b), and tweet-emotion intensity (Mohammad and Kiritchenko, 2018). The largest BWS dataset is the NRC Valence, Arousal, and Dominance Lexicon, which has valence, arousal, and dominance scores for over 20,000 English words (Mohammad, 2018a).

4.5 Conclusion

We created a dataset of semantic relatedness for term pairs involving bigrams with fine-grained human ratings. We used the comparative annotation technique BWS, which addresses the limitations of traditional rating scales technique. We showed that the ratings obtained are highly reliable (high SHR $r = 0.937$). We analyzed the dataset to obtain insights into the distributions of semantic relatedness scores for pairs associated through various relations, such as WordNet assigned lexical–semantic relations, transposed bigrams, and co-aligned terms in a parallel corpus. We showed that co-aligned terms can be related to varying degrees (from unrelated to synonymous), making them a useful source of term pairs to include in relatedness datasets. Finally, we presented benchmark experiments on using BiRD as a testbed to evaluate various methods of semantic composition in vector space. We found that the vector addition performed best among different composition methods when tested using different representation models. Moreover, giving more weight to the modifier word in the weighted addition model can improve results further. Among the different representation models, ELMo and fastText competitively outperform the other models. In other words, when addition is applied to ELMo and fastText representation models, it captures the semantic composition representations better than the other representation models and composition methods.

The introduced dataset provides a quantification of semantic relatedness that can be used (1) for evaluating different measures of semantic relatedness, such as those introduced in the work by Budanitsky and Hirst (2006); (2) for evaluating a large number of proposed supervised methods of semantic composition, such as LSTM-based methods (Zhu et al., 2016); and (3) for evaluations in real-word spelling correction (Hirst and Budanitsky, 2005) and textual entailment (Mirkin et al., 2006) tasks, which often require judgments of the form '*is* AB–X_1 *more related or less related than* AB–X_2'. We made BiRD freely available to foster further research.

Chapter 5

CMSMs as Word Representation Models

In Chapter 2, we studied distributed word representation models (also called word embeddings)[1] in vector space, such as word2vec (Mikolov et al., 2013b). However, one of the main challenges in word embeddings is how to achieve a word representation that captures semantic compositionality. Successful performance of CMSMs on capturing compositionality and semantics in NLP tasks studied in Chapter 3 motivated us to investigate CMSMs as generic word representation models (or word embeddings) in matrix space as opposed to word embeddings in vector space, such as word2vec.

In this chapter, we propose learning methods for CMSMs to introduce word-level matrix embeddings as an alternative model to word representations in matrix and vector spaces. As opposed to Chapter 3, where CMSMs are trained on a specific NLP task, such as sentiment analysis, in this chapter, the introduced models are not trained on any specific task to capture task-specific information. Therefore, they are called task-agnostic representation models. Moreover, they are trained using distributional information of words based on the distributional hypothesis and PMI between words in a given text corpus. The introduced matrix embeddings reflect the semantic relationships between words and can be used to represent words in downstream NLP tasks. Our goal to introduce the word-level embeddings, as opposed to sentence-level embeddings, is to allow for a dynamic composition of word matrices to longer phrases and even sentences using matrix multiplication.

We first review the related work on learning word representation models in vector and matrix spaces in Section 5.1. These models are then compared against our proposed models in the experiments.

Section 5.2 introduces two learning methods for CMSMs which result in two word-level matrix embeddings:

1. The first method is a supervised learning technique for word-level matrix embeddings based on linear regression. The training dataset consists of bigrams (two-word sequences) labeled with associated Normalized Point-wise Mutual Information (NPMI), which is a scalar. NPMI presents global information about the association between words co-occurring in a given text corpus. We create the training dataset from a large corpus of sentences by extracting bigrams and computing their associated NPMI. The objective of the training is to train a function that best approximates the NPMI value of bigrams by multiplying their word matrices and mapping the resulting matrix to a scalar. (Details in Section 5.2.2).

2. The second method is a learning technique inspired by the skip-gram method (Mikolov et al., 2013b) in vector space, explained in Chapter 2.1.3, adapted for matrix space. Similar to skip-gram, a neural network is trained using a large unlabeled corpus of sentences to maximize the probability of context words being predicted correctly, given a specific input word. However, there are substantial differences between our learning method and skip-gram, which we describe in Section 5.2.3. Since training is not based on a predetermined labeled dataset, the learning task is referred to as self-supervised, explained in Chapter 2.1.3. Moreover, after training the network, it is not used for the prediction of previously unseen

[1] The terms *representation model* and *embeddings* will be used interchangeably throughout the chapter.

input data. Instead, the weights of the network are extracted and introduced as generic word matrix embeddings. (Details in Section 5.2.3). At the core of the method is non-commutative matrix multiplication during training, which, as opposed to the skip-gram method, is word-order-sensitive when training the word matrices.
Since the network can be trained with different kinds of context window sizes, we propose three variations of the context window and report the performance of all variations. (Details in Section 5.2.3).

In Section 5.3, we evaluate the performance of our word-level matrix embeddings in capturing semantic representation and compositionality using the two following evaluation tasks:

1. The Semantic Textual Similarity (STS) task using the SentEval repository (Conneau and Kiela, 2018), which is about determining the similarity degree of a given pair of sentences.
2. The semantic relatedness task using the BiRD dataset introduced in Chapter 4 (Asaadi et al., 2019), which is about determining the relatedness degree of a given bigram–bigram or bigram–unigram pair.

In the experiment section, we compare the performance of our proposed matrix embeddings against the following three existing embeddings:

1. A state-of-the-art sentence embedding in matrix space proposed by Mai et al. (2019). They proposed a learning technique for CMSMs to obtain sentence matrix embeddings inspired by the CBOW method in word2vec.
2. skip-gram word vector embedding (Mikolov et al., 2013b) with two vector composition operations, addition and element-wise multiplication, to obtain the sentence representations for evaluation purposes.
3. Word matrix representations trained on the natural language inference task (Chung et al., 2018), which are then used to represent the meaning of words in downstream NLP tasks. Matrix multiplication is used to obtain the phrase or sentence representations for evaluation purposes.

Finally, we conclude the chapter with discussion and conclusion in Section 5.4.

5.1 Related Work

In this section, we review two very closely related works to CMSMs, which we require for the rest of the chapter. We also briefly mention the skip-gram method in word2vec introduced in Chapter 2.1.3.

Chung et al. (2018) propose a model in which CMSMs are augmented with Tree-structured LSTM (TreeLSTM), called LMS-TreeLSTM (Lifted Matrix-Space TreeLSTM). LSTM is a variation of recurrent neural networks, which, unlike feedforward NNs, has feedback connections between the hidden layers and a memory cell to maintain information in memory for a long period of time (Hochreiter and Schmidhuber, 1997).

TreeLSTM improves the semantic representation of sentences in LSTM by using the sentences' tree structure (a dependency or a constituency tree) (Tai et al., 2015). In their model, Chung et al. (2018) incorporate CMSMs into TreeLSTMs to capture multiplicative interaction in the composition of words to sentences. The superiority of multiplicative composition has been confirmed in previous studies such as (Rudolph and Giesbrecht, 2010; Socher et al., 2012, 2013) and also in our studies in Chapter 3. The whole model has three parts summarized in Fig. 5.1: the LIFT layer, the composition layer, and the Tree-LSTM. The objective is to train the network in the natural language inference task (Bowman et al., 2015). The task refers to determining the inferential relation (entailment, neutral, or contradiction) between a pair of sentences. It is used as an evaluation task for natural language understanding and is closely related to understanding the meaning of linguistic structures (e.g., phrases, sentences). They train and optimize the weights of the neural network to obtain word matrix representations in matrix space. Then sentence representations are captured utilizing the multiplicative composition, to be used in inferential relation predictions. In the LIFT layer, each input word vector $\mathbf{v} \in \mathbb{R}^d$ is transformed to matrix H using the following computation:

$$H = \tanh(\mathbf{v}\mathbf{W}_{\text{LIFT}} + B_{\text{LIFT}}), \tag{5.1}$$

where $\mathbf{W}_{\text{LIFT}} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}\times d}$ is a shared third-order tensor in the network, B_{LIFT} is the bias matrix, and $H \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}}$ is the corresponding word matrix of the vector $\mathbf{v}$. The resulting word matrices serve as input to the composition layer in which they are composed using matrix multiplication in a tree-structured LSTM to obtain the sentence matrix representation. Note that nonlinear function is applied after the multiplication of every two matrices in the tree. The final sentence matrices are then fed into a three-layer neural network as the classifier to predict the inferential relation between a pair of sentences, which is a classification task with three classes.

Details of the TreeLSTM architecture are not required for our purposes in this chapter as we only extract the LIFT layer of the network after training (i.e., the weight tensor and bias in Equation 5.1), to obtain matrix representations from word vectors. We compare our word matrix embeddings against the extracted word matrix representations in this model.

Very recently, a new task-agnostic CMSM-based model called Continual Multiplication Of Words (CMOW) and inspired by the CBOW method in word2vec has been introduced by Mai et al. (2019). The CBOW method (Mikolov et al., 2013b) trains distributed word vector representations in NLP using a two-layer feedforward NN. The network consists of trainable weight matrices $W_{V\times d}$ and $W'_{d\times V}$. V is the size of the vocabulary Σ created from a given corpus and d is the size of the hidden layer, which equals the final number of embedding dimensions. For each word $w_t \in \Sigma$, a set of context words co-occurred with w_t with a window size of k, $C_t = \{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$, is extracted from the corpus. The objective is to train the model parameters θ to maximize the log probability of the word co-occurring with the given context words, or to minimize the negative of log probability, as defined in the following:

$$\min \mathcal{O} = -\log P(w_t | w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}; \theta).$$

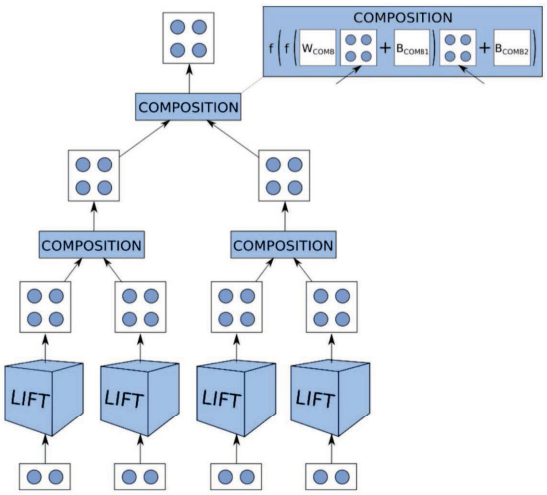


Figure 5.1: The LMS-TreeLSTM in schematic form. The word vectors are fed to the LIFT layer to produce corresponding word matrices, which are then composed using matrix multiplication to finally obtain the sentence matrix representation. Note that the TreeLSTM that contains the tree structure is not visible in the illustration. Illustration is taken from (Chung et al., 2018). Copyright (2018) by ACL.

After training the network, the weight matrix W' is extracted in which each column in the matrix is the vector representation of the corresponding word in Σ.

Mai et al. (2019) adapt the CBOW to matrix space and produce sentence-level matrix embeddings. The objective is to train the network to predict a randomly removed word from a given sentence, using its context words. While an interesting preliminary model, it restricts compositionality to the sentence level, whereas our objective in this thesis is to obtain word matrices to test composition operations on a more fine-grained level. Moreover, the composition operations used during the training procedure are based on element-wise vector multiplication and matrix multiplication as opposed to our training procedure, which is only based on the matrix multiplication.

A successful word embedding in vector space has been the skip-gram method in word2vec proposed by Mikolov et al. (2013b). In skip-gram, a two-layer feedforward NN is trained to predict the context words of a word (called center word) based on a given corpus of sentences. Similar to CBOW, the network consists of trainable weight matrices $W_{V\times d}$ and $W'_{d\times V}$. For each word $w_t \in \Sigma$, a set of context words co-occurred with w_t with a window size of k, $C_t = \{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$, are extracted from the corpus. The objective of the network is to train the model parameters θ to maximize the log probability of context words co-occurring with the given w_t, or to minimize the negative of log probability, as defined in the following:

$$\min \mathcal{O} = -\log P(w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k} | w_t; \theta).$$

After training the network, the weight matrix W is extracted in which each row in

the matrix is the vector representation of the corresponding word in Σ. Details of the method is explained in Chapter 2.1.3.

In word2vec, compositionality has primarily been achieved by training phrase embeddings (i.e., treating each phrase as a token and train the token vectors), or by applying mathematical operations to word vectors, such as averaging, element-wise multiplication, and addition. Mitchell and Lapata (2010) study these composition operations in DSMs, where they observed that multiplication performs significantly better than addition. We apply these composition operations to the skip-gram word embeddings in order to obtain phrase and sentence representations. This way we compare our matrix embeddings with vector embeddings in meaning representation and compare matrix multiplication with the vector composition operations in capturing the meaning composition of sentences.

5.2 Learning Methods

In this section, we introduce our proposed two learning methods for CMSMs to obtain word-level matrix embeddings. The first method is based on linear regression, while the second method is inspired by the skip-gram method (Mikolov et al., 2013b) adapted to the matrix space to train word matrix embeddings.

5.2.1 Matrix Initialization

A corpus consisting of a large set of English sentences is employed as the training dataset to train word matrices in both learning methods. A vocabulary Σ of size V is created by pre-processing the corpus and extracting all words. Then, each word $w \in \Sigma$ is assigned a quadratic word matrix M_w and a quadratic context matrix C_w of the same dimensionality. Yessenalina and Cardie (2011) and Asaadi and Rudolph (2017) point out that the optimization problem in word-to-matrix mapping is non-convex. Thus, random initialization of matrices is little effective and a more principled matrix initialization method is required. After experimenting with several settings, such as sampling the initial matrices uniformly in the range of $[\frac{-0.5}{d}, \frac{0.5}{d}]$, where d represents the matrix dimensionality, we found the initialization close to the identity matrix also suggested by Mai et al. (2019) to be most successful for our models. This method combines the identity matrix $\mathbb{I} \in \mathbb{R}^{d \times d}$ with a noise value from normal distribution $\mathcal{N}(0, 0.1)$ for each element of the matrix:

$$M = \mathbb{I} + \begin{pmatrix} \mathcal{N}(0,0.1) & \cdots & \mathcal{N}(0,0.1) \\ \vdots & \ddots & \vdots \\ \mathcal{N}(0,0.1) & \cdots & \mathcal{N}(0,0.1) \end{pmatrix}$$

All word and context matrices are initialized in this way.

5.2.2 Method 1: Regression-Based Method (PMI-based CMSM)

The regression-based learning method is a supervised approach, and therefore, we require a labeled training dataset. For this purpose, we use a large corpus of sentences to create the labeled dataset consisting of bigrams and their corresponding normalized PMI

(Church and Hanks, 1990). PMI is a measure of association between two terms in a given text. The PMI score between two words x and y is computed given the probability of each word $p(x)$ and $p(y)$ and the joint probability of the words $p(x, y)$ in the given text as follows:

$$PMI_{x,y} = \log \frac{p(x, y)}{p(x)p(y)}.$$

The word probabilities are estimated by counting the number of occurrences of words x and y and normalizing by the size of the corpus (i.e., the total number of words in the corpus). $p(x, y)$ is estimated by counting the number of co-occurrence of the words (i.e., number of times y follows x) and normalizing by the size of the corpus. PMI can take negative and positive values, where a value of zero indicates statistical independence between occurrences of two words, and higher values show more association between them. PMI can be seen as representing global information of a word in relation to the entire corpus.

Bouma (2009) argues that since PMI has no fixed upper and lower bound, it is not known how close a bigram is to perfect association. As the PMI value gets closer to zero, however, independence between two words given their context can be assumed. Moreover, terms of low frequency get relatively high PMI scores. It has been shown that NPMI (Bouma, 2009) illustrates the association between terms better than PMI. NPMI is a value between -1 to 1, where 1 indicates that two words only occur together, 0 shows that the two words occur independently, and -1 shows that the two words always happen separately. NPMI is also less sensitive to low-frequency data (Bouma, 2009). Thus, we use NPMI in this work which is computed as follows:

$$NPMI_{x,y} = \frac{PMI_{x,y}}{-\log p(x, y)}.$$

The sentences from the corpus are chunked into bigrams. Word and bigram frequencies are computed without any frequency cutoff since our goal is to obtain the true associative information between words. Finally, we obtain a labeled training dataset consisting of bigrams and their corresponding NPMI scores (b_i, y_i) with $1 \leq i \leq n$, where n is the size of the training dataset (i.e., the number of bigrams considered). Based on the created dataset, each word can occur as the first word or the second word in a bigram. In this method, we always consider the second word as the context of the first word.

The input to the learning algorithm is the created training dataset. Our goal is to obtain word and context matrices, which produce the bigram matrix using matrix multiplication as the composition operation. Then a mapping function is applied to map the resulting matrix to a scalar, which is the predicted NPMI value of the corresponding bigram.

The objective of the learning method is to train the model parameters to minimize the objective function defined as the MSE:

$$E = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{5.2}$$

where y_i is the target NPMI value of the bigram b_i and $\hat{y}_i$ is the predicted NPMI of the

bigram obtained from the CMSM. n is the size of the training set. Assuming that b_i has the form $w_j w_k$, we obtain $\hat{y}_i$ via

$$\begin{aligned} M_i &= M_j C_k, \\ \mathbf{v}_i &= vec(M_i), \\ \hat{y}_i &= \mathbf{v}_i \boldsymbol{\beta}, \end{aligned} \tag{5.3}$$

where M_j and C_k are the word and context matrices of bigram b_i, which produce the bigram matrix M_i. Then $vec : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d^2 \times 1}$ reshapes the matrix to a vectorized representation. Finally, the dot product of the vector-turned matrix $\mathbf{v}_i$ with a global d^2-dimensional vector $\boldsymbol{\beta}$ is computed to obtain $\hat{y}_i$.

In this method, the model parameters are the word and context matrices as well as the global mapping vector $\boldsymbol{\beta}$. We apply a gradient descent optimization technique to update word and context matrices as well as the global mapping vector $\boldsymbol{\beta}$ at each iteration of the training procedure toward minimizing the error value computed by E. Therefore, at each iteration, the objective function of the input training set is computed and based on the obtained error value, model parameters are updated as follows:

$$\begin{aligned} M_w &= M_w - \eta \frac{\partial E}{\partial M_w}, \\ C_w &= C_w - \eta \frac{\partial E}{\partial C_w}, \\ \boldsymbol{\beta} &= \boldsymbol{\beta} - \eta \frac{\partial E}{\partial \boldsymbol{\beta}}, \end{aligned}$$

where η is a fixed learning rate that determines the step size towards the minimum of the objective function. $\frac{\partial E}{\partial x}$ is the partial derivative of the objective function with respect to parameter x.

Training the model parameters stops after T iterations. Finally, we introduce M_w for all $w \in \Sigma$ as the final word matrix embedding (or word representation model). Fig. 5.2 illustrates the learning procedure.

5.2.3 Method 2: Compositional Order-Sensitive Matrix Model (COSMo)

This second method for training matrix embeddings, which we call Compositional Order-Sensitive Matrix Model (COSMo), is adapted from the skip-gram learning method in vector space (Mikolov et al., 2013b), explained in Chapter 2.1.3. In skip-gram, word vector embeddings are trained using a feedforward two-layer neural network. COSMo also trains the word matrix embeddings using a feedforward two-layer NN. As explained in Section 5.2.1, a corpus consisting of a large set of sentences is employed as the training dataset to train word matrices. A vocabulary Σ of size V is created by pre-processing the corpus and extracting all words. Recall that each word w is assigned a word matrix M_w and a context matrix C_w. Similar to the original idea in the skip-gram method, the task is to train the NN to predict the context words of a given input word based on the available corpus of sentences.

At each iteration of training in the algorithm, we input a sentence $s =$

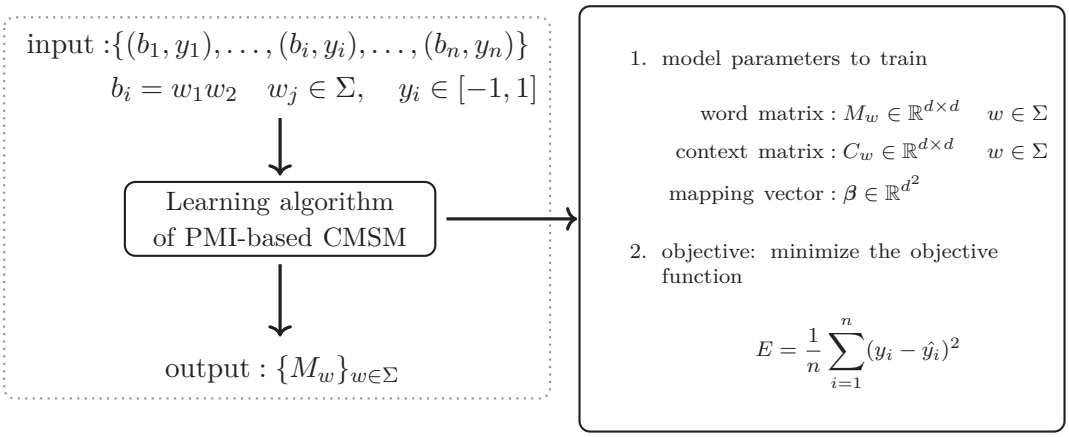


Figure 5.2: Supervised learning procedure of PMI-based CMSM. $\{(b_1, y_1), \ldots, (b_, y_n)\}$ is a training set of size n. Σ is the vocabulary extracted from the corpus.

$w_1 w_2 \ldots w_{i-1} w_i w_{i+1} \ldots w_c$ from which a predefined number n of words are randomly selected, called *center words*, that serve as the input to the network. A set of context words with a window size of l are also extracted from the same input sentence for each center word. Note that this learning method considers two versions of training with respect to the position of the context words:

1. Asymmetric training: From a given center word w_i, a specified number of context words is selected to the right. Given an input sentence s, and a context window size of l, the context words of a center word w_i are $\{w_{i+1}, \ldots, w_{i+l}\}$;
2. Symmetric training: From a given center word w_i, a specified number of context words is selected to both sides. Therefore, the context words of a center word w_i in s with a window size of l are $\{w_{i-l}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+l}\}$.

During the training, a matrix $Q^i_{pos} \in \mathbb{R}^{d \times d}$ representing the input sequence for each given center word w_i and its context words is computed utilizing matrix multiplication. In the case of symmetric context window size of l, Q^i_{pos} is calculated as follows, where the composition operation is matrix multiplication:

$$Q^i_{pos} = C_{w_{i-l}} \ldots C_{w_{i-1}} M_{w_i} C_{w_{i+1}} \ldots C_{w_{i+l}}, \tag{5.4}$$

and, in the case of asymmetric training, Q^i_{pos} is computed as:

$$Q^i_{pos} = M_{w_i} C_{w_{i+1}} \ldots C_{w_{i+l}},$$

where M_w and C_w denote the word and context matrices of a word w, respectively. Note that if the randomly chosen center word happens to be first or last word of the input sentence, which means there is no preceding or proceeding context word, each "missing" position is filled with the global Out-Of-Sentence (OOS) matrix C_{oos} that is also trained alongside the other word matrices.

In line with the original skip-gram with negative sampling (Mikolov et al., 2013b), we choose k negative samples for each center word in the input sentence that we retrieve

from a unigram probability distribution with a smoothing parameter α:

$$q(w_j) = \frac{f(w_j)^\alpha}{\Sigma_{z=1}^{V}(f(w_z)^\alpha)},$$

where $f(w)$ represents the frequency of a word w in the given corpus, and V is the size of the vocabulary. Frequencies are raised to the power of α ($0 < \alpha \leq 1$) to increase the selection probability of less frequent words and decrease the probability of more frequent words. $q(w_j)$ computes the probability that the word w_j is selected as the negative sample.

Since the multiplication of the center word matrix M_{w_i} with negative sample matrices C_{w_j} is order-sensitive, we can consider the two following variations of negative sampling in our method:

1. Asymmetric context of negative samples similar to skip-gram. In this case, the matrix of each center word w_i is multiplied with k number of negative samples as follows:

$$Q^i_{neg} = \sum_{j=1}^{k} M_{w_i} C_{w_j}; \tag{5.5}$$

2. Symmetric context of negative samples. In other words, negative samples are evenly distributed to the left and to the right of the center word in the matrix multiplication process. In this case, Q^i_{neg} is computed as follows:

$$Q^i_{neg} = \sum_{j=1}^{k} M_{w_i} C_{w_j} + \sum_{j=1}^{k} C_{w_j} M_{w_i}.$$

The objective function (also called loss function) to maximize, sums up both positive and negative samples of the center words, computed as follows:

$$\mathcal{L} = \sum_{i=1}^{m} \left(\log\ \sigma(RS(Q^i_{pos}) + \log\ \sigma(-RS(Q^i_{neg})]\right),$$

where Q^i_{pos} and Q^i_{neg} represent the composition of a center word with the specified correct context words and k negative samples, respectively. Since the composition operation results in matrices, RS represents an operation that sums all elements in the matrices Q^i_{pos} and Q^i_{neg}; in other words, for any matrix $X \in \mathbb{R}^{d \times d}$:

$$RS(X) = \sum_{i',j'} X(i', j')$$

for $1 \leq i', j' \leq d$, where then the $\log(\sigma(\cdot))$ function is applied on the resulting values, before adding those two terms. σ is the *sigmoid* function. m is the total number of center words in the training batch at each training iteration. It is computed by multiplying the batch size (number of input sentences to training at a time) with n, the number of randomly drawn center words from each sentence in the batch. Note that the objective

is to minimize the value of the composition of the center word with negative samples, and therefore, the negative of $RS(Q^i_{neg})$ is used in the equation. The composition of the center word with context words is to be maximized, and therefore, the positive of $RS(Q^i_{pos})$ is used in the equation. Similar objective functions can be found in the literature such as the work by Kaji and Kobayashi (2017).

Finally, the negative of the objective function $\mathcal{L}$ is minimized in the training process. Model parameters for training are the word and context matrices, M_w and C_w for all $w \in \Sigma$. In terms of the optimizer used, we experimentally determined gradient descent as the most efficient choice. Word and context matrices are updated at each iteration of training. A *batch* at each iteration of training is defined as a set of input sentences from the corpus, and an *epoch* is defined as a complete pass over all sentences in the corpus. Therefore, after a T number of epochs over the corpus (i.e., iterating T times over the corpus), we stop the training of matrices. Finally, word matrices M_w for all $w \in \Sigma$ are extracted and introduced as the matrix embeddings or the word representation model. Fig. 5.3 illustrates the learning procedure.

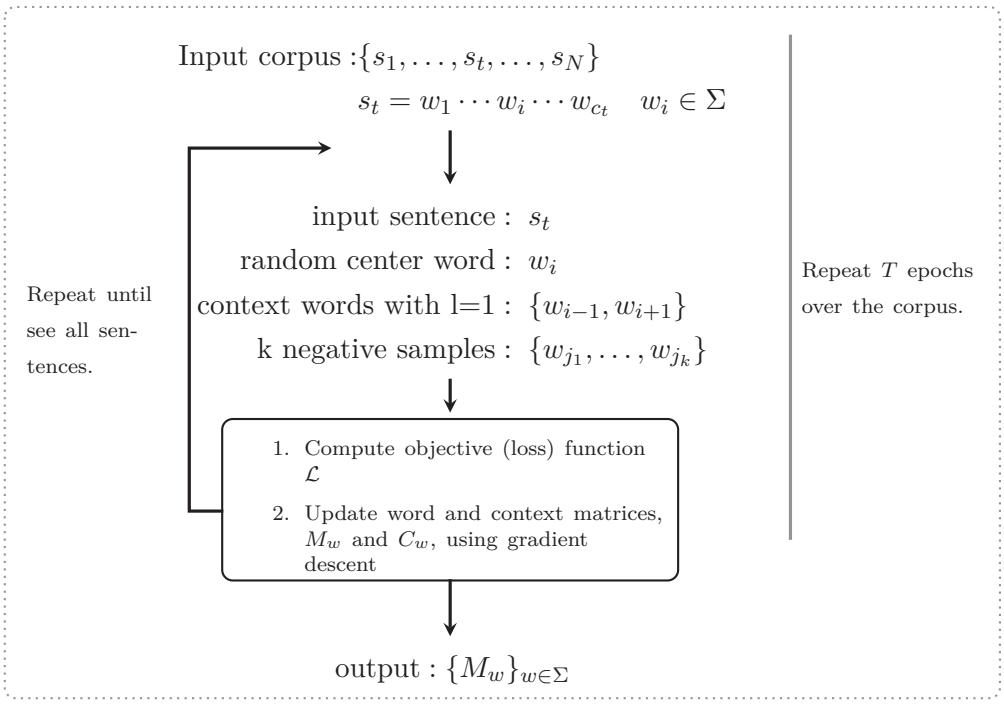


Figure 5.3: Learning procedure for COSMo. Each epoch of training is a pass over all sentences in the corpus. After T epochs $\{M_w\}_{w \in \Sigma}$ are extracted as the learned word representations.

We can denote all word and context matrices with a $V \times d \times d$-dimensional tensors $\mathbf{T}$ and $\mathbf{T}'$; that is, each slice of the tensors is a $d \times d$-dimensional matrix corresponding to a word in the vocabulary. Fig. 5.4 shows our feedforward two-layer NN, which, as opposed to skip-gram, consists of weight tensors instead of weight matrices. In this figure, we assume that the input to the training iteration is a center word w_i and w_{i+1} is the context word to predict, both are extracted from an input sentence s. The connections

in the input-to-hidden and hidden-to-output layers show the matrices corresponding to word and context matrices of w_i and w_{i+1}, respectively, which are extracted from the i-th slice of the corresponding tensors $\mathbf{T}$ and $\mathbf{T}'$.

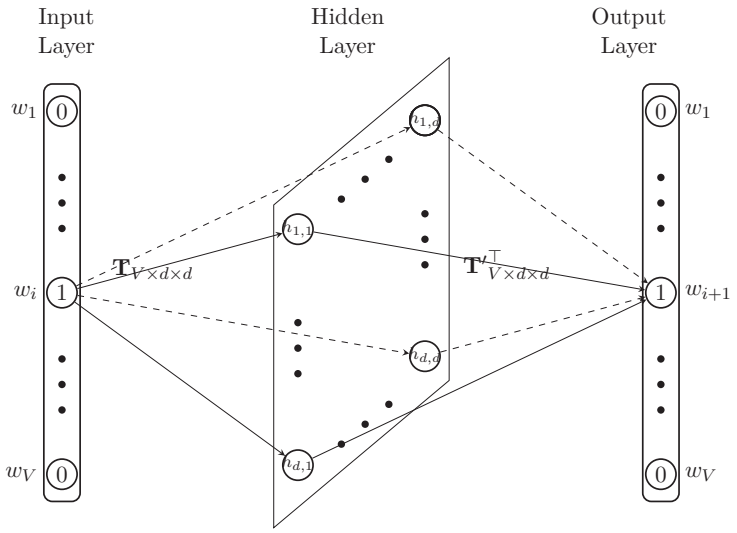


Figure 5.4: COSMo neural network architecture for an input word w_i to predict an output context word w_{i+1}. $\mathbf{T}$ and $\mathbf{T}'$ are tensors presenting the word and context matrices. The hidden layer is a matrix with dimensionality of size $d \times d$. V is the size of vocabulary. The input and output layers are one-hot encodings of the corresponding words.

5.2.4 COSMo versus skip-gram

In skip-gram, the composition of center and context words for training is based on the element-wise vector multiplication, which is not sensitive to word order in the sentences. In other words, skip-gram discards the position of the context word against the center word during the training of vector embeddings. In a target space $\mathbb{S}$ of matrices in $\mathbb{R}^{d\times d}$ the composition function in CMSMs requires ordered sequences. It ensures that the composition of two matrices $M_i \in \mathbb{R}^{d\times d}$ and $M_j \in \mathbb{R}^{d\times d}$ in this order M_iM_j differs from its reverse order M_jM_i. COSMo introduces order-sensitivity during training by utilizing matrix multiplication when composing the center and context word matrices, as shown in Equation 5.4. The same argumentation holds for composing center words with negative samples. Therefore, symmetric and asymmetric learning methods are introduced in COSMo.

5.2.5 COSMo versus CMOW

Relatively few matrix-space models have been proposed, especially on a word- or phrase-level representation. A common approach has been the combination of matrix and vector space representations by Socher et al. (2013), as described in more detail in related work of Chapter 3, generally trained on a specific task. To the best of our knowledge, the only

other approach to train task-agnostic matrix representations is that of Mai et al. (2019), called CMOW. However, there are significant differences between CMOW and COSMo, which clearly distinguish the two, which are:

- While CMOW learns a sentence representation in matrix format by adapting CBOW from the word2vec method, our objective is to learn a word-level matrix representation, which we tackle by proposing a new PMI-based CMSM method and a skip-gram adapted COSMo method. A word-level representation allows for dynamic composition of words to unseen phrases and sentences.
- As is shown in Equations 5.4 and 5.5, our training objective is based on CMSM, that is standard matrix multiplication is applied in the training process and objective function. In contrast, CMOW applies the objective function in Equation 5.6. In this equation, s is a sequence of words in a sentence from which an output word w_O is removed to be predicted by the model. $enc^E_\Delta(s)$ is the vectorized representation of the sequence of words s initially obtained from word matrix multiplication. The matrix of the output word w_O and random negative samples w_j are also vectorized. Then, element-wise vector multiplication is used as the composition operation to compose $\mathbf{v}_{w_O}$ and $\mathbf{v}_{w_j}$ with $enc^E_\Delta(s)$ during the training process. This goes against the original idea of CMSMs and relates more closely to the original word2vec-type of models.

$$\mathcal{L}_{\mathcal{CMOW}} = \log\ \sigma(\mathbf{v}^T_{w_O} enc^E_\Delta(s)) + \sum_{j=1}^{k} \mathbb{E}_{w_j \sim q(w_j)}[\ \log\ \sigma(-\mathbf{v}^T_{w_j} enc^E_\Delta(s))]. \tag{5.6}$$

5.3 Experiments

We evaluate the performance of our proposed word-level matrix embeddings in two ways. First, we evaluate them on five STS tasks from the SentEval repository (Conneau and Kiela, 2018). SentEval is a toolkit appropriate for evaluating the quality of representation models (word-level or sentence-level models), which consists of 17 downstream NLP (un)supervised tasks with the corresponding gold standard evaluation datasets. The downstream tasks include semantic textual similarity, natural language inference, sentiment classification, etc. The STS tasks are about predicting the similarity degree of the given pairs of sentences. Each task includes several subtasks. We report the average performance on all subtasks for each STS task. The STS tasks are unsupervised in SentEval[1] and are mainly used to evaluate the performance of representation models in capturing the meaning representation of sentences. Datasets used in these tasks consist of pairs of sentences labeled with a real-valued score between 0 and 5 as the similarity degree between the two sentences with 0 and 5 showing the least and the most similarity degrees. We use the STS tasks from SentEval for the evaluation of our matrix-embeddings to allow for comparability with existing approaches. The supervised tasks in SentEval, such as sentiment analysis, are outside the scope of this chapter

[1]Unsupervised task in SentEval refers to the task that only evaluates the representation models using a provided evaluation dataset, and it does not train any model within the SentEval to perform the task.

because their goal is to train a model to perform on these specific tasks. Therefore, we are not describing the supervised tasks.

Second, we use BiRD, introduced in Chapter 4, to evaluate our proposed matrix embeddings on capturing semantic relatedness degrees between pairs of short phrases and compare with the existing vector and matrix embeddings. In both tasks, word matrices are multiplied to achieve phrase and sentence representations. Vector addition and element-wise vector multiplication are used for word vectors to obtain phrase and sentence representations in those tasks, and they are reported separately.

5.3.1 Experimental Setup

Corpus:

We train our matrix embeddings on the UMBC WebBase Corpus[1] (Han et al., 2013), which contains about 134 million sentences and three billion words, to make our results comparable to the related work, Mai et al. (2019). For the same reason we fix the vocabulary to 30,000 most frequent words. No preprocessing to corpus or vocabulary is applied.

Hyperparameter Settings for the PMI-based CMSM:

For the PMI-based CMSM, the best results were obtained with matrix dimensions 20, that is, 20×20 matrices, learning rate 0.01, batch size 256, and 100 training epochs. One epoch always represents an entire pass over the whole training set. Note that $\boldsymbol{\beta}$ in Equation 5.3 is initialized with a normal distribution $\mathcal{N}(0, 0.1)$.

Hyperparameter Settings for the COSMo method:

For COSMo we experimented with three different kinds of context window sizes:

- Asymmetric training with three context words to the right of the center word (*COSMo-3asym*);
- Symmetric training with two context words to both sides of the center word (*COSMo-2sym*); and
- Symmetric with three context words to both sides of the center word (*COSMo-3sym*).

Experiments show the best performance at embedding size 20, that is, 20×20 matrices, with a learning rate of 0.003, and drawing 20 negative samples for each center word. We experimented with the two variations in the multiplication of center words with negative samples. As we did not observe a significant difference in the performance, we report the results with the first variation computed as in Equation 5.5. Three different values for the smoothing parameter α for negative sampling from a unigram probability distribution were tested, that is, 0.25, 0.50, and 0.75 (the last being the best value for skip-gram embeddings reported in (Mikolov et al., 2013b)). For symmetric models, a smoothing parameter of 0.50 performed best, while asymmetric models showed a preference for 0.25. When testing the number of sentences to be utilized per batch, symmetric models showed

[1] https://ebiquity.umbc.edu/resource/html/id/351

a preference for larger number of sentences, 1024 for symmetric and 256 sentences for asymmetric models. Each model was trained for a minimum of 20 epochs, stopping when improvements from one epoch to the next started dropping to below a threshold value $\epsilon = 0.10$. For COSMo-2sym, no improvement after 20 epochs could be observed. For COSMo-3sym and COSMo-3asym, this point was reached after 25 epochs. A detailed hyperparameter study is also provided in the following.

5.3.2 Results on Semantic Textual Similarity

Performance results of all proposed matrix embeddings on STS tasks of SentEval (Conneau and Kiela, 2018) are reported in comparison to the two baseline matrix-space models: (1) CMOW (Mai et al., 2019) with matrix embedding size of 28×28, and (2) LMS-TreeLSTM (Chung et al., 2018) with matrix embedding size of 18×18. We also compare them with the vector space skip-gram model using a vector embedding size of 300. Mai et al. (2019) report their model trained on the UMBC corpus. For a fair comparison, we trained our proposed models, the skip-gram model, as well as LMS-TreeLSTM on the same corpus (UMBC) and with the same vocabulary. We utilized the corresponding vectors trained on UMBC and input them to the method of Chung et al. (2018) for a fair comparison. For CMOW model, we took the results from their paper and did not attempt to reproduce the experiments. Performance is measured by the Spearman correlation since Mai et al. (2019) report their results as Spearman rather than Pearson values. Spearman measures the degree of relationship between two variables. While SentEval provides supervised and unsupervised tasks, our main interest here is on the capability of word-level matrices to compose utilizing matrix multiplication to represent sentence meanings. Therefore, we use five STS unsupervised tasks providing different sentence lengths, where the first three tasks in Table 5.1 generally have shorter sequences, and the last two tasks comparatively longer sentences.

The first two lines in Table 5.1 reports the results achieved by CMOW and LMS-TreeLSTM baseline models. The subsequent three lines show the results of the proposed COSMo models, the first two with a symmetric window size of 2 (COSMo-2sym) and 3 (Cosmo-3sym) respectively, while the third line provides the results for training COSMo with a window size of three to the right only (COSMo-3asym). Line six shows the results for the PMI-based CMSM learning method. Finally, the last two lines in Table 5.1 represent the output of vector embeddings obtained from the same corpus with the original skip-gram method and two different vector composition operations to obtain sentence embeddings: vector addition and element-wise multiplication.

Regarding the CMOW baseline matrices, the proposed learning methods obtain competitive results on all tasks. LMS-TreeLSTM performs poorly, and our methods outperform LMS-TreeLSTM significantly on all tasks. On the first three STS tasks, our trained matrices outperform the baseline matrices that were trained as sentence embeddings (CMOW). For tasks STS15 and STS16 the CMOW sentence embeddings achieve better performance. The task STS16 tends to have comparatively longer sentences than the others on average. This seems to constitute a problem for our approach of multiplying resulting word matrices to sentence matrices since our models perform worst on this task. On closer analysis, the most devastating task was that of STS16 question–question subtask without which our model could achieve a Spearman value

Line	Method \ Task	STS12	STS13	STS14	STS15	STS16
	Baseline					
1	CMOW (Mai et al., 2019)	39.2	31.9	38.7	**49.7**	**52.2**
2	LMS-TreeLSTM (Chung et al., 2018)	21.8	5.2	3.8	9.6	6.6
	Proposed models					
3	COSMo-2sym	40.1	32.2	40.7	44.6	48.1
4	COSMo-3sym	39.4	30.4	40.7	42.1	48.0
5	COSMo-3asym	**40.5**	**33.4**	**41.2**	44.2	46.5
6	PMI-based CMSM	32.7	26.0	36.8	40.3	43.5
	VSM					
7	skip-gram (addition)	47.7	50.6	53.0	58.9	54.6
8	skip-gram (multiplication)	10.7	8.6	7.2	7.6	10.7

Table 5.1: Spearman correlation value (times 100) of methods on STS unsupervised tasks of SentEval.

(times 100) of up to 58.0. On closer inspection it turns out that this question–question subtask of STS16 has the most varied vocabulary and by far the highest number of questions (naturally), which we believe to have been underrepresented in the training corpus. This disfavors a method that composes resulting word matrices to sentence matrices.

While these results are promising for CMSMs in general, an outperformance of VSMs would be desirable, and CMSMs may achieve better performance with another learning method entirely independent of VSM-inspired ones.

Comparing the results of our learning methods suggests that overall COSMo performs better than PMI-based CMSM does. As regards COSMo, only marginal differences in terms of symmetry and size of the context window could be observed. Nevertheless, a slight improvement could be observed with an asymmetric window size of three, that is, training with three context words to the right only. Similar performance of the asymmetric model to the symmetric model can be due to the fact that the composition operation utilized during the training is order sensitive, which considers symmetric properties implicitly.

Hyperparameter Testing

It has been shown that small adaptations in hyperparameter settings tend to impact final training outcomes with VSMs and neural networks strongly. It is good practice to evaluate model behaviors with tests of different hyperparameter combinations experimentally. This subsection presents the hyperparameter testing for the proposed models in terms of matrix sizes (embedding size), number of center words in each batch, number of negative samples randomly drawn for each center word, and the training batch size. Performance is measured in terms of average Spearman value on five STS tasks of SentEval. Additional tests regarding the smoothing parameter for drawing negative samples (α=0.5 for COSMo-sym and α=0.25 for COSMo-asym), learning rate (0.003 for all COSMo models), and different optimizers (gradient descent for all models) were performed.

Past approaches on vector- and matrix-space models have shown the central importance of embedding sizes (see Section 5.1). Fig. 5.5 showcases the different performances of our models with different embedding matrix sizes. Starting from 15×15 size matrices we experimented up to size 34×34, with a clear peak performance of size 20×20 for symmetric and asymmetric COSMo models as well as the PMI-based CMSM. For these tests, we utilized a context window size of two for the symmetric model (COSMo-2sym) and of three for the asymmetric model (COSMo-3asym). While the performance of the asymmetric model seems comparatively stable across embedding sizes with a marginal improvement on of size of 20×20, the symmetric COSMo model shows a clear preference for this size and a more considerable drop in performance with higher dimensionalities. To compare across models, the reported tests kept all other hyperparameters than the one tested fixed and identical for all COSMo models; that is, the same batch size (1024 sentences), number of negative samples (20), epochs (5), number of center words (30), learning rate (0.003). The only variation is the smoothing parameter α with a value of 0.50 for COSMo-2sym and 0.25 as a more successful value for COSMo-3asym.

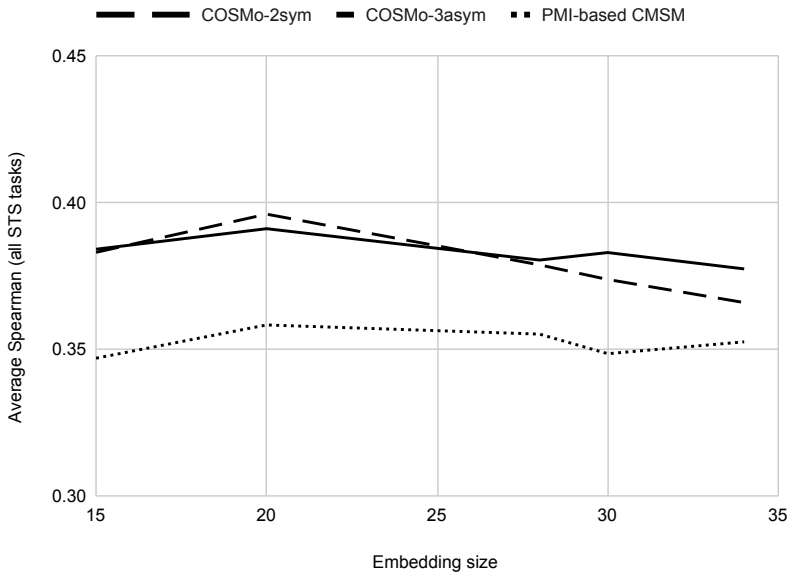


Figure 5.5: Performance comparison of embedding size in the COSMo-2sym and COSMo-3asym models as the most successful ones and the PMI-based CMSM. Hyperparameters: number of center words = 30, number of negative samples = 20, batch size = 1024 sentences, learning rate = 0.003, optimizer = gradient descent, number of epochs = 5.

A second high-impact hyperparameter in the COSMo learning method is the decision on how often a randomly selected center word is drawn from a given sentence in the training process, called the number of *skips*. Fig. 5.6 shows the results of our tests, where we compared three COSMo models and no PMI-based CMSM model since this is not a hyperparameter of the PMI-based CMSM method. For symmetric versions of the COSMo model, the number of skips should not be set any lower than 30. For the

asymmetric model already a number of skips of 25 starts to perform well.

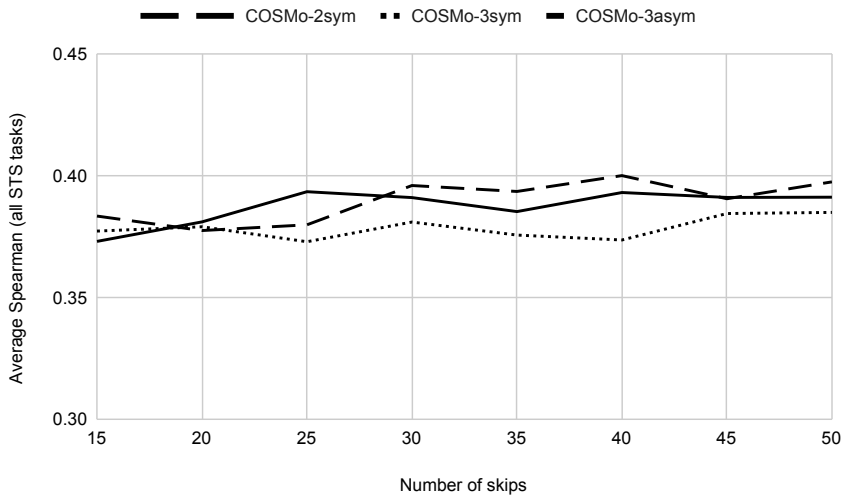


Figure 5.6: Performance comparison of the number of skips (center words) in the COSMo-2sym, COSMo-3sym, and COSMo-3asym models. Hyperparameters: embedding size = 20, number of negative samples = 20, batch size = 1024 sentences, learning rate = 0.003, optimizer = Gradient descent, number of epochs = 5.

COSMo models also require negative samples drawn from a uniform probability distribution, the number of which needs to be determined before training. We experimented with both variations in the multiplication of center words with negative samples. As we did not observe a significant difference in their performance, we report on the first variation computed as in Equation 5.5. Fig. 5.7 shows that the impact of this hyperparameter is relatively low with a rather flat curve for all three tested models. Only for COSMo-2sym, 20 negative samples seems better than a higher number, while for the other models the performance is almost the same across different settings of these hyperparameters. Again, all other hyperparameters have been fixed and identical with a difference of $\alpha = 0.50$ for symmetric and $\alpha = 0.25$ for asymmetric context models.

One hyperparameter that strongly depends on the training corpus for its fine-tuning is that of the batch size (i.e., the number of input sentences), and there seems to be a correlation between batch size and context window configuration. Table 5.2 shows a tendency for the symmetric COSMo model of window size of two (COSMo-2sym) to perform best with a lower dimensionality (dim = 20) and a larger batch size of 1024. For the asymmetric window size of three (COSMo-3asym), a smaller batch size of around 256 or 384 depending on the dimensionality seems to work better. We experimentally tested the learning rate and optimizers and concluded that 0.003 with gradient descent performed best in all experiments. The best overall performance in these hyperparameter tests could be achieved with COSMo-2sym of matrix size 20×20 and a batch size of 1024.

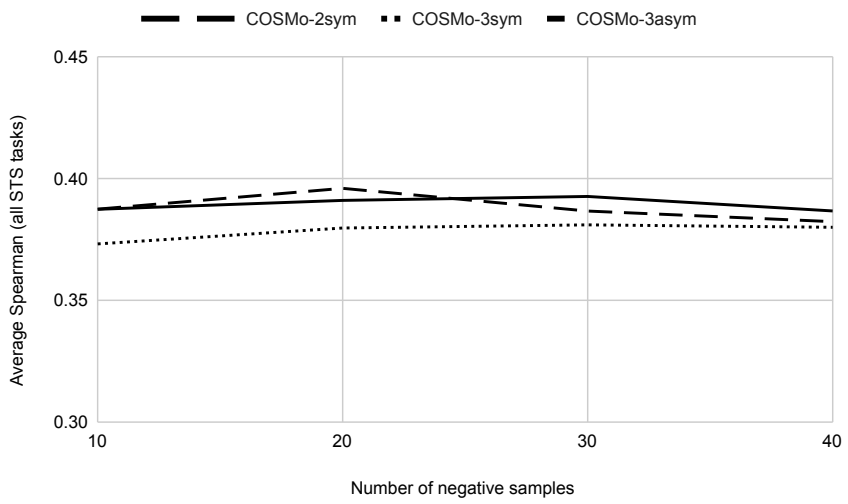


Figure 5.7: Performance comparison of the number of negative samples utilized to train the COSMo-2sym, COSMo-3sym, and COSMo-3asym models. Hyperparameters: embedding size = 20, number of skips = 30, batch size = 1024 sentences, learning rate = 0.003, optimizer = gradient descent, number of epochs = 5.

Method \ Batch size	128	256	384	512	640	1024
COSMo-2sym 20 × 20	37.05	35.10	38.69	39.09	38.94	**39.61**
COSMo-2sym 28 × 28	38.16	37.76	37.53	**38.41**	38.20	37.88
COSMo-3asym 20 × 20	37.11	**38.83**	38.73	38.54	38.25	38.03
COSMo-3asym 28 × 28	37.24	37.44	**38.15**	37.98	37.51	38.04

Table 5.2: Spearman value (times 100) of COSMo method with different batch sizes. Hyperparameters: $\alpha = 0.50$, negative samples = 20, learning rate = 0.003, optimizer= gradient descent, epoch number = 5. Best result among the batch sizes in each method is in bold.

Matrix Initialization

When training word vectors, the way of initializing the vectors has shown to have a significant impact on the final result of the trained embeddings. To this end, we have decided to try several methods for initializing our matrices on the COSMo-2sym model. Completely random initialization in this model led to disastrous results, which is why we resorted to testing various approaches of initializations close to the identity matrix. Each matrix is initialized with the identity matrix with the addition of some noise. The difference lies in how this noise is drawn from a normal distribution. We tested the initialization within a range of $[-\frac{0.5}{d}, \frac{0.5}{d}]$, where d represents the matrix dimensionality, as well as with the normal distribution of $\mathcal{N}\{0, 0.001\}$, both of which had proven successful for word embeddings. In comparison, we also report on the initialization with identity

matrix plus noise from $\mathcal{N}\{0, 0.1\}$, which we chose as an initialization method in the end. As can be seen from Table 5.3, the last setting outperforms the other two initialization methods.

Method \ Task	STS12	STS13	STS14	STS15	STS16
$\mathbb{I} + [-\frac{0.5}{d}, \frac{0.5}{d}]$	20.96	8.78	13.56	22.96	33.44
$\mathbb{I} + \mathcal{N}\{0, 0.001\}$	19.43	4.84	6.59	13.94	2.52
$\mathbb{I} + \mathcal{N}\{0, 0.1\}$	**38.73**	**29.30**	**39.80**	**43.96**	**46.27**

Table 5.3: Spearman value (times 100) of methods on STS tasks with different initializations of the COSMo-2sym model matrices after five epochs of training.

5.3.3 Results on Semantic Relatedness

A common approach to test representation models on semantic representations is exploiting their ability to rank pairs of terms (words or phrases) by closeness in meaning. Semantic relatedness is a measure to compute the meaning closeness of two terms. The BiRD dataset introduced in Chapter 4 (Asaadi et al., 2019) provides pairs of terms labeled with semantic relatedness scores. One term in each pair is a bigram and the other term is either a bigram or a unigram. In this section, we evaluate COSMo models as our most successful models, and the LMS-treeLSTM (Chung et al., 2018) and skip-gram baseline models on their ability to semantically represent words as well as phrases with different composition operations utilizing the BiRD dataset. The composition operation in matrix space is the standard matrix multiplication, and in the vector space are addition and element-wise multiplication; these approaches are reported separately. As our trained vocabulary contains $30,000$ words, some words in the dataset are missing, and therefore, we evaluate the models on 3,165 pairs from the BiRD. Table 5.4 shows the results of our introduced embeddings compared with the skip-gram and the LMS-treeLSTM embeddings.

Method	Pearson (r)
COSMo-2sym	0.327
COSMo-3sym	0.306
COSMo-3asym	0.357
skip-gram (Mikolov et al., 2013b) (multiplication)	0.378
skip-gram (Mikolov et al., 2013b) (addition)	0.587
LMS-treeLSTM (Chung et al., 2018)	0.161

Table 5.4: Pearson correlation value of methods on capturing semantic relatedness of term pairs in BiRD.

The results show that our matrices outperform LMS-treeLSTM, trained on the task of natural language inference estimation. The results of our individual learning methods suggest that COSMo-3asym performs the best. It also shows a very close performance

to the vector multiplication. As reported in the STS task, vector addition outperforms all other models.

5.4 Discussion and Conclusion

To the best of our knowledge, we are the first to propose a learning method to obtain word-level matrix embeddings that truly utilizes matrix multiplication (see Section 5.2.5 for a detailed comparison to the most closely related previous approach). In this chapter, we proposed two methods with several variations to learn word-level matrix embeddings. The first method is based on the linear regression that utilizes PMI values for bigrams in training word matrices, and the second method is an adaptation of the skip-gram method, which leads to a better performance than the first method when evaluated on the STS tasks. The learned matrix embeddings exhibit slightly different behaviors than the task-agnostic sentence-level embedding CMOW proposed by Mai et al. (2019) and the task-specific matrix learning method proposed by Chung et al. (2018). These differences become evident when comparing matrix dimensionality.

In terms of matrix dimensionality, previous studies (Irsoy and Cardie, 2015; Chung et al., 2018) observed an improvement of performance with an increase in matrix dimensionality, which also corresponds to vector space findings. The task-agnostic sentence-level model CMOW (Mai et al., 2019) equally reported improved results with an increase in dimensionality, where their final size reported is 28×28 matrices. All of our models, symmetric and asymmetric COSMo, as well as PMI-based CMSM, peaked in performance at a dimensionality of size 20×20, which is lower than CMOW and still performs competitively. Moreover, our models outperformed LMS-treeLSTM with the dimensionality of size 18×18, which is not significantly smaller than our models' dimensionality.

In terms of context window, asymmetric context window in our case means contexts to the right only, since English composes words from left to right rather than from right to left. However, also for VSMs with bag-of-words contexts (i.e., order-insensitive contexts), it has been shown (Lison and Kutuzov, 2017) that asymmetric contexts to the right are on par with symmetric ones, while asymmetric contexts to the left perform worse. Our findings also confirm that the symmetric and asymmetric (right) models in matrix space generate matrices that are almost on par in performance on STS tasks.

Since we propose matrix models based on multiplicative composition, one of the most central aspects is the discussion on additive and multiplicative composition in VSMs in comparison to matrix models. In the task of compositional semantic similarity, Turney (2012) showed that addition outperforms multiplication, which could also be confirmed for later learning methods for embeddings. However, those operations are naive in the sense that they assign the same representations to phrases containing the same words irrespective of the word order. Matrix multiplication with its sensitivity to order, and thus, also sensitivity to word order when trained on natural language, holds the promise to improve on this naivety and provide a semantic composition method. Our results show that matrix multiplication outperforms vector multiplication; however, vector addition performs considerably better than both. The same phenomenon could be observed for the sentence-level matrix embedding CMOW (Mai et al., 2019), which is also outperformed

by vector addition.

Having conducted extensive experiments on the proposed learning methods for matrix-space models, we conclude that their performance is competitive with VSMs. However, this finding shows that our expectation of a richer semantic composition operation leading to an improvement of performance on composition tasks could not be met. Our intuition is that an entirely different learning method for matrix-space models far from the ideas of VSMs (e.g., skip-gram), may be a viable research path for the future.

The learning technique for PMI-based CMSM utilizes PMI values for training word matrices, which present global information about the association between words co-occurring in a given text corpus. The COSMo is trained based on the distributional hypothesis, which provides local information about the association between words. A future direction is combining both techniques to propose a joint learning method employing both local and global information for training word matrices. Note that the two methods define different objective functions, and therefore, the error values to minimize have different scales. Thus, suitable methods of rescaling the error values of the learning techniques are needed for joint training.

Chapter 6

Conclusions and Outlook

In this dissertation, we investigated CMSMs as alternative word representation models in NLP. In these models, matrix space, as opposed to vector space, is used to reflect the meaning of words in real-valued quadratic matrix representations. The compositional representations of the larger linguistic units, such as phrases, are obtained utilizing matrix multiplication of word matrices in the same order as they appear in the phrase. We focused on the two following research directions in this dissertation:

1. Machine learning approaches for learning CMSMs as representation models in NLP; and
2. Evaluation approaches for examining the capability of the proposed and existing representation models on meaning composition in NLP.

Chapter 3 presented experimental investigation of CMSMs in sentiment analysis and compositionality detection tasks in NLP. Focusing on the first research direction, supervised learning techniques were proposed in both tasks to train word matrices in CMSMs from available training datasets.

In sentiment analysis, word matrices were trained using linear regression to contain semantic and sentiment-related information. The novelty of our learning method consists in a two-step learning procedure, called gradual learning, where the result of the first step is used as initialization for the second step. We identified gradient descent as the most efficient optimizer in our learning technique.

The results of the experiments showed the outperformance of the proposed approach over existing approaches for learning CMSMs (Yessenalina and Cardie, 2011; Irsoy and Cardie, 2015) in predicting the sentiment scores of compositional phrases. The results also demonstrated that the matrix-based compositionality operation shows sensitivity to word order, arguably reflecting the sentiment scores of phrases better than any commutative operation could. In general, our findings confirm those of Yessenalina and Cardie (2011): Adverbs and negators in natural language play an important role in determining the sentiment scores of phrases. The results in the sentiment analysis task showed that multiplicative interaction in CMSMs captures the effect of adverbs and negators on the sentiment score when composed with a phrase. For instance, the word “very” seems to intensify the sentiment score of the subsequent phrase, while “not” not only flips the sentiment of the phrase syntactically following it but also gradually the sentiment of the phrases. In contrast, the scores of phrases containing “not very” defy the assumption that the described effects of these operators can be combined in a straightforward way.

The proposed learning technique can also capture the sentiment scores of phrases that contain words with opposing polarities. For instance, the phrase “happy tears” contains a positive word (happy) and a negative word (tear), while the overall sentiment of the whole phrase is positive. We compared the performance of our approach for this type of phrases with SVR in VSMs. The results showed the outperformance of our model over SVR with vector averaging as the composition operation in VSMs; however, it could not outperform SVR with vector concatenation as the composition operation. Overall, our learning approach for CMSMs achieved competitive performance to the SVR approach in VSMs.

Since the CMSM optimization problem in matrix space is a non-convex problem, attention to the starting point to avoid local optima is important. To accomplish this,

different matrix initialization methods were studied in this chapter, and the results showed that identity initialization of matrices with noise values on all matrix elements achieves the best performance.

In the compositionality detection task, we investigated CMSMs' ability to identify (non-)compositional short phrases—in this case bigrams, that is, two-word sequences. For this purpose, we first proposed a learning technique for CMSMs based on linear regression that was trained to produce compositional representations of bigrams from their words' representations. Then, we used gold standard evaluation datasets consisting of bigrams with an associated real-valued compositionality degree, and we investigated whether CMSMs could detect the compositionality degree of the bigrams. The performance of CMSMs in compositionality detection was compared with that of various compositional models in vector space. These models included unsupervised compositional models, that is, vector addition and element-wise multiplication on word vector embeddings (i.e., fastText and word2vec), and supervised models, that is, feedforward neural network, polynomial regression, and RNN. The results showed an outperformance of CMSM over vector multiplication and feedforward NN, and competitive performance to vector addition, polynomial regression, and RNN on the compositionality detection of bigrams. One of the gold standard evaluation datasets had more fine-grained compositionality degrees for noun–noun and adjective–noun compounds. We observed that the CMSMs tend to be more accurate than other models in capturing more fine-grained values and predicting the compositionality of adjective–noun compounds better than the studied compositional models.

One immediate advantage of employing CMSMs in the two NLP tasks was that matrix multiplication is an operation that is natural, plausible on several levels, and word-order sensitive. Experiments on both tasks showed that there are scalable learning methods for training CMSMs for downstream NLP tasks that require significantly fewer training parameters. Thus, CMSMs are capable of embedding relevant information in considerably fewer dimensions compared with VSMs, which gives a clear advantage in terms of computational cost, storage, and convergence in learning. Overall, experimental investigations suggest that CMSMs are a promising framework to model task-specific semantic compositionality processes, such as compositional sentiment analysis and compositionality detection of short sequences.

We are aware that experiments have been only done on short length sequences, and further investigation is needed for examining CMSMs on longer sequences, such as sentences. Matrix multiplication on long sequences can cause the final matrix to contain extremely small values, which affects the updating process of word matrices in the gradient descent; that is, word matrix values may not be updated adequately. Therefore, mechanisms are needed to avoid this issue when training CMSMs on long sequences. Moreover, when CMSMs are trained on long sequences in a specific task, such as, sentiment analysis, not all words contain task-specific information. A method is needed to pay attention and give more weights to those words that carry the relevant information, for instance, sentiment-carrying words in sentiment analysis.

Chapter 4 focused on the second research direction, which involved evaluating semantic composition methods. A common evaluation approach exploits the ability of these methods to rank pairs of terms (e.g., phrases) by their relatedness in meaning. Two terms

are semantically related if they belong to any classical or non-classical lexical–semantic relations. Gold standard semantic relatedness datasets are useful for evaluating semantic composition. Existing semantic relatedness datasets focus on single words, and therefore, they cannot be used to evaluate semantic composition. Moreover, existing datasets suffer from shortcomings due to the annotation techniques employed. To address these issues, this chapter introduced BiRD, a new gold standard bigram semantic relatedness dataset. BiRD consists of pairs of terms (i.e., bigram–bigram and bigram–unigrams) with an associated semantic relatedness score obtained from human annotations. Each bigram occurs in about eight distinct pairs in BiRD. This is yet another aspect that makes BiRD unique, as existing datasets were not designed to include terms in multiple pairs.

A comparative annotation technique, BWS (Louviere, 1991; Cohen, 2003; Louviere et al., 2015; Kiritchenko and Mohammad, 2017), was employed. This approach addresses common issues of previous annotation techniques, such as inconsistencies in annotations by different annotators, inconsistencies in annotations by the same annotator at different times, bias in the scale region (annotators often have a bias toward a portion of the scale, usually toward the middle), and problems associated with a fixed granularity (an annotator may want to assign 1.5 to an item instead of 1 or 2; Presser and Schuman, 1996). We then assessed the reproducibility and quality of the annotations using the SHR technique (Cronbach, 1951), which showed high reliability of the annotations.

Benchmark experiments were done on using BiRD for evaluating various composition methods used to obtain meaning composition from different distributional word representation models. The underlying assumption was that the more accurately a method of semantic composition can determine the representation of a bigram, the more accurately it can determine the relatedness of that bigram with other terms. Results on different composition methods in vector space showed that vector addition performs best among different composition methods when tested using different representation models. Among the different representation models, ELMo and fastText competitively outperformed the other models. That is, ELMo (Peters et al., 2018) and fastText (Bojanowski et al., 2017), with addition as the composition operation, performed better than other representation models in capturing the meaning composition, and consequently, the semantic relatedness of term pairs in BiRD.

Since the dataset was created using crowdsourcing, we also studied how humans perceive the relatedness between terms with different types of lexical–semantic relations. We analyzed the dataset to obtain insights into the distributions of semantic relatedness scores for pairs associated through various relations. These included pairs with lexical–semantic relations (e.g., hyponyms, meronyms, and synonyms), transposed bigram pairs (AB–BA, e.g., traffic light–light traffic), and co-aligned term pairs extracted from phrase tables in statistical machine translation. We observed that co-aligned terms from phrase tables can be related to varying degrees (from unrelated to synonymous), making them a useful source of term pairs to include in semantic relatedness datasets. Moreover, the semantic relatedness scores of AB–BA pairs varied from 0.25 to 0.90. Those pairs with low relatedness scores, such as law school–school law and traffic light–light traffic, were especially useful in testing whether compositional representation models generate suitably different representations for the terms in such pairs.

The introduced dataset provides a quantification of semantic relatedness, and its applicability is broader than what we investigated in this chapter. It can be used for the

following purposes: (1) evaluating the large number of proposed supervised methods of semantic composition, such as LSTM-based methods (Zhu et al., 2016); (2) evaluating the different measures of semantic relatedness, such as those introduced by Budanitsky and Hirst (2006); and (3) for evaluations in real-word spelling correction (Hirst and Budanitsky, 2005) and textual entailment (Mirkin et al., 2006) tasks, which often require judgments of the form '*is* AB–X_1 *more related or less related than* AB–X_2'.

Finally, Chapter 5 introduced CMSMs as generic word representation models in matrix space (also called word matrix embeddings). The introduced models were task-agnostic representation models as they were not trained on any task to capture task-specific information. Word matrices were trained using distributional information of words based on the distributional hypothesis (Harris, 1954) and PMI between words in a given text corpus. The embeddings provide continuous word representations in natural language and reflect the semantic relationships between words.

We proposed two learning techniques to obtain word matrix embeddings in this chapter. The first technique, called PMI-based CMSM, was a supervised learning technique based on linear regression. The second technique, called COSMo, was a self-supervised method of training a feedforward two-layer neural network.

To investigate the proposed embeddings on semantic representation and compositionality, we evaluated them using the two following evaluation tasks: (1) STS and (2) semantic relatedness. The performance of the proposed matrix embeddings was then compared with that of two existing word matrix representation models, LMS-treeLSTM (Chung et al., 2018) and CMOW (Mai et al., 2019), which showed outperformance over the first and competitive performance with the second model.

In terms of matrix dimensionality, our matrix embedding size in all the experiments was set to 20×20 as the best configuration, which was lower than that of CMOW (28×28) and still performed competitively. Moreover, our models outperformed LMS-treeLSTM with dimensionality of 18×18, which is not significantly smaller than our models' dimensionality. Furthermore, we compared the performance of our matrix embeddings with skip-gram word vector embeddings (Mikolov et al., 2013b), using an embedding size of 300. The composition operations in VSMs were based on the addition and element-wise multiplication reported separately. Vector multiplication on skip-gram embeddings demonstrated poor performance and did not outperform our models in capturing meaning composition, while vector addition outperformed all models. Since COSMo was an adaptation of the skip-gram from vector space to matrix space, our intuition is that an entirely different training method for matrix-space models, far from the ideas of VSMs, may be a viable path for the future.

The learning technique for PMI-based CMSM utilized PMI values for training word matrices, which present global information about the association between words co-occurring in a given text corpus. COSMo was trained based on the distributional hypothesis, which provides local information about the association between words. The experiments suggested that COSMo performs better than PMI-based CMSMs do overall in capturing semantic representations. However, to take both local and global information into account when training word matrix representation models, a future direction is combining the two learning techniques to propose a joint method employing all the information. The two techniques define different objective functions, and therefore, the

error values to minimize have different scales. Thus, suitable methods of rescaling the error values of the learning techniques are needed for joint training.

A shortcoming of the proposed matrix embeddings is that one representation is assigned to each word. However, a word may have different meanings based on the context it occurs in. Recent approaches, such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018), train distinct representations for different meanings of the words based on the context; these are called contextualized embeddings (or context-aware embeddings). Learning context-aware word matrix embeddings is an interesting research direction in this area.

The extensive experiments on vector- and matrix-space models conducted in this dissertation have shown the superiority of matrix multiplication as an order-sensitive composition operation over vector addition or element-wise multiplication in NLP tasks. Moreover, CMSMs do not rely on parse trees, and therefore, preprocessing of texts in the training datasets is not required. Each word is represented only with matrices, where the composition function is the standard matrix multiplication that replaces the recursive computations in parse trees with a sequential computation.

As discussed in Chapter 2.3, matrix multiplication is an associative linear operation which is generally unable to disambiguate the different meanings of a sentence. CMSMs assign the same representation to several interpretations of a sentence. An approach to addressing the ambiguity issue is establishing methods that help the NLP systems to understand the correct meaning of a sentence based on its context. These methods train distinct representations for different meanings of the words based on the context, which are called contextualized embeddings (or context-aware embeddings). Therefore, learning approaches to train the context-aware matrix representation models can be a potential solution to this problem.

A central question related to CMSMs is how the linearity of matrix multiplications limits their applicability. In Chapter 2.3, we studied how Rudolph and Giesbrecht (2010) justified CMSMs by showing that matrix multiplication can realize mental state transitions triggered by the sequence of input signals. Using linear mappings to represent those functions introduces limitations that need to be further investigated. Therefore, we must equip the CMSMs with nonlinear functions to resolve the problems caused by linearity. Thus, a line of further research is proposing deep learning approaches to obtain CMSMs, or augmenting CMSMs with the existing deep neural network architectures. Nonlinear approaches to CMSMs can also solve the problems caused by the associativity property, such as word sense disambiguation in natural language.

Overall, this thesis has demonstrated that CMSMs offer attractive theoretical features and practical behavior. This strongly suggests CMSMs can be used as a suitable framework of semantic compositionality in NLP downstream applications.

We hope that the investigations in this thesis help to extend CMSMs to address the mentioned shortcomings and allow the proposed word matrix representation models to be further improved and used extensively in NLP applications. In addition, we hope that the introduced gold standard bigram semantic relatedness dataset will foster further research in this area.

Bibliography

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paundefinedca and Aitor Soroa (2009). "A Study on Similarity and Relatedness Using Distributional and WordNet-Based Approaches". In: *Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Ed. by Ulrich Germann, Chirag Shah, Svetlana Stoyanchev, Carolyn Penstein Rosé and Anoop Sarkar. Boulder, Colorado: Association for Computational Linguistics, pp. 19–27 (cit. on pp. 6, 44, 82).

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau and Janyce Wiebe (2014). "SemEval-2014 Task 10: Multilingual Semantic Textual Similarity". In: *Proc. of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Ed. by Preslav Nakov and Torsten Zesch. Dublin, Ireland: Association for Computational Linguistics, pp. 81–91 (cit. on p. 99).

Cem Akkaya, Janyce Wiebe and Rada Mihalcea (2012). "Utilizing Semantic Composition in Distributional Semantic Models for Word Sense Discrimination and Word Sense Disambiguation". In: *2012 IEEE Sixth International Conference on Semantic Computing*. Ed. by Randall Bilof. IEEE. IEEE Computer Society, pp. 45–51 (cit. on p. 28).

Adrian Akmajian, Richard A. Demers, Ann K. Farmer and Robert M. Harnish (2010). *Linguistics: An Introduction to Language and Communication*. 6th ed. USA: The MIT Press (cit. on p. 2).

Shima Asaadi, Saif Mohammad and Svetlana Kiritchenko (2019). "Big BiRD: A Large, Fine-Grained, Bigram Relatedness Dataset for Examining Semantic Composition". In: *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (NAACL-HLT)*. Ed. by Jill Burstein, Christy Doran and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 505–516 (cit. on pp. 10, 103, 120).

Shima Asaadi and Sebastian Rudolph (2016). "On the Correspondence between Compositional Matrix-Space Models of Language and Weighted Automata". In: *Proc. of the SIGFSM Workshop on Statistical NLP and Weighted Automata (StatFSM 2016)*. Ed. by Bryan Jurish, Andreas Maletti, Kay-Michael Würzner and Uwe Springmann. Association for Computational Linguistics, pp. 70–74 (cit. on pp. 8, 37, 39).

Shima Asaadi and Sebastian Rudolph (2017). "Gradual Learning of Matrix-Space Models of Language for Sentiment Analysis". In: *Proc. of the 2nd Workshop on Representation*

Learning for NLP (RepL4NLP 2017). Ed. by Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston and Scott Yih. Association for Computational Linguistics, pp. 178–185 (cit. on pp. 9, 52, 106).

Alan D. Baddeley and Graham Hitch (1974). "Working Memory". In: *Psychology of Learning and Motivation*. Ed. by Gordon H. Bower. Vol. 8. Academic Press, pp. 47–89 (cit. on p. 32).

Timothy Baldwin and Su Nam Kim (2010). "Multiword Expressions". In: *Handbook of natural language processing*. Ed. by Nitin Indurkhya and Fred J. Damerau. Vol. 2. USA: CRC Press, pp. 267–292 (cit. on pp. 42, 68).

Borja Balle, William Hamilton and Joelle Pineau (2014). "Methods of Moments for Learning Stochastic Languages: Unified Presentation and Empirical Comparison". In: *Proc. of the 31st International Conference on Machine Learning (ICML 2014)*. Ed. by Eric P. Xing and Tony Jebara. JMLR: W&CP, pp. 1386–1394 (cit. on p. 40).

Borja Balle and Mehryar Mohri (2015). "Learning Weighted Automata". In: *Proc. of 6th International Conference on Algebraic Informatics (CAI 2015)*. Ed. by Andreas Maletti. Stuttgart, Germany: Springer International Publishing, pp. 1–21 (cit. on p. 40).

Marco Baroni and Alessandro Lenci (2010). "Distributional Memory: A General Framework for Corpus-Based Semantics". In: *Computational Linguistics* 36.4, pp. 673–721 (cit. on pp. 4–6, 42, 44).

Marco Baroni and Roberto Zamparelli (2010). "Nouns are Vectors, Adjectives are Matrices: Representing Adjective-noun constructions in semantic space". In: *Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. Ed. by Hang Li and Lluís Màrquez. Association for Computational Linguistics, pp. 1183–1193 (cit. on pp. 30, 92).

Jean Berstel Jr. and Christophe Reutenauer (1988). *Rational Series and Their Languages*. Berlin, Heidelberg: Springer-Verlag (cit. on p. 37).

Dimitri P. Berstsekas (1999). *Nonlinear Programming*. Athena Scientific (cit. on pp. 21, 53).

Chris Biemann and Eugenie Giesbrecht (2011a). "Distributional Semantics and Compositionality 2011: Shared Task Description and Results". In: *Proc. of the Workshop on Distributional Semantics and Compositionality (DiSCO 2011)*. Ed. by Chris Biemann and Eugenie Giesbrecht. Portland, Oregon, USA: Association for Computational Linguistics, pp. 21–28 (cit. on p. 77).

Chris Biemann and Eugenie Giesbrecht, eds. (2011b). *Proc. of the Workshop on Distributional Semantics and Compositionality*. Portland, Oregon, USA: Association for Computational Linguistics (cit. on p. 77).

Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146 (cit. on pp. 69, 73, 94, 126).

Gerlof Bouma (2009). "Normalized (Pointwise) Mutual Information in Collocation Extraction". In: *From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference*. Ed. by Christian Chiarcos, Richard Eckart de Castilho and Manfred Stede. Tübingen: Gunter Narr Verlag, pp. 31–40 (cit. on p. 107).

Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning (2015). "A Large Annotated Corpus for Learning Natural Language Inference". In: *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Lluís Màrquez, Chris Callison-Burch and Jian Su. Lisbon, Portugal: Association for Computational Linguistics, pp. 632–642 (cit. on p. 104).

Alexander Budanitsky and Graeme Hirst (2001). "Semantic Distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures". In: *Proc. of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*. Ed. by Sanda Harabagiu, Dan Moldovan, Wim Peters, Mark Stevenson and Yorick Wilks. Vol. 2. Association for Computational Linguistics, pp. 1–6 (cit. on pp. 6, 44, 46, 82).

Alexander Budanitsky and Graeme Hirst (2006). "Evaluating WordNet-based Measures of Lexical Semantic Relatedness". In: *Computational Linguistics* 32.1, pp. 13–47 (cit. on pp. 45, 46, 82, 94, 100, 127).

Maja Buljan, Sebastian Padó and Jan Šnajder (2018). "Lexical Substitution for Evaluating Compositional Distributional Models". In: *Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) (NAACL-HLT)*. Ed. by Marilyn Walker, Heng Ji and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, pp. 206–211 (cit. on p. 99).

Augustin Cauchy (1847). "Méthode générale pour la résolution des systemes déquations simultanées". In: *Comp. Rend. Sci. Paris* 25, pp. 536–538 (cit. on pp. 21, 53).

Daniel Cera, Mona Diab, Eneko Agirrec, Inigo Lopez-Gazpioc, Lucia Speciad and Basque Country Donostia (2017). "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation". In: *Proc. of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*. Ed. by Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer and David Jurgens. Vancouver, Canada: Association for Computational Linguistics, pp. 1–14 (cit. on p. 99).

WooJin Chung, Sheng-Fu Wang and Samuel Bowman (2018). "The Lifted Matrix-Space Model for Semantic Composition". In: *Proc. of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018)*. Ed. by Anna Korhonen and Ivan Titov. Association for Computational Linguistics, pp. 508–518 (cit. on pp. 10, 11, 103–105, 115, 116, 120, 121, 127).

Kenneth Ward Church and Patrick Hanks (1990). "Word Association Norms, Mutual Information, and Lexicography". In: *Computational linguistics* 16.1, pp. 22–29 (cit. on pp. 94, 107).

Steven H. Cohen (2003). *Maximum Difference Scaling: Improved Measures of Importance and Preference for Segmentation*. Sawtooth Software, Inc. (Cit. on pp. 9, 83, 87, 126).

Alexis Conneau and Douwe Kiela (2018). "SentEval: An Evaluation Toolkit for Universal Sentence Representations". In: *Proc. of the Eleventh International Conference on Language Resources and Evaluation (LREC)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis and Takenobu Tokunaga. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 1699–1704 (cit. on pp. 103, 113, 115).

Silvio Cordeiro, Carlos Ramisch, Marco Idiart and Aline Villavicencio (2016). "Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time". In: *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (ACL, Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 1986–1997 (cit. on p. 78).

Silvio Cordeiro, Aline Villavicencio, Marco Idiart and Carlos Ramisch (2019). "Unsupervised Compositionality Prediction of Nominal Compounds". In: *Computational Linguistics* 45.1, pp. 1–57 (cit. on p. 78).

M. J. Cresswell (1976). "Formal Philosophy; Selected Papers of Richard Montague". In: *Philosophia* 6.1, pp. 193–207 (cit. on pp. 3, 28).

Lee J. Cronbach (1951). "Coefficient Alpha and the Internal Structure of Tests". In: *Psychometrika* 16.3, pp. 297–334 (cit. on pp. 9, 83, 89, 126).

David A. Cruse (1986). *Lexical Semantics*. Cambridge University Press (cit. on pp. 6, 43, 82).

Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer and Richard Harshman (1990). "Indexing by Latent Semantic Analysis". In: *Journal of the American society for information science* 41.6, pp. 391–407 (cit. on pp. 17, 30).

Arthur P Dempster, Nan M Laird and Donald B Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–22 (cit. on p. 41).

Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (NAACL-HLT)*. Ed. by Jill Burstein, Christy Doran and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics (cit. on p. 128).

Michel Marie Deza and Elena Deza (2014). *Encyclopedia of Distances*. Springer (cit. on p. 46).

Philip Edmonds and Graeme Hirst (2002). "Near-Synonymy and Lexical Choice". In: *Computational linguistics* 28.2, pp. 105–144 (cit. on p. 43).

Samuel Eilenberg (1974). *Automata, Languages, and Machines*. USA: Academic press (cit. on pp. 36, 37, 39).

Jeffrey L Elman (1990). "Finding Structure in Time". In: *Cognitive science* 14.2, pp. 179–211 (cit. on p. 22).

Ramona Enache, Inari Listenmaa and Prasanth Kolachina (2014). "Handling Non-compositionality in Multilingual CNLs". In: *Controlled Natural Language*. Ed. by Brian Davis, Kaarel Kaljurand and Tobias Kuhn. Cham: Springer International Publishing, pp. 147–154 (cit. on pp. 42, 68).

Meghdad Farahmand, Aaron Smith and Joakim Nivre (2015). "A Multiword Expression Data Set: Annotating Non-Compositionality and Conventionalization for English Noun Compounds". In: *Proc. of the 11th Workshop on Multiword Expressions (MWE 2015)*. Ed. by Valia Kordoni, Kostadin Cholakov, Markus Egg, Stella Markantonatou and Shuly Wintner. Denver, Colorado: Association for Computational Linguistics, pp. 29–33 (cit. on pp. 42, 68, 74, 78).

Manaal Faruqui and Chris Dyer (2014). "Improving Vector Space Word Representations Using Multilingual Correlation". In: *Proc. of the 14th Conference of the European*

Chapter of the Association for Computational Linguistics (EACL). Ed. by Shuly Wintner, Sharon Goldwater and Stefan Riezler. Gothenburg, Sweden: Association for Computational Linguistics, pp. 462–471 (cit. on p. 92).

Christiane Fellbaum (1998). *WordNet: An Electronic Lexical Database*. MIT Press (cit. on p. 84).

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman and Eytan Ruppin (2002). "Placing Search in Context: The Concept Revisited". In: *ACM Trans. Inf. Syst.* 20.1, pp. 116–131. ISSN: 1046-8188 (cit. on pp. 7, 46, 82, 99).

Mark A. Finlayson and Nidhi Kulkarni (2011). "Detecting Multi-Word Expressions Improves Word Sense Disambiguation". In: *Proc. of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE 2011)*. Ed. by Valia Kordoni, Carlos Ramisch and Aline Villavicencio. Association for Computational Linguistics, pp. 20–24 (cit. on pp. 42, 68).

John R. Firth (1957). "A Synopsis of Linguistic Theory 1930-55". In: 1952-59, pp. 1–32 (cit. on p. 16).

John R. Firth (1962). "A Synopsis of Linguistic Theory 1930-55". In: *Studies in Linguistic Analysis* 1952-59, pp. 1–32 (cit. on p. 16).

T. N. Flynn and A. A. J. Marley (2014). "Best-Worst Scaling: Theory and Methods". In: *Handbook of Choice Modelling*. Ed. by Stephane Hess and Andrew Daly. Edward Elgar Publishing, pp. 178–201 (cit. on pp. 9, 89).

Amir Gandomi and Murtaza Haider (2015). "Beyond the Hype: Big Data Concepts, Methods, and Analytics". In: *International Journal of Information Management* 35.2, pp. 137–144 (cit. on p. 2).

Yoav Goldberg (2017). "Neural Network Methods for Natural Language Processing". In: *Synthesis Lectures on Human Language Technologies* 10.1, pp. 1–309 (cit. on pp. 2, 3, 15, 17, 19, 21, 25).

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press (cit. on pp. 20, 22).

Iryna Gurevych (2006). *Thinking beyond the Nouns-Computing Semantic Relatedness across Parts of Speech*. Tech. rep. Darmstadt University of Technology, Germany, Department of Computer Science, Telecooperation (cit. on p. 99).

Per-Kristian Halvorsen and William A. Ladusaw (1979). "Montague's 'Universal Grammar': An Introduction for the Linguist". In: *Linguistics and Philosophy* 3.2, pp. 185–223. URL: `http://www.jstor.org/stable/25001017` (cit. on pp. 3, 28).

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield and Jonathan Weese (2013). "UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems". In: *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proc. of the Main Conference and the Shared Task: Semantic Textual Similarity*. Ed. by Mona Diab, Tim Baldwin and Marco Baroni. Vol. 1. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 44–52 (cit. on p. 114).

Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi and Jacky Montmain (2015). "Semantic Similarity from Natural Language and Ontology Analysis". In: *Synthesis Lectures on Human Language Technologies* 8.1. Ed. by Graeme Hirst, pp. 1–254 (cit. on pp. 6, 45, 46).

David Harris and Sarah Harris (2007). *Digital Design and Computer Architecture (2nd Edition)*. San Francisco, Calif: Morgan Kaufmann (cit. on p. 16).

Zellig S. Harris (1954). "Distributional Structure". In: *Word* 10, pp. 146–162 (cit. on pp. 2, 4, 16, 18, 127).

Roland Hausser (2001). *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language (2nd Edition).* Berlin, New York:Springer Verlag (cit. on p. 2).

Felix Hill, Roi Reichart and Anna Korhonen (2015). "SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation". In: *Computational Linguistics* 41.4, pp. 665–695 (cit. on pp. 46, 82, 99).

G. E. Hinton, J. L. McClelland and D. E. Rumelhart (1986). "Distributed Representations". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1.* Cambridge, MA, USA: MIT Press, pp. 77–109 (cit. on p. 18).

Graeme Hirst and Alexander Budanitsky (2005). "Correcting Real-Word Spelling Errors by Restoring Lexical Cohesion". In: *Natural Language Engineering* 11.1, pp. 87–111 (cit. on pp. 84, 100, 127).

Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 103).

James Hong and Michael Fang (2015). *Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts.* Tech. rep. Stanford University (cit. on p. 67).

John E. Hopcroft and Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company Inc. (cit. on p. 36).

Eric Huang, Richard Socher, Christopher Manning and Andrew Ng (2012). "Improving Word Representations via Global Context and Multiple Word Prototypes". In: *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL).* Ed. by Haizhou Li, Chin-Yew Lin, Miles Osborne, Gary Geunbae Lee and Jong C. Park. Jeju Island, Korea: Association for Computational Linguistics, pp. 873–882 (cit. on p. 99).

Keith A Hutchison (2003). "Is Semantic Priming Due to Association Strength or Feature Overlap? A Microanalytic Review". In: *Psychonomic Bulletin & Review* 10.4, pp. 785–813 (cit. on pp. 6, 46, 82).

Alexander G. Huth, Wendy A. de Heer, Thomas L. Griffiths, Frédéric E. Theunissen and Jack L. Gallant (2016). "Natural Speech Reveals the Semantic Maps that Tile Human Cerebral Cortex". In: *Nature* 532.7600, pp. 453–458 (cit. on pp. 6, 46, 82).

Ozan Irsoy and Claire Cardie (2015). "Modeling Compositionality with Multiplicative Recurrent Neural Networks". In: *3rd International Conference on Learning Representation (ICLR 2015).* Ed. by Yoshua Bengio and Yann LeCun. Sand Diego, CA, USA, pp. 1–10 (cit. on pp. 5, 8, 41, 50, 58–61, 67, 70, 121, 124).

Howard Johnson, Joel Martin, George Foster and Roland Kuhn (2007). "Improving Translation Quality by Discarding Most of the Phrasetable". In: *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Ed. by Jason Eisner. Prague, Czech Republic: Association for Computational Linguistics, pp. 967–975 (cit. on p. 85).

J. A. Johnson (1995). "Semantic Relatedness". In: *Computers and Mathematics with Applications* 29.5, pp. 51–63 (cit. on p. 46).

Daniel Jurafsky and James H. Martin (2014). *Speech and Language Processing (2nd Edition)*. Pearson Education Limited (cit. on pp. 15, 43).

David Jurgens (2013). "Embracing Ambiguity: A Comparison of Annotation Methodologies for Crowdsourcing Word Sense Labels". In: *Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Ed. by Lucy Vanderwende, Hal Daumé III and Katrin Kirchhoff. Atlanta, GA, USA, pp. 556–562 (cit. on p. 100).

David Jurgens, Saif M. Mohammad, Peter Turney and Keith Holyoak (2012). "SemEval-2012 Task 2: Measuring Degrees of Relational Similarity". In: *Proc. of the 6th International Workshop on Semantic Evaluation (SemEval)*. Ed. by Eneko Agirre, Johan Bos and Mona T. Diab. Montréal, Canada: The Association for Computer Linguistics, pp. 356–364 (cit. on p. 100).

Nobuhiro Kaji and Hayato Kobayashi (2017). "Incremental Skip-gram Model with Negative Sampling". In: *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Martha Palmer, Rebecca Hwa and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, pp. 363–371 (cit. on p. 111).

Walter Kintsch (2001). "Predication". In: *Cognitive Science* 25.2, pp. 173–202 (cit. on p. 29).

Svetlana Kiritchenko and Saif M. Mohammad (2016a). "Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best–Worst Scaling". In: *Proc. of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Ed. by Kevin Knight, Ani Nenkova and Owen Rambow. San Diego, California: Association for Computational Linguistics, pp. 811–817 (cit. on pp. 88, 100).

Svetlana Kiritchenko and Saif M. Mohammad (2016b). "Sentiment Composition of Words with Opposing Polarities". In: *Proc. of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Ed. by Bob Carpenter, Amanda Stent and Jason D. Williams. San Diego, California: Association for Computational Linguistics, pp. 1102–1108 (cit. on pp. 59, 63, 64, 67).

Svetlana Kiritchenko and Saif M. Mohammad (2016c). "The Effect of Negators, Modals, and Degree Adverbs on Sentiment Composition". In: *Proc. of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*. Ed. by Alexandra Balahur, Erik van der Goot, Piek Vossen and Andres Montoyo. San Diego, California: Association for Computational Linguistics, pp. 43–52 (cit. on p. 67).

Svetlana Kiritchenko and Saif M. Mohammad (2017). "Best-Worst Scaling More Reliable than Rating Scales: A Case Study on Sentiment Intensity Annotation". In: *Proc. of The Annual Meeting of the Association for Computational Linguistics (ACL)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, pp. 465–470 (cit. on pp. 9, 83, 87, 88, 126).

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba and Sanja Fidler (2015). "Skip-Thought Vectors". In: *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama and Roman Garnett. Curran Associates, Inc., pp. 3294–3302 (cit. on pp. 7, 82, 92).

Kevin Knight and Jonathan May (2009). "Applications of Weighted Automata in Natural Language Processing". In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich and Heiko Vogler. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 571–596 (cit. on p. 37).

Olga Kolesnikova and Alexander Gelbukh (2015). "Measuring Non-compositionality of Verb-Noun Collocations Using Lexical Functions and WordNet Hypernyms". In: *Advances in Artificial Intelligence and Its Applications*. Ed. by Obdulia Pichardo Lagunas, Oscar Herrera Alcántara and Gustavo Arroyo Figueroa. Cham: Springer International Publishing, pp. 3–25 (cit. on p. 45).

Valia Kordoni and Iliana Simova (2014). "Multiword Expressions in Machine Translation". In: *Proc. of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk and Stelios Piperidis. European Languages Resources Association, pp. 1208–1211 (cit. on pp. 42, 68).

Samuel Larkin, Boxing Chen, George Foster, Uli Germann, Eric Joanis, J. Howard Johnson and Roland Kuhn (2010). "Lessons from NRC's Portage System at WMT 2010". In: *Proc. of the 5th Workshop on Statistical Machine Translation (WMT-2010)*. Ed. by Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson and Omar Zaidan. Association for Computational Linguistics, pp. 127–132 (cit. on p. 85).

Quoc Le and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *Proc. of the 31st International Conference on Machine Learning (ICML)*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. JMLR: W&CP, pp. 1188–1196 (cit. on pp. 7, 82, 92).

Omer Levy and Yoav Goldberg (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Proc. of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS)*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence and Kilian Q. Weinberger. Cambridge, MA, USA: MIT Press, pp. 2177–2185 (cit. on pp. 19, 92).

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou and Yoshua Bengio (2017). "A Structured Self-attentive Sentence Embedding". In: *Proc. of the International Conference on Learning Representations (ICLR)*. OpenReview.net (cit. on pp. 7, 82, 92).

Pierre Lison and Andrey Kutuzov (2017). "Redefining Context Windows for Word Embedding Models: An Experimental Study". In: *Proc. of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*. Ed. by Jörg Tiedemann and Nina Tahmasebi. Gothenburg, Sweden: Association for Computational Linguistics, pp. 284–288 (cit. on p. 121).

Jordan J. Louviere (1991). *Best-Worst Scaling: A Model for the Largest Difference Judgments*. Working Paper (cit. on pp. 9, 83, 87, 88, 126).

Jordan J. Louviere, Terry N. Flynn and A. A. J. Marley (2015). *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press (cit. on pp. 9, 83, 87, 126).

Florian Mai, Lukas Galke and Ansgar Scherp (2019). "CBOW Is Not All You Need: Combining CBOW with the Compositional Matrix Space Model". In: *Proc. of Seventh*

International Conference on Learning Representations (ICLR), pp. 1–13 (cit. on pp. 10, 11, 103–106, 113–116, 121, 127).

Jean Maillard and Stephen Clark (2015). "Learning Adjective Meanings with a Tensor-Based Skip-Gram Model". In: *Proc. of the Nineteenth Conference on Computational Natural Language Learning (CoNLL 2015)*. Ed. by Afra Alishahi and Alessandro Moschitti. Association for Computational Linguistics, pp. 327–331 (cit. on p. 60).

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini and Roberto Zamparelli (2014). "SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment". In: *Proc. of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Ed. by Preslav Nakov and Torsten Zesch. Dublin, Ireland: Association for Computational Linguistics, pp. 1–8 (cit. on p. 99).

Diana McCarthy (2002). "Lexical Substitution as a Task for WSD Evaluation". In: *Proc. of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*. Ed. by Phil Edmonds, Rada Mihalcea and Patrick Saint-Dizier. Philadelphia, USA: Association for Computational Linguistics, pp. 109–115 (cit. on p. 98).

Diana McCarthy, Bill Keller and John Carroll (2003). "Detecting a Continuum of Compositionality in Phrasal Verbs". In: *Proc. of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*. Ed. by Francis Bond, Anna Korhonen, Diana McCarthy and Aline Villavicencio. Sapporo, Japan, pp. 73–80 (cit. on pp. 42, 68).

Diana McCarthy and Roberto Navigli (2007). "SemEval-2007 Task 10: English Lexical Substitution Task". In: *Proc. of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Ed. by Eneko Agirre, Lluís Màrquez and Richard Wicentowski. Prague, Czech Republic: Association for Computational Linguistics, pp. 48–53 (cit. on p. 98).

Diana McCarthy and Roberto Navigli (2009). "The English Lexical Substitution Task". In: *Language Resources and Evaluation* 43.2, pp. 139–159 (cit. on p. 99).

Igor Mel'cuk (1996). "Lexical Functions: a Tool for the Description of Lexical Relations in a Lexicon". In: *Lexical Functions in Lexicography and Natural Language Processing*. Ed. by Leo Wanner. Amesterdam: John Benjamins Publishing Company, pp. 37–102 (cit. on p. 45).

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean (2013a). "Efficient Estimation of Word Representations in Vector Space". In: *International Conference on Learning Representations (ICLR 2013)*. Ed. by Yoshua Bengio and Yann LeCun. Arizona, USA (cit. on pp. 10, 19, 23, 27, 64, 69, 73, 78).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado and Jeff Dean (2013b). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proc. of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS)*. Ed. by Chris J.C. Burges, Léon Bottou, Max Welling, Zoubin Ghahramani and Kilian Q. Weinberger. Red Hook, NY, USA: Curran Associates, Inc., pp. 3111–3119 (cit. on pp. 4, 10, 19, 23, 26, 27, 61, 67, 69, 73, 102–106, 108, 109, 114, 120, 127).

George A Miller and Walter G Charles (1991). "Contextual Correlates of Semantic Similarity". In: *Language and Cognitive Processes* 6.1, pp. 1–28 (cit. on pp. 7, 87, 99).

Shachar Mirkin, Ido Dagan and Maayan Geffet (2006). "Integrating Pattern-Based and Distributional Similarity Methods for Lexical Entailment Acquisition". In: *Proc. of the COLING/ACL 2006 Main Conference Poster Sessions*. Sydney, Australia: Association for Computational Linguistics, pp. 579–586 (cit. on pp. 84, 100, 127).

Jeff Mitchell and Mirella Lapata (2008). "Vector-based Models of Semantic Composition". In: *Proc. of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*. Ed. by Johanna D. Moore, Simone Teufel, James Allan and Sadaoki Furui. Association for Computational Linguistics, pp. 236–244 (cit. on p. 72).

Jeff Mitchell and Mirella Lapata (2010). "Composition in Distributional Models of Semantics". In: *Cognitive Science* 34.8, pp. 1388–1429 (cit. on pp. 30, 41, 46, 92, 94, 95, 97, 99, 106).

Thomas M. Mitchell (1997). *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc. (cit. on p. 19).

Saif Mohammad (2008). "Measuring Semantic Distance using Distributional Profiles of Concepts". PhD thesis. Canada: University of Toronto (cit. on pp. 6, 43–45, 82).

Saif M Mohammad (2016). "Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text". In: *Emotion Measurement*. Ed. by Herbert L. Meiselman. Elsevier, pp. 201–237 (cit. on pp. 41, 66).

Saif M. Mohammad (2018a). "Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words". In: *Proc. of The Annual Conference of the Association for Computational Linguistics (ACL)*. Ed. by Hinrich Schuetze, Pascale Fung and Massimo Poesio. Melbourne, Australia: Association for Computational Linguistics, pp. 174–184 (cit. on pp. 88, 100).

Saif M. Mohammad (2018b). "Word Affect Intensities". In: *Proc. of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis and Takenobu Tokunaga. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 174–183 (cit. on p. 100).

Saif Mohammad and Graeme Hirst (2005). "Distributional Measures as Proxies for Semantic Relatedness". In: *arXiv* 1203.1858 (cit. on pp. 6, 45).

Saif Mohammad and Svetlana Kiritchenko (2018). "Understanding Emotions: A Dataset of Tweets to Study Interactions between Affect Categories". In: *Proc. of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis and Takenobu Tokunaga. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 198–209 (cit. on p. 100).

Jane Morris and Graeme Hirst (2004). "Non-Classical Lexical Semantic Relations". In: *Proc. of the Computational Lexical Semantics Workshop (CLS) at HLT-NAACL*. Ed. by Dan Moldovan and Roxana Girju. Boston, Massachusetts: Association for Computational Linguistics, pp. 46–51 (cit. on pp. 6, 43, 44).

F. Mosteller and J. W. Tukey (1968). "Data Analysis, Including Statistics". In: *Revised Handbook of Social Psychology*. Ed. by G. Lindzey and E. Aronson. Vol. 2. Addison-Wesley, pp. 80–203 (cit. on p. 59).

Bryan Orme (2009). *MaxDiff Analysis : Simple Counting , Individual-Level Logit , and HB*. Research Paper Series (cit. on pp. 9, 89).

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch and Chris Biemann (2016). "Human and Machine Judgements for Russian Semantic Relatedness". In: *Proc. of the International Conference on Analysis of Images, Social Networks and Texts (AIST)*. Ed. by Dmitry I. Ignatov, Mikhail Yu. Khachay, Valeri G. Labunets, Natalia Loukachevitch, Sergey I. Nikolenko, Alexander Panchenko, Andrey V. Savchenko and Konstantin Vorontsov. Springer International Publishing, pp. 221–235 (cit. on p. 99).

Patrick Pantel (2005). "Inducing Ontological Co-occurrence Vectors". In: *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ed. by Kevin Knight, Hwee Tou Ng and Kemal Oflazer. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 125–132 (cit. on pp. 4, 16, 45).

Barbara Partee (2004). *Compositionality in Formal Semantics: Selected Papers by Barbara H. Partee*. Oxford, England, Blackwell Publishing (cit. on pp. 3, 28).

Karl Pearson (1894). "Contributions to the Mathematical Theory of Evolution". In: *Philosophical Transactions of the Royal Society of London. A* 185, pp. 71–110 (cit. on p. 41).

Jeffrey Pennington, Richard Socher and Christopher Manning (2014). "Glove: Global Vectors for Word Representation". In: *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang and Walter Daelemans. Association for Computational Linguistics, pp. 1532–1543 (cit. on pp. 23, 78, 92, 94).

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer (2018). "Deep Contextualized Word Representations". In: *Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (NAACL-HLT)*. Ed. by Marilyn Walker, Heng Ji and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237 (cit. on pp. 94, 126, 128).

Kaare B. Petersen and Michael S. Pedersen (2012). *The Matrix Cookbook*. Version 20121115 (cit. on p. 55).

Tony A. Plate (1995). "Holographic Reduced Representations". In: *IEEE Transactions on Neural Networks* 6.3, pp. 623–641 (cit. on p. 35).

Stanley Presser and Howard Schuman (1996). *Questions and Answers in Attitude Surveys: Experiments on Question Form, Wording, and Context*. Quantitative Studies in Social Relation. SAGE Publications (cit. on pp. 7, 87, 126).

Karl H. Pribram, George A. Miller and Eugene Galanter (1960). *Plans and the Structure of Behavior*. New York: Holt, Rinehart and Winston (cit. on p. 32).

Carlos Ramisch, Silvio Cordeiro, Leonardo Zilio, Marco Idiart and Aline Villavicencio (2016). "How Naked is the Naked Truth? A Multilingual Lexicon of Nominal Compound Compositionality". In: *Proceedings of the 54th Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers) (ACL). Ed. by Katrin Erk and Noah A. Smith. Association for Computational Linguistics, pp. 156–161 (cit. on p. 74).

Siva Reddy, Diana McCarthy and Suresh Manandhar (2011). "An Empirical Study on Compositionality in Compound Nouns". In: *Proc. of 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*. Ed. by Haifeng Wang and David Yarowsky. Asian Federation of Natural Language Processing, pp. 210–218 (cit. on pp. 72, 74, 77).

Philip Resnik (1995). "Using Information Content to Evaluate Semantic Similarity in a Taxonomy". In: *Proc. of the 14th International Joint Conference on Artificial Intelligence - Volume 1 (IJCAI)*. Ed. by Chris S. Mellish. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 448–453 (cit. on p. 46).

Nick Riemer (2010). *Introducing Semantics*. Cambridge Introduction to Language and Linguistics (cit. on pp. 2, 43, 44).

Herbert Robbins and Sutton Monro (1951). "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407 (cit. on p. 21).

Frank Rosenblatt (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory (cit. on p. 20).

Herbert Rubenstein and John B. Goodenough (1965). "Contextual Correlates of Synonymy". In: *Communications of the ACM* 8.10, pp. 627–633 (cit. on pp. 7, 46, 87, 99).

Sebastian Rudolph and Eugenie Giesbrecht (2010). "Compositional Matrix-space Models of Language". In: *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Ed. by Jan Hajic, Sandra Carberry and Stephen Clark. Uppsala, Sweden: Association for Computational Linguistics, pp. 907–916 (cit. on pp. 4, 5, 8, 28, 29, 31, 32, 34, 36, 37, 39, 51, 60, 79, 104, 128).

David E Rumelhart, Geoffrey E Hinton and Ronald J Williams (1986). "Learning Representations by Back-propagating Errors". In: *nature* 323.6088, pp. 533–536 (cit. on pp. 20, 22).

Virginia R de Sa (1994). "Learning Classification with Unlabeled Data". In: *Advances in Neural Information Processing Systems 7(NIPS)*. Ed. by Gerald Tesauro, David S. Touretzky and Todd K. Leen. MIT Press, pp. 112–119 (cit. on p. 23).

John I. Saeed (1997). *Semantics*. Oxford, Blackwell (cit. on p. 2).

Bahar Salehi, Paul Cook and Timothy Baldwin (2015). "A Word Embedding Approach to Predicting the Compositionality of Multiword Expressions". In: *Proc. of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Ed. by Rada Mihalcea, Joyce Chai and Anoop Sarkar. Association for Computational Linguistics, pp. 977–983 (cit. on p. 78).

Gerard Salton and Michael J. McGill (1986). *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc. (cit. on pp. 29, 95).

Gerard Salton, Anita Wong and Chung-Shu Yang (1975). "A Vector Space Model for Automatic Indexing". In: *Communications of the ACM* 18.11, pp. 613–620 (cit. on pp. 17, 30).

Enrico Santus, Frances Yung, Alessandro Lenci and Chu-Ren Huang (2015). "EVALution 1.0: an Evolving Semantic Dataset for Training and Evaluation of Distributional

Semantic Models". In: *Proc. of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*. Ed. by Christian Chiarcos, John Philip McCrae, Petya Osenova, Philipp Cimiano and Nancy Ide. Beijing, China: Association for Computational Linguistics, pp. 64–69 (cit. on pp. 82, 99).

M.P. Schützenberger (1961). "On the Definition of a Family of Automata". In: *Information and control* 4.2, pp. 245–270 (cit. on p. 37).

Richard Socher (2014). "Recursive Deep Learning for Natural Language Processing and Computer Vision". PhD thesis. Stanford University (cit. on p. 2).

Richard Socher, Brody Huval, Christopher D Manning and Andrew Y Ng (2012). "Semantic Compositionality through Recursive Matrix-Vector Spaces". In: *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Ed. by Junichi Tsujii, James Henderson and Marius Pasca. Association for Computational Linguistics, pp. 1201–1211 (cit. on pp. 60, 66, 92, 104).

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng and Christopher Potts (2013). "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proc. of the conference on empirical methods in natural language processing (EMNLP)*. Ed. by David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu and Steven Bethard. Association for Computational Linguistics, pp. 1631–1642 (cit. on pp. 66, 67, 104, 112).

Andrew Spencer (1991). *Morphological Theory: An Introduction to Word Structure in Generative Grammar*. Blackwell Publishers (cit. on pp. 3, 4).

James H. Steiger (1980). "Tests for Comparing Elements of a Correlation Matrix". In: *Psychological Bulletin* 87.2, pp. 245–251 (cit. on p. 96).

Luke Strongman (2017). "Language Evolution, Acquisition, Adaptation and Change". In: *Sociolinguistics: Interdisciplinary Perspectives*. Ed. by Xiaoming Jiang. InTechOpen, pp. 17–32 (cit. on p. 2).

Kai Sheng Tai, Richard Socher and Christopher D. Manning (2015). "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks". In: *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (ACL-IJCNLP)*. Beijing, China: Association for Computational Linguistics, pp. 1556–1566 (cit. on pp. 92, 104).

Peter D Turney (2012). "Domain and Function: A Dual-Space Model of Semantic Relations and Compositions". In: *Journal of Artificial Intelligence Research* 44.1, pp. 533–585 (cit. on pp. 31, 46, 97, 99, 121).

Peter D Turney (2013). "Distributional Semantics Beyond Words: Supervised Learning of Analogy and Paraphrase". In: *Transactions of the Association of Computational Linguistics* 1, pp. 353–366 (cit. on p. 17).

Peter D Turney, Yair Neuman, Dan Assaf and Yohai Cohen (2011). "Literal and Metaphorical Sense Identification through Concrete and Abstract Context". In: *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Regina Barzilay and Mark Johnson. Association for Computational Linguistics, pp. 680–690 (cit. on pp. 94, 95).

Xingyou Wang, Weijie Jiang and Zhiyong Luo (2016). "Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts". In: *Proc. of*

the 26th International Conference on Computational Linguistics (COLING). Ed. by Yuji Matsumoto and Rashmi Prasad. The COLING 2016 Organizing Committee, pp. 2428–2437 (cit. on p. 67).

Marion Weller, Fabienne Cap, Stefan Müller, Sabine Schulte im Walde and Alexander Fraser (2014). "Distinguishing Degrees of Compositionality in Compound Splitting for Statistical Machine Translation". In: *Proc. of the First Workshop on Computational Approaches to Compound Analysis (ComAComA 2014)*. Ed. by Ben Verhoeven, Walter Daelemans, Menno van Zaanen and Gerhard van Huyssteen. Dublin, Ireland: Association for Computational Linguistics and Dublin City University, pp. 81–90 (cit. on pp. 28, 42, 68).

Dominic Widdows (2008). "Semantic Vector Products: Some Initial Investigations". In: *Proc. of the Second AAAI Symposium on Quantum Interaction (QI-2008)*. Ed. by Peter D. Bruza, William Lawless, Keith Van Rijsbergen, Donald A. Sofge and Bob Coecke. College Publications (cit. on pp. 30, 94, 95).

Dong Wu and Mingmin Chi (2017). "Long Short-Term Memory With Quadratic Connections in Recursive Neural Networks for Representing Compositional Semantics". In: *IEEE Access* 5, pp. 16077–16083 (cit. on p. 78).

Majid Yazdani, Meghdad Farahmand and James Henderson (2015). "Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions". In: *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Lluís Màrquez, Chris Callison-Burch and Jian Su. Association for Computational Linguistics, pp. 1733–1742 (cit. on pp. 72, 75, 76, 78).

Ainur Yessenalina and Claire Cardie (2011). "Compositional Matrix-space Models for Sentiment Analysis". In: *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Regina Barzilay and Mark Johnson. Edinburgh, United Kingdom: Association for Computational Linguistics, pp. 172–182 (cit. on pp. 5, 8, 28, 41, 50, 58–62, 66, 67, 106, 124).

Torsten Zesch and Iryna Gurevych (2010). "Wisdom of Crowds Versus Wisdom of Linguists–Measuring the Semantic Relatedness of Words". In: *Natural Language Engineering* 16.1, pp. 25–59 (cit. on pp. 44, 46, 47, 82).

Xiaodan Zhu, Parinaz Sobhani and Hongyu Guo (2016). "DAG-Structured Long Short-Term Memory for Semantic Compositionality". In: *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACLL-HLT)*. Ed. by Kevin Knight, Ani Nenkova and Owen Rambow. San Diego, California: Association for Computational Linguistics, pp. 917–926 (cit. on pp. 94, 100, 127).

BiRD Questionnaire

Instructions on "Judgment of Closeness In Meaning (or Relatedness) of Pairs of Common English Words and Phrases"

A.1 General Instructions

- Attempt these questions only if you are fluent in English.
- Your responses are confidential. Any publications based on these responses will not include your specific responses, but rather aggregate information from many individuals. We will not ask any information that can be used to identify who you are.

A.2 The Task

Your task is to judge the degree of closeness in meaning (or amount of relatedness) of pairs of English terms (words or phrases). Since it is difficult to give numerical scores of closeness, we ask you to judge which pair of terms is considered closer in meaning by most people. For example, most people will agree that:

- doctor and physician are closer in meaning (or more related to each other) than fish and table
- student and tutor are closer in meaning than student and mother
- piano player and pianist are closer in meaning than piano player and piano
- piano player and pianist are closer in meaning than air show and system

In each instance, you will be given four pairs at a time. Think about the meanings of each of the pairs and answer these questions:

1. Which pair is most close in meaning (or most related)?
2. Which pair is least close in meaning (or least related)?

Notes:

- If in the given set of four pairs, two term pairs are equally close to each other and they are also the most close pairs, then select either one of them as the most related. Similarly, if two equally related pairs are also the least related pairs, then select either one of them as the least related.
- Synonyms (words with identical meanings) are maximally close to each other (or most related).
- Terms that are not synonymous may be close in meaning to different degrees. For example, bank and money are highly related to each other, whereas bank and peace are not very related. Non-synonymous terms that are close in meaning tend to be used together in sentences more often than random chance.
- If a term has more than one meaning, consider that meaning which is closest to the meaning of the other term in the pair. If both terms have multiple meanings, then consider those meanings that are closest to each other.
- The word order in phrases matters. Two phrases with the same words but in different order can be very close in meaning, somewhat related, or unrelated. Thus, judge the relatedness for such phrases after careful consideration of the meanings.
- If you do not know the meaning of a term, check the
  - English Wikipedia (https://en.wikipedia.org/wiki/MainPage),
  - Merriam-Webster Dictionary (https://www.merriam-webster.com/),
  - or Cambridge dictionary (https://dictionary.cambridge.org/).

A.3 Examples

Example 1:

Q1: Which pair is most close in meaning (or most related)?

- (building block, unit)
- (traffic light, intersection)
- (water quality, health)
- (fantasy world, system)

Answer: (building block, unit)

Q2: Which pair is least close in meaning (or least related)?

- (building block, unit)
- (traffic light, intersection)
- (water quality, health)

- (fantasy world, system)

Answer: (fantasy world, system)

Example 2:

Q1: Which pair is most close in meaning (or most related)?

- (school council, student)
- (west coast, mountain)
- (relevant information, historical data)
- (blue light, light blue)

Answer: (school council, student)

Q2: Which pair is least close in meaning (or least related)?

- (school council, student)
- (west coast, mountain)
- (relevant information, historical data)
- (blue light, light blue)

Answer: (blue light, light blue)

Example 3:

Q1: Which pair is most close in meaning (or most related)?

- (house dog, animal)
- (data processing, execution)
- (jazz band, dance)
- (recording studio, art school)

Answer: (house dog, animal)

Q2: Which pair is least close in meaning (or least related)?

- (house dog, animal)
- (data processing, execution)
- (jazz band, dance)
- (recording studio, art school)

Answer: (recording studio, art school)

Purpose

Your responses will be used to study how people perceive closeness in meaning (or relatedness). Our eventual goal is to build computer systems that automatically predict the relatedness of words and phrases. The data is collected only for research purposes.