Sentential Representations in Distributional Semantics

by

The Nghia Pham

Submitted to the Doctoral School in Cognitive and Brain Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

UNIVERSITY OF TRENTO

November 2016

Author . . . . . .
Doctoral School in Cognitive and Brain Sciences
November 2, 2016

Certified by . . . . . .
Marco Baroni
Associate Professor
Thesis Supervisor

Sentential Representations in Distributional Semantics

by

The Nghia Pham

Submitted to the Doctoral School in Cognitive and Brain Sciences
on November 2, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis is about the problem of representing sentential meaning in distributional semantics. Distributional semantics obtains the meanings of words through their usage, based on the hypothesis that words occurring in similar contexts will have similar meanings. In this framework, words are modeled as distributions over contexts and are represented as vectors in high dimensional space. Compositional distributional semantics attempts to extend this approach to higher linguistics structures. Some basic composition models proposed in literature to obtain the meaning of phrases or possibly sentences show promising results in modeling simple phrases. The goal of the thesis is to further extend these composition models to obtain sentence meaning representations.

The thesis puts more focus on unsupervised methods which make use of the context of phrases and sentences to optimize the parameters of a model. Three different methods are presented. The first model is the PLF model, a practical composition and linguistically motivated model which is based on the lexical function model introduced by Baroni and Zamparelli (2010) and Coecke et al. (2010). The second model is the Chunk-based Smoothed Tree Kernels (CSTKs) model, extending Smoothed Tree Kernels (Mehdad et al., 2010) by utilizing vector representations of chunks. The final model is the C-PHRASE model, a neural network-based approach, which jointly optimizes the vector representations of words and phrases using a context predicting objective.

The thesis makes three principal contributions to the field of compositional distributional semantics. The first is to propose a general framework to estimate the parameters and evaluate the basic composition models. This provides a fair way to comparing the models using a set of phrasal datasets. The second is to extend these basic models to the sentence level, using syntactic information to build up the sentence vectors. The third contribution is to evaluate all the proposed models, showing that they perform on par with or outperform competing models presented in the literature.

Thesis Supervisor: Marco Baroni
Title: Associate Professor

Acknowledgments

I would like to thank my advisor, Marco Baroni, for his continuous support during the course of my PhD. His immense knowledge, creative insight, amazing sense of humor and endless dedication have helped me in all the time of research and writing of this thesis. I cannot express enough how much I appreciate his guidance and encouragement throughout my PhD career.

I would also like to thank my Oversight Committee, Raffaella Bernardi and Roberto Zamparelli, for all the discussion and invaluable suggestions.

I am indebted to Georgiana Dinu, Angeliki Lazaridou, German Kruszewski, Denis Paperno, Fabio Massimo Zanzotto and Lorenzo Ferrone for their contribution to my work. A very special thank to them and all the exceptional researchers with whom I had the chance to collaborate.

Thank you to every member of COMPOSES and the CLICers for creating the most enjoyable working environment and for the precious feedback. I would also like to express my gratitude to the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES) for the financial support.

I am grateful to the Doctoral School in Cognitive and Brain Sciences, especially the head of the school, Francesco Pavani, and the administrator, Leah Mercanti, for their dedication to the program and the students.

Finally, I thank my family for their unconditional love and tremendous patience; all my friends, especially Jan Bim, Aidas Aglinskas, Adam Liska, Gergely David, Marco Marelli and again Georgiana Dinu, Angeliki Lazaridou and German Kruszewski for the wonderful time full of fun and adventures in Rovereto.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decades, computers have become important assistants for human and kept on making our jobs faster and easier. They can help us in many tasks ranging from simple ones like designing circuits to complicated ones like predicting weather or flying airplanes. Despite being so useful and increasingly powerful, computers are still not able to understand human language and therefore cannot use language as a mean of communication. Until recently, interaction between human and computers mostly took place through constrained interfaces such as keyboards. It has been a great challenge to enable computers to understand natural languages.

A fundamental aspect of this challenge is how to represent meaning and how to automatically transform natural language into this type of representation. Traditionally, formal semantics has attempted to represent language utterances such as sentences as logic formulas (Montague, 1970). The meaning of a sentence such as *"every student left"* can be represented as the following first order logic formula

$\forall x(student(x) \rightarrow left(x))$

This type of representation is quite attractive since it allows automatic reasoning. For example: given that *every student left*, $\forall x(student(x) \rightarrow left(x))$, and *John is a student*, $student(John)$, using logic tools, a computer can deduce that *John left*, $left(John)$. Formal semantics also introduces methods, such as lambda calculus, for building the meaning of sentences compositionally based on their components.

Despite its attractiveness, formal semantics cannot automatically learn the meaning of

the words themselves and it has no way to obtain the relationship between words from the data. For instance, formal semantics cannot learn that *murder* $\rightarrow$ *kills* or *dislike* is similar to *hate*. Although this type of information can be manually encoded, this limitation highly restricts the use of formal semantics in practical applications.

Distributional semantics, on the contrary, can learn the relationship between words from increasingly available text corpora (Turney and Pantel, 2010; Erk, 2012; Clark, 2015). It is based on the distributional hypothesis *"Words that appear in similar contexts tend to have similar meanings"* (Miller and Charles, 1991). Given a text corpus, distributional semantics can build the meaning of words as co-occurrence vectors and, based on this type of representation, it can tell how similar words are to each other. This is very useful because assessing similarity in meaning is central to many language technology applications such as question answering or information retrieval.

However, historically, there has been no obvious way to extract the meaning of linguistic utterance beyond words. Unlike formal semantics, distributional semantics lacks a framework to construct the meaning of phrases and sentences based on their components. The question of assessing meaning similarity above the word level within the distributional paradigm has received a lot of attention in recent years (Mitchell and Lapata, 2008, 2010; Zanzotto et al., 2010; Guevara, 2010; Baroni and Zamparelli, 2010; Coecke et al., 2010). A number of compositional frameworks have been proposed in the literature, each of these defining operations to combine word vectors into representations for more complex structures such as phrases. However, these works pay more attention to simple phrases, e.g. adjective noun construction, with limited experiments at the sentence level. This thesis focus on how to obtain vectors for sentences using distributional methods, as sentences are the basic language structure that human uses to interact with each other. Moreover, It looks at how information such as syntactic structure can be incorporated into the process of building sentence representations.

In chapter 2, I begin with an overview of distributional semantics. I discuss the two general ways of obtaining word vectors from the literature: using co-occurrence count and predicting the context. Also, I will briefly go over the composition models in the literature, which can combine single word vectors into vectors for phrases. These methods

range from simple parameter-free models like addition to complex models involving tensor multiplication and estimation.

Chapter 3 takes a more thorough look at these composition models. First, I introduce a general and effective method for estimating the parameters of every composition model. Then, based on this estimation method, I describe a truly fair way to compare all the models I consider using a variety of phrase datasets. This chapter is of great importance since, based on the results presented in it, we can see which models are more effective in building phrasal vectors, that thus can further be adapted to derive sentential vectors.

The rest of this thesis will focus on methods to achieve the goal of obtain meaning representation of sentences. These methods are very different from each other but are built using the same building blocks: simple composition models.

Chapter 4 describes the practical lexical function (PLF), a purely compositional method, which is based on the lexical function model introduced by Baroni and Zamparelli (2010) and Coecke et al. (2010). This lexical function model is inspired by the function application view of composition that is common in formal semantics. In this model, nouns are represented as vectors, adjectives are matrices that modify the meaning of a noun by transforming its vector to another vector. Other content words such as transitive verbs or adverb are represented as high-order tensors. In PLF, high order tensors are avoided and replaced with multiple matrices, therefore problems inherent to the lexical function model such as data sparseness and highly complex computation are avoided while other attractive features like syntax-sensitive are maintained.

Another method, which is introduced in chapter 5, is a combination of tree kernels and basic composition. Syntactic tree kernels are powerful tools, which use the set of subtrees of the syntactic tree of a sentence as its vector representation. Using this type of representation, task-specific similarity between sentences are learned by assigning weights to the syntactic subtrees. An disadvantage of this method is the lack of a way to merge similar subtrees using the similarity betweens words and simple phrases, which is nicely supplied by the basic composition models.

Finally, in chapter 6, I present C-PHRASE, another alternative to obtain sentence vector using neural networks. C-PHRASE is built based on Skipgram and CBOW by Mikolov

et al. (2013c), two very efficient methods to obtain good word vector representation. By incorporating syntactic information into the objective function, the C-PHRASE model is able to learn word and phrase vectors jointly. It does not only improve the quality of the word vectors but also obtain good sentence vectors which are then tested on several sentential task such as textual similarity, entailment and sentiment analysis.

Additionally, in the appendix A, I describes the practical tools I have built during the course of this thesis, which are put together to form the DISSECT toolkit. The toolkit is written in Python, a high-level, general purpose programming language, which is popular in the NLP community. The toolkit consists of a set of ways to build word vectors from co-occurrence counts, methods to combine them into phrase or sentence vectors and to estimate the parameters for those models. It is highly extensible, which hopefully will be useful for researchers who are interested in distributional semantics in general or more specifically to those who are interested in compositional distributional semantics.

Chapter 2

Background

2.1 Distributional Semantics

The need to assess similarity in meaning is central to many language technology applications, and distributional methods are the most robust approach to the task. These methods measure word similarity based on patterns of occurrence in large corpora, following the intuition that similar words occur in similar contexts. More precisely, vector space models, the most widely used distributional models, represent words as high-dimensional vectors, where the dimensions represent (functions of) context features, such as co-occurring context words. The relatedness of two words is assessed by comparing their vector representations.

There are two prominent approaches in distributional semantics to obtain the word vectors. The traditional approach, which is called *count model* (Baroni et al., 2014), is a direct realization of the distributional hypothesis where the context features are the words that co-occur with the target words. The more recent approach, which is referred as *predict* model (Baroni et al., 2014), can be seen as a more indirect realization of the distributional hypothesis. In this approach the word vectors are typically learned by predicting the context, e.g. the words that occur after them.

	house	sleep	eat	milk	walk	chase
cat	10	22	35	16	5	12
dog	14	23	50	13	35	11
couch	13	19	15	0	0	0

Table 2.1: Example vectors in distributional space

2.1.1 Count model

In count models, to obtain the word vectors, the first step is to select a fixed set of contexts, which are typically the most common content words (e.g. nouns, verbs, adjectives, adverbs). These words are used as the dimension of the vectors in the high-dimensional space. Then, (a function of) the the number of times the target word occurs with a given context word is computed and used as the value of the corresponding dimension. Table 2.1 illustrates some of the vectors in such a semantic space.

Normally, vectors with raw frequency (without applying a frequency weighting function) do not perform well in practical applications since the dimensions corresponding to frequent context words, which contribute little to the meaning of other words, tend to have the highest values dominating other dimensions. Weighting schemes are therefore used to lessen the effect of highly frequent context words. Some popular weighting schemes include TF-IDF (Salton, 1979), pointwise mutual information (PMI) (Church and Hanks, 1990).

$$pmi(x, y) = log \frac{p(x, y)}{p(x)p(y)} \tag{2.1}$$

Other weighting schemes inspired by PMI are positive pointwise mutual information PPMI or local mutual information (LMI) (Bullinaria and Levy, 2007; Evert, 2005).

To reduce the number of dimension in the vector space, thus reducing the amount of computation, methods such as truncated SVD (Golub and Van Loan, 1996) or NMF (Lee and Seung, 1999) are used. SVD, singular value decomposition, factorizes an matrix $\mathbf{M}$ into 3 components $\mathbf{M} = \mathbf{U} * \mathbf{S} * \mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{S}$ is a non-negative diagonal matrix. Typically, in SVD, $\mathbf{U} * \mathbf{S}$ are used as a dimension-reduced version of the original matrix $\mathbf{M}$ and $\mathbf{V}$ is discarded. With NMF, the original matrix $\mathbf{M}$ is

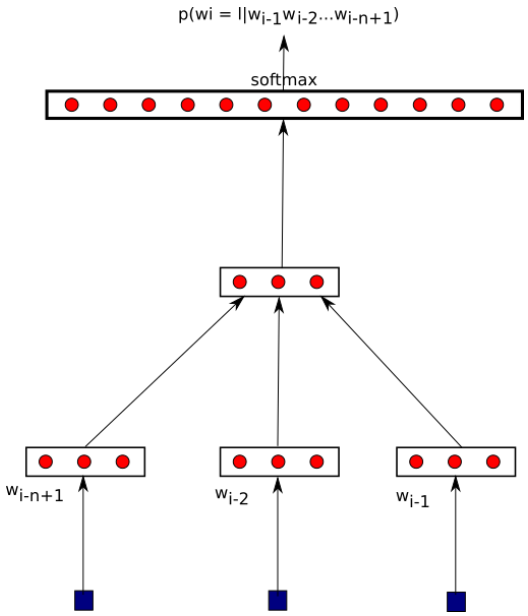


Figure 2-1: Neural probablistic language model

factorized into two non-negative matrices $\mathbf{M} = \mathbf{W} * \mathbf{H}$, and $\mathbf{W}$ is used in place of $\mathbf{M}$. These dimensionality reduction methods can also have a side effect of reducing noises from the corpus thus resulting in semantic space with higher quality.

2.1.2 Predict model

As the name suggests, predict models learn word vectors by using them to predict the context. In these models, the dimensions in a word vector are trained to maximize the probability of the contexts where the word is observed in the corpus. Because similar words occur in similar contexts, similar vectors are naturally assigned to similar words.

One of the first predict models was the neural language model. The main goal of language modeling is to predict the current word given the history (all the words before the current word). An early neural network for language modeling by Bengio et al. (2003) is presented in figure 2-1. Here words are presented as low-dimension vectors and a function based on the presentation of the last n words is used to compute the distribution of the words that occur after them. The word vector representations obtained from these methods, despite originally being a byproduct, are a good proxy of word meaning.

Since Bengio et al. (2003), other frameworks to learn word representations have emerged,

some notable models being those presented by Collobert and Weston (2008); Collobert et al. (2011); Huang et al. (2012); Turian et al. (2010).

One of the most influential works in this direction is Mikolov et al. (2013c), where the authors introduce two very simple and efficient method to obtain good word meaning representation: the **Skip-gram** and **C-BOW** models. The Skip-gram model derives the vector of a target word by setting its weights to predict the words surrounding it in the corpus. More specifically, the objective function is:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.2}$$

where the word sequence $w_1, w_2, ..., w_T$ is the training corpus and c is the size of the window around the target word w_t, consisting of the context words w_{t+j} that must be predicted by the induced vector representation for the target.

While Skip-gram learns the word vector by using the target word to predict the surrounding words, C-BOW, on the other hand, uses the combination of the surround words to predict the word in the middle. The objective function is:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}..w_{t-1}, w_{t+1}..w_{t+c}) \tag{2.3}$$

Despiting being very simple, Skip-gram and C-BOW models obtain extremely high quality vectors. They achieved the state of the art results in several lexical tasks, including the word analogy task (Mikolov et al., 2013c) where given a pair exemplifying a relation (*boy*/*son*) and a test word (*girl*); they can find the word (*daughter*) that instantiates the same relation with the test word as that of the example pair.

2.2 Compositional models

The vector representations provided by distributional semantic models are very useful in applications that require a representation of word meaning in general, or in particular require an measure of similarity between word meanings. Examples of such applications include question answering, machine translation and automated thesaurus construction (Du-

mais, 2003; Turney and Pantel, 2010). However, these models cannot be used directly to obtain the meaning representations for complex phrases or sentences since there are unlimited number of possible sentences and few of them occur frequently enough in text corpora in order for distributional methods to obtain a reliable vector representation. In recent year, the question of obtaining semantic representation *above the word level* within the distributional paradigm has received a lot of attention (Mitchell and Lapata, 2008, 2010; Grefenstette and Sadrzadeh, 2011a; Guevara, 2010; Zanzotto et al., 2010). Most work dealing with this question relies on the compositionality of languages, in which the meaning of an utterance can be represented as a combination of the meaning of its parts. A number of compositional frameworks have been proposed in the literature, each of these defining operations to combine word vectors into representations for phrases or even sentences.

Being probably the most notable work on the topic, Mitchell and Lapata (2010) have proposed several classes of composition models. The first method, also the most intuitive way to compose the phrase vector from its component words, is by summing its word vectors. This model is called additive model (*Add*)

$$\overrightarrow{p} = \overrightarrow{u} + \overrightarrow{v} \tag{2.4}$$

where $\overrightarrow{p}$ is the vector of the phrase and $\overrightarrow{u}$ and $\overrightarrow{v}$ are the vectors of its component words.

An extension of this model is the weighted additive model (*WAdd*), where the contribution of the components are determined by the two pre-defined weights

$$\overrightarrow{p} = w_1\overrightarrow{u} + w_2\overrightarrow{v} \tag{2.5}$$

Note that if $w_1 = w_2 = 1$, WAdd becomes the simple Add model.

Another simple alternative to additive is the multiplicative model *Mult*, which utilizes the pointwise multiplication operation

$$\overrightarrow{p} = \overrightarrow{u} \odot \overrightarrow{v} \tag{2.6}$$

Both addition and multiplication are commutative, thus making simple additive and multiplicative model symmetric, meaning they do not take word order into account. There-

fore, *pandas eat bamboo* and *bamboo eats pandas* will have the same meaning representation for these models.

Another model proposed by Mitchell and Lapata (2010), the dilation model, is asymmetric. First the vector $\vec{v}$ is decomposed into 2 components, one is parallel to $\vec{u}$ and the other is orthogonal to $\vec{u}$. The component parallel to u is then stretched with a weight λ and then recombined with the other component so that the final vector is more similar to $\vec{u}$.

$$\vec{p} = ||\vec{u}||_2^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u} \tag{2.7}$$

Guevara (2010) and Zanzotto et al. (2010) introduce the full additive (*FullAdd*) model, a similar model to WAdd but replacing the scalar weights with matrices.

$$\vec{p} = \mathbf{W}_1\vec{u} + \mathbf{W}_2\vec{v} \tag{2.8}$$

All the models above were inspired by mathematical operations on vectors with little insight from linguistics. Baroni and Zamparelli (2010) and Coecke et al. (2010) introduce the lexical function model (*LF*), which follows the function application intuition of formal semantics, where a noun is a set of entities and an adjective is a function that maps a set of entities to another set. Here, a noun is represented as a vector and an adjective is a matrix that maps a vector to another.

$$\vec{p} = \mathbf{A}_u\vec{v} \tag{2.9}$$

This model can also be generalized to other parts of speech, e.g. a transitive verb can be represented as a 3-order tensor that maps a pair of vectors to a vector.

The last model, introduced by Socher et al. (2012) is a neural network-based generalization of LF. In this model, every word is represented as a matrix and a vector ($[\mathbf{A}_w, \vec{w}]$), and the composition function is:

$$\vec{p} = tanh\left(\mathbf{M}\begin{bmatrix}\mathbf{A}_u\vec{v} \\ \mathbf{A}_v\vec{u}\end{bmatrix}\right) \tag{2.10}$$

where the same matrix $\mathbf{M}$ is used for every phrase.

The composition models discussed in this chapter lay a good foundation for obtaining semantic representation of sentences. However, in order to achieve this goal, we must perform a systematic comparison between the model in order to evaluate their strengths and weaknesses, thus enabling us to select the most promising model to further develop it into a good model to obtain sentential meaning. In chapter 3 and 4, I will present my attempt to perform this selection and development process.

Chapter 3

General Parameter Estimation and Evaluation of Basic Composition Models

3.1 Introduction

The question of assessing meaning similarity *above the word level* within the distributional paradigm has received a lot of attention in recent years. A number of compositional frameworks have been proposed in the literature, each of these defining operations to combine word vectors into representations for phrases. These models are briefly discussed in chapter 2 and range from simple but robust methods such as vector addition to more advanced methods, such as learning function words as tensors and composing constituents through inner product operations.

Empirical evaluations in which alternative methods are tested in comparable settings are thus called for. This is complicated by the fact that the proposed compositional frameworks package together a number of choices that are conceptually distinct, but difficult to disentangle. Broadly, these concern:

(i) the input representations fed to composition; (ii) the composition operation proper; (iii) the method to estimate the parameters of the composition operation.

For example, Mitchell and Lapata in their classic 2010 study propose a set of composition operations (multiplicative, additive, etc.), but they also experiment with two different kinds of input representations (vectors recording co-occurrence with words vs. distribu-

tions over latent topics) and use supervised training via a grid search over parameter settings to estimate their models. Guevara (2010), to give just one further example, is not only proposing a different composition method with respect to Mitchell and Lapata, but he is also adopting different input vectors (word co-occurrences compressed via SVD) and an unsupervised estimation method based on minimizing the distance of composed vectors to their equivalents directly extracted from the source corpus.

Blacoe and Lapata (2012) have recently highlighted the importance of teasing apart the different aspects of a composition framework, presenting an evaluation in which different input vector representations are crossed with different composition methods. However, two out of three composition methods they evaluate are parameter-free, so that they can side-step the issue of fixing the parameter estimation method.

In this chapter, I evaluate *all* composition methods I know of, excluding a few that lag behind the state of the art or are special cases of those I consider, *while keeping the estimation method constant*. This evaluation is made possible by our extension to all target composition models of the corpus-extracted phrase approximation method originally proposed in *ad-hoc* settings by Baroni and Zamparelli (2010) and Guevara (2010).

For the models for which it is feasible, I compare the phrase approximation approach to supervised estimation with crossvalidation, and show that phrase approximation is competitive, thus confirming that I am not comparing models under poor training conditions. Our tests are conducted over three tasks that involve different syntactic constructions and evaluation setups. Finally, I consider a range of parameter settings for the input vector representations, to insure that our results are not too brittle or parameter-dependent.

3.2 Least-squares model estimation using corpus-extracted phrase vectors

Notation Given two matrices $X, Y \in \mathbf{R}^{m \times n}$ I denote their inner product by $\langle X, Y \rangle$, ($\langle X, Y \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij}$). Similarly I denote by $\langle u, v \rangle$ the dot product of two vectors $u, v \in \mathbf{R}^{m \times 1}$ and by $||u||$ the Euclidean norm of a vector: $||u|| = \langle u, u \rangle^{1/2}$. I use

the following Frobenius norm notation: $||X||_F = \langle X, X \rangle^{1/2}$. Vectors are assumed to be column vectors and I use x_i to stand for the i-th $(m \times 1)$-dimensional column of matrix X. I use $[X, Y] \in \mathbf{R}^{m \times 2n}$ to denote the horizontal concatenation of two matrices while $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] \in \mathbf{R}^{2m \times n}$ is their vertical concatenation.

General problem statement I assume vocabularies of constituents $\mathcal{U}$, $\mathcal{V}$ and that of resulting phrases $\mathcal{P}$. The training data consist of a set of tuples (u, v, p) where p stands for the phrase associated to the constituents u and v:

$$T = \{(u_i, v_i, p_i) | (u_i, v_i, p_i) \in \mathcal{U} \times \mathcal{V} \times \mathcal{P}, 1 \leq i \leq k\}$$

I build the matrices $U, V, P \in \mathbf{R}^{m \times k}$ by concatenating the vectors associated to the training data elements as columns.[1]

Given the training data matrices, the general problem can be stated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||P - fcomp_{\boldsymbol{\theta}}(U, V)||_F$$

where $fcomp_{\boldsymbol{\theta}}$ is a composition function and $\boldsymbol{\theta}$ stands for a list of parameters that this composition function is associated to. The composition functions are defined: $fcomp_{\boldsymbol{\theta}} : \mathbf{R}^{m \times 1} \times \mathbf{R}^{m \times 1} \rightarrow \mathbf{R}^{m \times 1}$ and $fcomp_{\boldsymbol{\theta}}(U, V)$ stands for their natural extension when applied on the individual columns of the U and V matrices.

WAdd The weighted additive model returns the sum of the composing vectors which have been re-weighted by some scalars w_1 and w_2: $\vec{p} = w_1 \vec{u} + w_2 \vec{v}$. The problem becomes:

$$w_1^*, w_2^* = \arg\min_{w_1, w_2 \in \mathbf{R}} ||P - w_1 U - w_2 V||_F$$

The optimal w_1 and w_2 are given by:

$$w_1^* = \frac{||V||_F^2 \langle U, P \rangle - \langle U, V \rangle \langle V, P \rangle}{||U||_F^2 ||V||_F^2 - \langle U, V \rangle^2} \tag{3.1}$$

[1] In reality, not all composition models require u, v and p to have the same dimensionality.

$$w_2^* = \frac{||U||_F^2 \langle V, P\rangle - \langle U, V\rangle \langle U, P\rangle}{||U||_F^2 ||V||_F^2 - \langle U, V\rangle^2} \tag{3.2}$$

Dil Given two vectors $\vec{u}$ and $\vec{v}$, the dilation model computes the phrase vector $\vec{p} = ||\vec{u}||^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$ where the parameter λ is a scalar. The problem becomes:

$$\lambda^* = \arg\min_{\lambda \in \mathbf{R}} ||P - VD_{||u_i||^2} - UD_{(\lambda-1)\langle u_i, v_i\rangle}||_F$$

where by $D_{||u_i||^2}$ and $D_{(\lambda-1)\langle u_i, v_i\rangle}$ I denote diagonal matrices with diagonal elements (i, i) given by $||u_i||^2$ and $(\lambda - 1)\langle u_i, v_i\rangle$ respectively. The solution is:

$$\lambda^* = 1 - \frac{\sum_{i=1}^k \langle u_i, (||u_i||^2 v_i - p_i)\rangle \langle u_i, v_i\rangle}{\sum_{i=1}^k \langle u_i, v_i\rangle^2 ||u_i||^2}$$

Mult Given two vectors $\vec{u}$ and $\vec{v}$, the weighted multiplicative model computes the phrase vector $\vec{p} = \vec{u}^{w_1} \odot \vec{v}^{w_2}$ where $\odot$ stands for component-wise multiplication. I assume for this model that $U, V, P \in \mathbf{R}_{++}^{m\times n}$, i.e. that the entries are strictly larger than 0: in practice I add a small smoothing constant to all elements to achieve this (Mult performs badly on negative entries, such as those produced by SVD). I use the w_1 and w_2 weights obtained when solving the much simpler related problem:[2]

$$w_1^*, w_2^* = \arg\min_{w_1, w_2 \in \mathbf{R}} ||log(P) - log(U.^\wedge w_1 \odot V.^\wedge w_2)||_F$$

where $.^\wedge$ stands for the component-wise power operation. The solution is the same as that for WAdd, given in equations (1) and (2), with $U \rightarrow log(U)$, $V \rightarrow log(V)$ and $P \rightarrow log(P)$.

FullAdd The full additive model assumes the composition of two vectors to be $\vec{p} = W_1\vec{u} + W_2\vec{v}$ where $W_1, W_2 \in \mathbf{R}^{m\times m}$. The problem is:

$$[W_1, W_2]^* = \arg\min_{[W_1, W_2] \in \mathbf{R}^{m\times 2m}} ||P - [W_1 W_2] \begin{bmatrix} U \\ V \end{bmatrix} ||$$

[2] In practice training Mult this way achieves similar or lower errors in comparison to WAdd.

This is a multivariate linear regression problem (Hastie et al., 2009) for which the least squares estimate is given by: $[W_1, W_2] = ((X^T X)^{-1} X^T Y)^T$ where I use $X = [U^T, V^T]$ and $Y = P^T$.

LF The lexical function composition method learns a matrix representation for each functor (given by $\mathcal{U}$ here) and defines composition as matrix-vector multiplication. More precisely: $\vec{p} = A_u \vec{v}$ where A_u is a matrix associated to each functor $u \in \mathcal{U}$. I denote by T_u the training data subset associated to an element u, which contains only tuples which have u as first element. Learning the matrix representations amounts to solving the set of problems:

$$A_u = \underset{A_u \in \mathbf{R}^{m \times m}}{\arg\min} ||P_u - A_u V_u||$$

for each $u \in \mathcal{U}$ where $P_u, V_u \in \mathbf{R}^{m \times |T_u|}$ are the matrices corresponding to the T_u training subset. The solutions are given by: $A_u = ((V_u V_u^T)^{-1} V_u P_u^T)^T$. This composition function does not use the functor vectors.

FullLex This model can be seen as a generalization of LF which makes no assumption on which of the constituents is a functor, so that both words get a matrix and a vector representation. The composition function is:

$$\vec{p} = tanh([W_1, W_2] \begin{bmatrix} A_u \vec{v} \\ A_v \vec{u} \end{bmatrix})$$

where A_u and A_v are the matrices associated to constituents u and v and $[W_1, W_2] \in \mathbf{R}^{m \times 2m}$. The estimation problem is given in Figure 3-1.

$$\begin{aligned} W_1^*, W_2^*, A_{u_1}^*, ..., A_{v_1}^*, ... &= \underset{\mathbf{R}^{m \times m}}{\arg\min} ||atanh(P^T) - [W_1, W_2] \begin{bmatrix} [A_{u_1} \vec{v_1}, ..., A_{u_k} \vec{v_k}] \\ [A_{v_1} \vec{u_1}, ..., A_{v_k} \vec{u_k}] \end{bmatrix} ||_F \\ &= \underset{\mathbf{R}^{m \times m}}{\arg\min} ||atanh(P^T) - W_1 [A_{u_1} \vec{v_1}, ..., A_{u_k} \vec{v_k}] - W_2 [A_{v_1} \vec{u_1}, ..., A_{v_k} \vec{u_k}]||_F \end{aligned}$$

Figure 3-1: FullLex estimation problem.

This is the only composition model which does not have a closed-form solution. I use

a block coordinate descent method, in which I fix each of the matrix variables but one and solve the corresponding least-squares linear regression problem, for which I can use the closed-form solution. Fixing everything but $[W_1, W_2]$:

$$[W_1^*, W_2^*] = ((X^T X)^{-1} X^T Y)^T$$
$$X = \begin{bmatrix} [A_{u_1}\vec{v_1}, ..., A_{u_k}\vec{v_k}] \\ [A_{v_1}\vec{u_1}, ..., A_{v_k}\vec{u_k}] \end{bmatrix}^T$$
$$Y = atanh(P^T)$$

Fixing everything but A_u for some element u, the objective function becomes:

$$||atanh(P_u) - W_1 A_u V_u - W_2[A_{v_1}\vec{u}, ..., A_{v_{k'}}\vec{u}]||_F$$

where $v_1 ... v_{k'} \in \mathcal{V}$ are the elements occurring with u in the training data and V_u the matrix resulting from their concatenation. The update formula for the A_u matrices becomes:

$$A_u^* = W_1^{-1}((X^T X)^{-1} X^T Y)^T$$
$$X = V_u^T$$
$$Y = (atanh(P_u) - W_2[A_{v_1}\vec{u}, ..., A_{v_{k'}}\vec{u}])^T$$

In all the experiments, FullLex estimation converges after very few passes though the matrices. Despite the very large number of parameters of this model, when evaluating on the test data I observe that using a higher dimensional space (such as 200 dimensions) still performs better than a lower dimensional one (e.g., 50 dimensions).

3.3 Evaluation setup and implementation

3.3.1 Datasets

I evaluate the composition methods on three phrase-based benchmarks that test the models on a variety of composition processes and similarity-based tasks.

Intransitive sentences The first dataset, introduced by Mitchell and Lapata (2008), focuses on simple sentences consisting of intransitive verbs and their noun subjects. It contains a total of 120 sentence pairs together with human similarity judgments on a 7-point scale. For example, *conflict erupts/conflict bursts* is scored 7, *skin glows/skin burns* is scored 1. On average, each pair is rated by 30 participants. Rather than evaluating against mean scores, I use each rating as a separate data point, as done by Mitchell and Lapata. I report Spearman correlations between human-assigned scores and model cosine scores.

Adjective-noun phrases Turney (2012) introduced a dataset including both noun-noun compounds and adjective-noun phrases (ANs). I focus on the latter, and I frame the task differently from Turney's original definition due to data sparsity issues.[3] In our version, the dataset contains 620 ANs, each paired with a single-noun paraphrase. Examples include: *archaeological site/dig*, *spousal relationship/marriage* and *dangerous undertaking/adventure*. I evaluate a model by computing the cosine of all 20K nouns in our semantic space with the target AN, and looking at the rank of the correct paraphrase in this list. The lower the rank, the better the model. I report median rank across the test items.

Determiner phrases The last dataset, introduced in Bernardi et al. (2013), focuses on a class of *grammatical* terms (rather than content words), namely determiners. It is a multiple-choice test where target nouns (e.g., *amnesia)* must be matched with the most closely related determiner(-noun) phrases (DPs) (e.g., *no memory*). The task differs from the previous one also because here the targets are single words, and the related items are composite. There are 173 target nouns in total, each paired with one correct DP response, as well as 5 foils, namely the determiner (*no*) and noun (*memory*) from the correct response and three more DPs, two of which contain the same noun as the correct phrase (*less memory, all memory*), the third the same determiner (*no repertoire*). Other examples of targets/related-phrases are *polysemy/several senses* and *trilogy/three books*. The models compute cosines between target noun and responses and are scored based on their accuracy

[3]Turney used a corpus of about 50 billion words, almost 20 times larger than ours, and I have very poor or no coverage of many original items, making the "multiple-choice" evaluation proposed by Turney meaningless in our case.

at ranking the correct phrase first.

3.3.2 Input vectors

I extracted distributional semantic vectors using as source corpus the concatenation of ukWaC, Wikipedia (2009 dump) and BNC, 2.8 billion tokens in total.[4] I use a bag-of-words approach and I count co-occurrences within sentences and with a limit of maximally 50 words surrounding the target word. By tuning on the MEN lexical relatedness dataset,[5] I decided to use the top 10K most frequent content lemmas as context features (vs. top 10K inflected forms), and I experimented with positive Pointwise and Local Mutual Information (Evert, 2005) as association measures (vs. raw counts, log transform and a probability ratio measure) and dimensionality reduction by NMF (Lee and Seung, 2000) and SVD (Golub and Van Loan, 1996) (both outperforming full dimensionality vectors on MEN). For both reduction techniques, I varied the number of dimensions to be preserved from 50 to 300 in 50-unit intervals. As Local Mutual Information performed very poorly across composition experiments and other parameter choices, I dropped it. I will thus report, for each experiment and composition method, the distribution of the relevant performance measure across 12 input settings (NMF vs. SVD times 6 dimensionalities). However, since the Mult model, as expected, worked very poorly when the input vectors contained negative values, as is the case with SVD, for this model I report result distributions across the 6 NMF variations only.

3.3.3 Composition model estimation

Training by approximating the corpus-extracted phrase vectors requires corpus-based examples of input (constituent word) and output (phrase) vectors for the composition processes to be learned. In all cases, training examples are simply selected based on corpus frequency. For the first experiment, I have 42 distinct target verbs and a total of ≈20K training instances, that is, $\langle\langle$*noun*, *verb*$\rangle$, *noun-verb*$\rangle$ tuples (505 per verb on average). For

[4] `http://wacky.sslmit.unibo.it`; `http://www.natcorp.ox.ac.uk`

[5] `http://clic.cimec.unitn.it/~elia.bruni/MEN`

the second experiment, I have 479 adjectives and ≈1 million $\langle\langle$*adjective*, *noun*$\rangle$, *adjective-noun*$\rangle$ training tuples (2K per adjective on average). In the third, 50 determiners and 50K $\langle\langle$*determiner*, *noun*$\rangle$, *determiner-noun*$\rangle$ tuples (1K per determiner). For all models except LF and FullLex, training examples are pooled across target elements to learn a single set of parameters. The LF model takes only argument word vectors as inputs (the functors in the three datasets are verbs, adjectives and determiners, respectively). A separate weight matrix is learned for each functor, using the corresponding training data.[6] The FullLex method jointly learns distinct matrix representations for both left- and right-hand side constituents. For this reason, I must train this model on balanced datasets. More precisely, for the intransitive verb experiments, I use training data containing noun-verb phrases in which the verbs and the nouns are present in the lists of 1,500 most frequent verbs/nouns respectively, adding to these the verbs and nouns present in our dataset. I obtain 400K training tuples. I create the training data similarity for the other datasets obtaining 440K adjective-noun and 50K determiner phrase training tuples, respectively (I also experimented with FullLex trained on the same tuples used for the other models, obtaining considerably worse results than those reported). Finally, for Dil I treat direction of stretching as a further parameter to be optimized, and find that for intransitives it is better to stretch verbs, in the other datasets nouns.

For the simple composition models for which parameters consist of one or two scalars, namely WAdd, Mult and Dil, I also tune the parameters through 5-fold crossvalidation on the datasets, directly optimizing the parameters on the target tasks. For WAdd and Mult, I search w_1, w_2 through the crossproduct of the interval $[0 : 5]$ in 0.2-sized steps. For Dil I use $\lambda \in [0 : 20]$, again in 0.2-sized steps.

3.4 Evaluation results

[6]For the LF model I have experimented with least squeares regression with and without regularization, obtaining similar results.

[8]For intransitive sentences, figure of merit is Spearman correlation, for ANs median rank of correct paraphrase, and for DPs correct response accuracy. The boxplots display the distribution median as a thick horizontal line within a box extending from first to third quartile. Whiskers cover 1.5 of interquartile range in each direction from the box, and extreme outliers outside this extended range are plotted as circles.

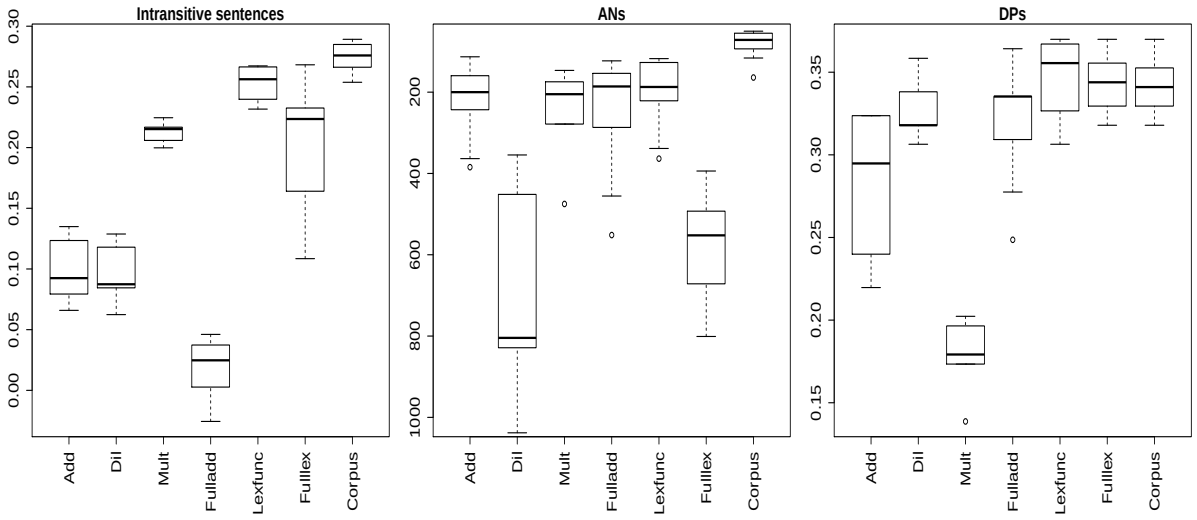


Figure 3-2: Boxplots displaying composition model performance distribution on three benchmarks, across input vector settings (6 datapoints for Mult, 12 for all other models).[8]

I begin with some remarks pertaining to the overall quality of and motivation for corpus-phrase-based estimation. In seven out of nine comparisons of this unsupervised technique with fully supervised crossvalidation (3 "simple" models –WAdd, Dil and Mult– times 3 test sets), there was no significant difference between the two estimation methods.[9] Supervised estimation outperformed the corpus-phrase-based method only for Dil on the intransitive sentence and AN benchmarks, but crossvalidated Dil was outperformed by at least one phrase-estimated simple model on both benchmarks.

The rightmost boxes in the panels of Figure 3-2 depict the performance distribution for using phrase vectors directly extracted from the corpus to tackle the various tasks. This non-compositional approach outperforms all compositional methods in two tasks over three, and it is one of the best approaches in the third, although in all cases even its top scores are far from the theoretical ceiling. Still, performance is impressive, especially in light of the fact that the non-compositional approach suffers of serious data-sparseness problems. Performance on the intransitive task is above state-of-the-art despite the fact that for almost half of the cases one test phrase is not in the corpus, resulting in 0 vectors and consequently 0 similarity pairs. The other benchmarks have better corpus-phrase coverage (nearly perfect AN coverage; for DPs, about 90% correct phrase responses are in the corpus), but many

[9]Significance assessed through Tukey Honestly Significant Difference tests (Abdi and Williams, 2010), $\alpha = 0.05$.

target phrases occur only rarely, leading to unreliable distributional vectors. I interpret these results as a good motivation for corpus-phrase-based estimation. On the one hand they show how good these vectors are, and thus that they are sensible targets of learning. On the other hand, they do not suffice, since natural language is infinitely productive and thus no corpus can provide full phrase coverage, justifying the whole compositional enterprise.

The other boxes in Figure 3-2 report the performance of the composition methods trained by corpus phrase approximation. Nearly all models are significantly above chance in all tasks, except for FullAdd on intransitive sentences. To put AN median ranks into perspective, consider that a median rank as high as 8,300 has near-0 probability to occur by chance. For DP accuracy, random guessing gets 0.17% accuracy.

LF emerges consistently as the best model. On intransitive constructions, it significantly outperforms all other models except Mult, but the difference approaches significance even with respect to the latter ($p = 0.071$). On this task, LF's *median* correlation (0.26) is nearly equivalent to the *best* correlation across a wide range of parameters reported by Erk and Padó (2008) (0.27). In the AN task, LF significantly outperforms FullLex and Dil and, visually, its distribution is slightly more skewed towards lower (better) ranks than any other model. In the DP task, LF significantly outperforms WAdd and Mult and, visually, most of its distribution lies above that of the other models. Most importantly, LF is the only model that is consistent across the three tasks, with all other models displaying instead a brittle performance pattern.[10]

Still, the top-performance range of all models on the three tasks is underwhelming, and none of them succeeds in exploiting compositionality to do significantly better than using whatever phrase vectors can be extracted from the corpus directly. Clearly, much work is still needed to develop truly successful cDSMs.

The AN results might look particularly worrying, considering that even the top (lowest) median ranks are above 100. A qualitative analysis, however, suggests that the actual performance is not as bad as the numerical scores suggest, since often the nearest neighbours of the ANs to be paraphrased are nouns that are as strongly related to the ANs as

[10]No systematic trend emerged pertaining to the input vector parameters (SVD vs. NMF and retained dimension number).

the gold standard response (although not necessarily proper paraphrases). For example, the gold response to *colorimetric analysis* is *colorimetry*, whereas the LF (NMF, 300 dimensions) nearest neighbour is *chromatography*; the gold response to *heavy particle* is *baryon*, whereas LF proposes *muon*; for *melodic phrase* the gold is *tune* and LF has *appoggiatura*; for *indoor garden*, the gold is *hothouse* but LF proposes *glasshouse* (followed by the more sophisticated *orangery*!), and so on and so forth.

3.5 Summary

In this chapter, I extended the unsupervised corpus-extracted phrase approximation method of Guevara (2010) and Baroni and Zamparelli (2010) to estimate all known state-of-the-art cDSMs, using closed-form solutions or simple iterative procedures in all cases. Equipped with a general estimation approach, I thoroughly evaluated the cDSMs in a comparable setting. The linguistically motivated LF model of Baroni and Zamparelli (2010) and Coecke et al. (2010) was the winner across three composition tasks, also outperforming the more complex FullLex model, our re-implementation of Socher et al. (2012)'s composition method (of course, the composition method is only one aspect of Socher et al.'s architecture). All other composition methods behaved inconsistently.

Chapter 4

Practical Lexical Function

4.1 The lexical function model

In the general evaluation in the previous chapter, I have established that the LF model is the most stable across different types of phrases. It is also a model that captures the intuition of composition from formal semantics, thus is suitable for further development into a full-fledged model of sentential meaning.

The full range of semantic types required for natural language processing, including those of adverbs and transitive verbs, has to include, however, tensors of greater rank. The estimation method originally proposed by Baroni and Zamparelli (2010); Guevara (2010) has been extended to 3-way tensors representing transitive verbs by Grefenstette et al. (2013) with preliminary success. Grefenstette et al.'s method works in two steps. First, one estimates matrices of verb-object phrases from subject and subject-verb-object vectors; next, transitive verb tensors are estimated from verb-object matrices and object vectors.

4.1.1 Problems with the extension of the lexical function model to sentences

With all the advantages of LF, scaling it up to arbitrary sentences, leads to several issues. In particular, it is desirable for all practical purposes to limit representation size. For example,

if noun meanings are encoded in vectors of 300 dimensions, adjectives become matrices of 300^2 cells, and transitive verbs are represented as tensors with 300^3=$27,000,000$ dimensions.

Estimating tensors of this size runs into data sparseness issues already for less common transitive verbs. Indeed, in order to train a transitive verb tensor (e.g., *eat*), the method of Grefenstette et al. (2013) requires a sufficient number of distinct verb object phrases with that verb (e.g., *eat cake*, *eat fruits*), each attested in combination with a certain number of subject nouns with sufficient frequency to extract sensible vectors. It is not feasible to obtain enough data points for all verbs in such a training design.

Things get even worse for other categories. Adverbs like *quickly* that modify intransitive verbs have to be represented with $300^{2^2} = 8,100,000,000$ dimensions. Modifiers of transitive verbs would have even greater representation size, which may not be possible to store and learn efficiently.

Another issue is that the same or similar items that occur in different syntactic contexts are assigned different semantic types with incomparable representations. For example, verbs like *eat* can be used in transitive or intransitive constructions (*children eat meat*/*children eat*), or in passive (*meat is eaten*). Since predicate arity is encoded in the order of the corresponding tensor, *eat* and the like have to be assigned different representations (matrix or tensor) depending on the context. Deverbal nouns like *demolition*, often used without mention of who demolished what, would have to get vector representations while the corresponding verbs (*demolish*) would become tensors, which makes immediately related verbs and nouns incomparable. Nouns in general would oscillate between vector and matrix representations depending on argument vs. predicate vs. modifier position (*an animal runs* vs. *this is an animal* vs. *animal shelter*). Prepositions are the hardest, as the syntactic positions in which they occur are most diverse (*park in the dark* vs. *play in the dark* vs. *be in the dark* vs. *a light glowing in the dark*).

In all those cases, the same word has to be mapped to tensors of different orders. Since each of these tensors must be learned from examples individually, their obvious relation is missed. Besides losing the comparability of the semantic contribution of a word across syntactic contexts, I also worsen the data sparseness issues.

The last, and related, point is that for the tensor calculus to work, one needs to model, for each word, each of the constructions in the corpus that the word is attested in. In its pure form LF does not include an emergency backoff strategy when unknown words or constructions are encountered. For example, if we only observe transitive usages of *to eat* in the training corpus, and encounter an intransitive or passive example of it in testing data, the system would not be able to compose a sentence vector at all. This issue is unavoidable since we don't expect to find all words in all possible constructions even in the largest corpus.

4.2 The practical lexical function model

As follows from section 4.1.1, it would be desirable to have a compositional distributional model that encodes function-argument relations but avoids the troublesome high-order tensor representations of the pure lexical function model, with all the practical problems that come with them. We may still want to represent word meanings in different syntactic contexts differently, but at the same time we need to incorporate a formal connection between those representations, e.g., between the transitive and the intransitive instantiations of the verb *to eat*. Last but not least, all items need to include a common aspect of their representation (e.g., a vector) to allow comparison across categories (the case of *demolish* and *demolition*).

To this end, I propose a new model of composition that maintains the idea of function application, while avoiding the complications and rigidity of LF. I call our proposal *practical lexical function* model, or *PLF*. In PLF, a functional word is not represented by a single tensor of arity-dependent order, but by a vector plus an ordered set of matrices, with one matrix for each argument the function takes. After applying the matrices to the corresponding argument vectors, a single representation is obtained by summing across all resulting vectors.

4.2.1 Word meaning representation

In PLF, all words are represented by a vector, and functional words, such as predicates and modifiers, are also assigned one or more matrices. The general form of a semantic representation for a linguistic unit is an ordered tuple of a vector and $n \in \mathbb{N}$ matrices:[1]

$$\left\langle \overrightarrow{x}, \overset{\square_1}{x}, \ldots, \overset{\square_n}{x} \right\rangle$$

The number of matrices in the representation encodes the arity of a linguistic unit, i.e., the number of other units to which it applies as a function. Each matrix corresponds to a function-argument relation, and words have as many matrices as many arguments they take: none for (most) nouns, one for adjectives and intransitive verbs, two for transitives, etc. The matrices formalize argument slot saturation, operating on an argument vector representation through matrix by vector multiplication, as described in the next section.

Modifiers of n-ary functors are represented by n+1-ary structures. For instance, I treat adjectives that modify nouns (0-ary) as unary functions, encoded in a vector-matrix pair. Adverbs have different semantic types depending on their syntactic role. Sentential adverbs are unary, while adverbs that modify adjectives (*very*) or verb phrases (*quickly*) are encoded as binary functions, represented by a vector and two matrices. The form of semantic representations I am using is shown in Table 4.1.[2]

4.2.2 Semantic composition

My system incorporates semantic composition via two composition rules, one for combining structures of different arity and the other for symmetric composition of structures with the same arity. These rules incorporate insights of two empirically successful models, lexical function and the simple additive approach, used as the default structure merging strategy.

The first rule is *function application*, illustrated in Figure 4-1. Table 4.2 illustrates

[1] Matrices associated with term *x* are symbolized $\overset{\square}{x}$.

[2] To determine the number and ordering of matrices representing the word in the current syntactic context, our PLF implementation relies on the syntactic type assigned to the word in the categorial grammar parse of the sentence.

dog	$\overrightarrow{dog}$
run	$\overrightarrow{run}, \overset{\square}{run}$
chase	$\overrightarrow{chase}, \overset{\square_s}{chase}, \overset{\square_o}{chase}$
give	$\overrightarrow{give}, \overset{\square_s}{give}, \overset{\square_o}{give}, \overset{\square_{io}}{give}$
big	$\overrightarrow{big}, \overset{\square}{big}$
very	$\overrightarrow{very}, \overset{\square_n}{very}, \overset{\square_a}{very}$
quickly	$\overrightarrow{quickly}, \overset{\square_s}{quickly}, \overset{\square_v}{quickly}$

Table 4.1: Examples of word representations. Subscripts encode, just for mnemonic purposes, the constituent whose vector the matrix combines with: **s**ubject, **o**bject, **i**ndirect **o**bject, **n**oun, **a**djective, **v**erb phrase.

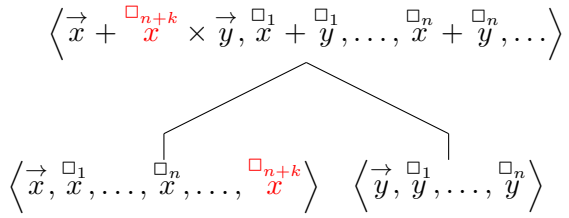


Figure 4-1: Function application: If two syntactic sisters have different arity, treat the higher-arity sister as the functor. Compose by multiplying the last matrix in the functor tuple by the argument vector and summing the result to the functor vector. Unsaturated matrices are carried up to the composed node, summing across sisters if needed.

simple cases of function application. For transitive verbs semantic composition applies iteratively as shown in the derivation of Figure 4-2. For ternary predicates such as *give* in a ditransitive construction, the first step in the derivation absorbs the innermost argument by multiplying its vector by the third *give* matrix, and then composition proceeds like for transitives.

The second composition rule, *symmetric composition* applies when two syntactic sisters are of the same arity (e.g., two vectors, or two vector-matrix pairs). Symmetric composition simply sums the objects in the two tuples: vector with vector, n-th matrix with n-th matrix.

Symmetric composition is reserved for structures in which the function-argument distinction is problematic. Some candidates for such treatment are coordination and nominal compounds, although I recognize that the headless analysis is not the only possible one here. See two examples of Symmetric Composition application in Table 4.3.

Note that the *sing and dance* composition in Table 4.3 skips the conjunction. My

dogs	$\overrightarrow{dogs}$
run	$\overrightarrow{run}, \overset{\square}{run}$
dogs run	$\overrightarrow{run} + \overset{\square}{run} \times \overrightarrow{dog}$
house	$\overrightarrow{house}$
big	$\overrightarrow{big}, \overset{\square}{big}$
big house	$\overrightarrow{big} + \overset{\square}{big} \times \overrightarrow{house}$

Table 4.2: Examples of function application.

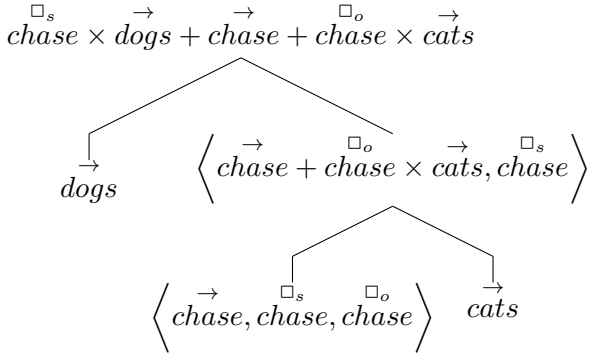


Figure 4-2: Applying function application twice to derive the representation of a transitive sentence.

current PLF implementation treats most grammatical words, including conjunctions, as "empty" elements, that do not project into semantics. This choice leads to some interesting "serendipitous" treatments of various constructions. For example, since the copula is empty, a sentence with a predicative adjective (*cars are red*) is treated in the same way as a phrase with the same adjective in attributive position (*red cars*) – although the latter, being a phrase and not a full sentence, will later be embedded as argument in a larger construction. Similarly, leaving the relative pronoun empty makes *cars that run* identical to *cars run*, although, again, the former will be embedded in a larger construction later in the

sing: $\overrightarrow{sing}, \overset{\square}{sing}$	dance: $\overrightarrow{dance}, \overset{\square}{dance}$
sing and dance:	$\overrightarrow{sing} + \overrightarrow{dance}, \overset{\square}{sing} + \overset{\square}{dance}$
rice: $\overrightarrow{rice}$	cake: $\overrightarrow{cake}$
rice cake	$\overrightarrow{rice} + \overrightarrow{cake}$

Table 4.3: Examples of symmetric composition.

derivation.

I conclude my brief exposition of PLF with an alternative intuition for it: the PLF model is also a more sophisticated version of the additive approach, where argument words are adapted by matrices that encode the relation to their functors before the sentence vector is derived by summing.

4.2.3 Satisfying the desiderata

Let us now outline how PLF addresses the shortcomings of LF listed in Section 4.1.1. First, all issues caused by representation size disappear. An n-ary predicate is no longer encoded as an n+1-way tensor; instead I have a sequence of n matrices. The representation size grows linearly, not exponentially, for higher semantic types, allowing for simpler and more efficient parameter estimation, storage, and computation.

As a consequence of this architecture, we no longer need to perform the complicated step-by-step estimation for elements of higher arity. Indeed, one can estimate each matrix of a complex representation individually using the simple method of Baroni and Zamparelli (2010). For instance, for transitive verbs I estimate the verb-subject combination matrix from subject and verb-subject vectors, the verb-object combination matrix from object and verb-object vectors. I expect a reasonably large corpus to feature many occurrences of a verb with a variety of subjects and a variety of objects (but not necessarily a variety of subjects with each of the objects as required by Grefenstette et al.'s training), allowing us to avoid the data sparseness issue.

The semantic representations I propose include a semantic vector for constituents of any semantic type, thus enabling semantic comparison for words of different parts of speech (the case of *demolition* vs. *demolish*).

Finally, the fact that I represent the predicate interaction with each of its arguments in a separate matrix allows for a natural and intuitive treatment of argument alternations. For instance, as shown in Table 4.4, one can distinguish the transitive and intransitive usages of the verb *to eat* by the presence of the object-oriented matrix of the verb while keeping the rest of the representation intact. To model passive usages, I insert the object matrix of the

boys	$\overrightarrow{boys}$
eat (intrans.)	$\overrightarrow{eat}, \overrightarrow{\square_s eat}$
boys eat	$\overrightarrow{\square_s eat} \times \overrightarrow{boys} + \overrightarrow{eat}$
meat	$\overrightarrow{meat}$
eat (trans.)	$\overrightarrow{eat}, \overrightarrow{\square_s eat}, \overrightarrow{\square_o eat}$
boys eat meat	$\overrightarrow{\square_s eat} \times \overrightarrow{boys} + \overrightarrow{eat} + \overrightarrow{\square_o eat} \times \overrightarrow{meat}$
(is) eaten (pass.)	$\overrightarrow{eat}, \overrightarrow{\square_o eat}$
meat is eaten	$\overrightarrow{eat} + \overrightarrow{\square_o eat} \times \overrightarrow{meat}$

Table 4.4: The verb *to eat* associated to different sets of matrices in different syntactic contexts.

verb only, which will be multiplied by the syntactic subject vector, capturing the similarity betIen *eat meat* and *meat is eaten*.

So keeping the verb's interaction with subject and object encoded in distinct matrices not only solves the issues of representation size for arbitrary semantic types, but also provides a sensible built-in strategy for handling a word's occurrence in multiple constructions. Indeed, if we encounter a verb used intransitively which was only attested as transitive in the training corpus, we can simply omit the object matrix to obtain a type-appropriate representation. On the other hand, if the verb occurs with more arguments than usual in testing materials, we can add a default diagonal identity matrix to its representation, signaling agnosticism about how the verb relates to the unexpected argument. This flexibility makes my model suitable to compute vector representations of sentences without stumbling at unseen syntactic usages of words.

To summarize, PLF is an extension of the lexical function model that inherits its strengths and overcomes its weaknesses. I still employ a linguistically-motivated notion of semantic composition as function application and use distinct kinds of representations for different semantic types. At the same time, I avoid high order tensor representations, produce semantic vectors for all syntactic constituents, and allow for an elegant and transparent correspondence between different syntactic usages of a lexeme, such as the transitive, the intransitive, and the passive usages of the verb *to eat*. Last but not least, my implementation is suitable for realistic language processing since it allows to produce vectors for sentences

of arbitrary size, including those containing novel syntactic configurations.

4.3 Evaluation

4.3.1 Evaluation materials

Here I want to evaluate the models on sentence level, instead of phrase level like in chapter 3. Therefore, I consider 5 different benchmarks that focus on different aspects of sentence-level semantic composition. The first data set, created by Edward Grefenstette and Mehrnoosh Sadrzadeh and introduced in Kartsaklis et al. (2013), features 200 sentence pairs that were rated for similarity by 43 annotators. In this data set, sentences have fixed adjective-noun-verb-adjective-noun (anvan) structure, and they were built in order to crucially require context-based verb disambiguation (e.g., *young woman filed long nails* is paired with both *young woman smoothed long nails* and *young woman registered long nails*). I also consider a similar data set introduced by Grefenstette (2013), comprising 200 sentence pairs rated by 50 annotators. I will call these benchmarks **anvan1** and **anvan2**, respectively. Evaluation is carried out by computing the Spearman correlation between the annotator similarity ratings for the sentence pairs and the cosines of the vectors produced by the various systems for the same sentence pairs.

The benchmark introduced by Pham et al. (2013) at the TFDS workshop (**tfds** below) was specifically designed to test compositional methods for their sensitivity to word order and the semantic effect of determiners. The tfds benchmark contains 157 target sentences that are matched with a set of (approximate) paraphrases (8 on average), and a set of "foils" (17 on average). The foils have high lexical overlap with the targets but very different meanings, due to different determiners and/or word order. For example, the target *A man plays an acoustic guitar* is matched with paraphrases such as *A man plays guitar* and *The man plays the guitar*, and foils such as *The man plays no guitar* and *A guitar plays a man*. A good system should return higher similarities for the comparison with the paraphrases with respect to that with the foils. Performance is assessed through the t-standardized cross-target average of the difference between mean cosine with paraphrases and mean

cosine with foils (Pham and colleagues, equivalently, reported non-standardized average and standard deviations).

The two remaining data sets are larger and more 'natural', as they were not constructed by linguists under controlled conditions to focus on specific phenomena. They are aimed at evaluating systems on the sort of free-form sentences one encounters in real-life applications. The **msrvid** data set from the SemEval-2012 Semantic Textual Similarity (STS) task (Agirre et al., 2012) consists of 750 sentence pairs that describe brief videos. Sentence pairs were scored for similarity by 5 subjects each. Following standard practice in paraphrase detection studies (e.g., Blacoe and Lapata (2012)), I use cosine similarity between sentence pairs as computed by one of my systems together with two shallow similarity cues: word overlap between the two sentences and difference in sentence length. I obtain a final similarity score by weighted addition of the 3 cues, with the optimal weights determined by linear regression on separate msrvid train data that were also provided by the SemEval task organizers (before combining, I checked that the collinearity between cues was low). System scores are evaluated by their Pearson correlation with the human ratings.

The final set I use is **onwn**, from the *SEM-2013 STS shared task (Agirre et al., 2013). This set contains 561 pairs of glosses (from the WordNet and OntoNotes databases), rated by 5 judges for similarity. My main interest in this set stems from the fact that glosses are rarely well-formed full sentences (consider, e.g., *cause something to pass or lead somewhere*; *coerce by violence, fill with terror*). For this reason, they are very challenging for standard parsers. Indeed, I estimated from a sample of 40 onwn glosses that the C&C parser (see below) has only 45% accuracy on this set. Since *PLF* needs syntactic information to construct sentence vectors compositionally, I test it on onwn to make sure that it is not overly sensitive to parser noise. Evaluation proceeds as with msrvid (cue weights are determined by 10-fold cross-validation).[3]

[3] I did not evaluate on other STS benchmarks since they have characteristics, such as high density of named entities, that would require embedding my compositional models into more complex systems, obfuscating their impact on the overall performance.

4.3.2 Semantic space construction and composition model implementation

My source corpus was given by the concatenation of ukWaC (`wacky.sslmit.unibo.it`), a mid-2009 dump of the English Wikipedia (`en.wikipedia.org`) and the British National Corpus (`www.natcorp.ox.ac.uk`), for a total of about 2.8 billion words.

I collected a 30K-by-30K matrix by counting co-occurrence of the 30K most frequent content lemmas (nouns, adjectives and verbs) within a 3-word window. The raw count vectors were transformed into positive Pointwise Mutual Information scores and reduced to 300 dimensions by the Singular Value Decomposition. All vectors were normalized to length 1. This setup was picked without tuning, as I found it effective in previous, unrelated experiments.[4]

I consider four composition models: Add, Mult, LF and the new PLF model. Of all the models discussed in chapters 2 and 3, Add and Mult are selected for their simplicity and effectiveness. For Add and Mult, I only sum or multiply the vectors of the content words in the sentences.

For the LF model, I construct functional matrix representations of adjectives, determiners and intransitive verbs. These are trained using Ridge regression with generalized cross-validation from corpus-extracted vectors of nouns, as input, and phrases including those nouns as output (e.g., the matrix for *red* is trained from corpus-extracted $\langle$*noun*, *red-noun*$\rangle$ vector pairs). Transitive verb tensors are estimated using the two-step regression procedure outlined by Grefenstette et al. (2013). I did not attempt to train a LF model for the larger and more varied msrvid and onwn data sets, as this would have been extremely time consuming and impractical for all the reasons I discussed in Section 4.1.1 above.

Training PLF proceeds similarly, but I also build preposition matrices (from $\langle$*noun*, *preposition-noun*$\rangle$ vector pairs), and for verbs I prepare separate subject and object matrices.

Since syntax guides LF and PLF composition, I supplied all test sentences with cat-

[4]With the multiplicative composition model I also tried Nonnegative Matrix Factorization instead of Singular Value Decomposition, because the negative values produced by SVD are potentially problematic for Mult. In addition, I repeated the evaluation for the multiplicative and additive models without any form of dimensionality reduction. The overall pattern of results did not change significantly, and thus for consistency I report all models' performance only for the SVD-reduced space.

models	anvan 1	anvan 2	tfds	msr vid	onwn
Add	8	22	-0.2	78	66
Mult	8	-4	-2.3	77	55
LF	15	30	**5.90**	NA	NA
PLF	**20**	**36**	2.7	**79**	**67**
soa	22	27	11.4	87	75
baseline	8	22	7.9	77	55

Table 4.5: Performance of composition models on all evaluation sets. Figures of merit follow previous art on each set and are: percentage Spearman coefficients for anvan1 and anvan2, t-standardized average difference between mean cosines with paraphrases and with foils for tfds, percentage Pearson coefficients for msrvid and onwn.

egorial grammar parses. Every sentence in the anvan1 and anvan2 datasets has the form (subject) Adjective + Noun + Transitive Verb + (object) Adjective + Noun, so parsing them is trivial. All sentences in tfds have a predictable structure that allows perfect parsing with simple finite state rules. In all these cases, applying a general-purpose parser to the data would have, at best, had no impact and, at worst, introduced parsing errors. For msrvid and onwn, I used the output of the C&C parser (Clark and Curran, 2007).

4.3.3 Results

Table 4.5 summarizes the performance of my models on the chosen tasks, and compares it to the state of the art reported in previous work, as well as to various strong baselines.

The PLF model performs very well on both anvan benchmarks, outperforming not only Add and Mult, but also the full-fledged LF model. Given that these data sets contain, systematically, transitive verbs, the major difference between PLF and LF lies in their representation of the latter. Evidently, the separately-trained subject and object matrices of PLF, being less affected by data sparseness than the 3-way tensors of LF, are better able to capture how verbs interact with their arguments. For anvan1, PLF is just below the state of the art, which is based on disambiguating the verb vector in context (Kartsaklis and Sadrzadeh, 2013), and LF outperforms the baseline, which consists in using the verb vector only as a proxy to sentence similarity.[5] On anvan2, PLF outperforms the best model

[5]I report state of the art from Kartsaklis and Sadrzadeh (2013) rather than Kartsaklis et al. (2013), since

reported by Grefenstette (2013) (an implementation of the lexical function ideas along the lines of Grefenstette and Sadrzadeh (Grefenstette and Sadrzadeh, 2011a,b)). And LF is, again, the only model, besides PLF, that performs better than the baseline.

In the tfds task, not surprisingly the Add and Mult models, lacking determiner representations and being order-insensitive, fail to distinguish between true paraphrases and foils (indeed, for the Mult model foils are significantly *closer* to the targets than the paraphrases, probably because the latter have lower content word overlap than the foils, that often differ in word order and determiners only). My PLF approach is able to handle determiners and word order correctly, as demonstrated by a highly significant ($p < 0.01$) difference between paraphrase and foil similarity (average difference in cosine .017, standard deviation .077). In this case, however, the traditional LF model (average difference .044, standard deviation .092) outperforms PLF. Since determiners are handled identically under the two approaches, the culprit must be word order. I conjecture that the LF 3-way tensor representation of transitive verbs leads to a stronger asymmetry between sentences with inverted arguments, and thus makes this model particularly sensitive to word order differences. Indeed, if we limit evaluation to those foils characterized by word order changes only, LF discriminates between paraphrases and foils even more clearly, whereas the PLF difference, while still significant, decreases slightly.

The state-of-the-art row for tfds reports the LF implementation by Pham et al. (2013), which outperforms PLF. The main difference is that Pham and colleagues do not normalize vectors like I do. If we don't normalize, we do get larger differences for the models as well, but consistently lower performance in all other tasks. More worryingly, the simple word overlap baseline reported in the table sports a larger difference than our best model. Clearly, this baseline is exploiting the systematic determiner differences in the foils and, indeed, when it is evaluated on foils where only word order changes its performance is no longer significant.

On msrvid, the PLF approach outperforms Add and Mult, although the difference between the three is not big. Our result stands in contrast with Blacoe and Lapata (2012), the only study I am aware of that compared a sophisticated composition model (Socher et

only the former used a source corpus that is comparable to mine.

al.'s 2011 model) to Add and Mult on realistic sentences, which attained the top performance with the simple models for both figures of merit they used.[6] The best 2012 STS system (Bär et al., 2012), obtained 0.87 correlation, but with many more and considerably more complex features than the ones I used here. Indeed, our simple system would have obtained a respectable 25/89 ranking in the STS 2012 msrvid task. Still, I must also stress the impressive performance of our baseline, given by the combination of the word overlap and sentence length cues. This suggests that the msrvid benchmark lacks the lexical and syntactic variety I would like to test our systems on.

Our PLF model is again the best on the onwn set (albeit by a small margin over Add). This is a very positive result, in the light of the fact that the parser has very low performance on the onwn glosses, thus suggesting that PLF can produce sensible semantic vectors from noisy syntactic representations. Here the overlap+length baseline does not perform so well, and again the best STS 2013 system (Han et al., 2013) uses considerably richer knowledge sources and algorithms than ours. Our PLF-based method would have reached a respectable 20/90 rank in the STS 2013 onwn task.

As a final remark, in all experiments the running time of PLF was only slightly larger than for the simpler models, but orders of magnitude smaller than LF, confirming another practical side of our approach.

4.4 Summary

I introduced an approach to compositional distributional semantics based on a linguistically-motivated syntax-to-semantics type mapping, but simple and flexible enough that it can produce representations of English sentences of arbitrary size and structure.

I showed that this approach is competitive against the more complex lexical function model when evaluated on the simple constructions the latter can be applied to, and it outperforms the additive and multiplicative compositionality models when tested on more realistic benchmarks (where the full-fledged lexical function approach is difficult or impossible to

[6] I refer here to the results reported in the erratum available at `http://homepages.inf.ed.ac.uk/s1066731/pdf/emnlp2012erratum.pdf`. The Add/Mult advantage was even more marked in the original paper.

use), even in presence of strong noise in its syntactic input.

One of the strengths of this framework is that it allows for incremental improvement focused on specific constructions. For example, one could add representations for different conjunctions (*and* vs. *or*), train matrices for verb arguments other than subject and direct object, or include new types of modifiers into the model, etc.

While there is potential for local improvements, this framework, which extends and improves on existing compositional semantic vector models, has demonstrated its ability to account for full sentences in a principled and elegant way. My implementation of the model relies on simple and efficient training, works fast, and shows good empirical results.

Chapter 5

Chunk-based Smoothed Tree Kernels

5.1 Introduction

The PLF model, introduced in chapter 4, uses syntactic information to guide the composition model in composing sentence vectors. This way, syntactic information can also be implicitly encoded in the sentence representations. In this chapter, I explore an alternative approach, the Chunk-based Smooth Tree Kernels (CSTKs) method. Instead of simply using syntactic information as a guideline, CSTKs encode both the syntactic structure and semantic information into the meaning representations. Therefore, when comparing the meanings of sentences, CSTKs explicitly take into account both semantic similarity and syntactic similarity.

Convolution kernels are a popular way to exploit both syntactic and semantic information in NLP systems (Collins and Duffy, 2002; Moschitti et al., 2008). Convolution kernels are naturally defined on spaces where there exists a similarity function between terminal nodes. This feature has been used to integrate distributional semantics within tree kernels, which focus on syntactic similarity. This class of kernels is often referred to as *smoothed tree kernels* (Mehdad et al., 2010; Croce et al., 2011), yet, these models only use distributional vectors for single words, thus not exploiting the semantic similarity between simple non-recursive phrases, often referred to as chunks (Abney, 1996).

Compositional models: such as those discussed in chapters 2 and 3 can map text fragments to vectors, providing a distributional meaning for simple phrases or sentences.

Therefore, compositional models might be good and reliable models of meaning for chunks.

In this chapter, I present the *chunk-based smoothed tree kernels* (CSTKs) as a way to merge the two approaches: the smoothed tree kernels and the models for compositional distributional semantics. This approach overcomes the limitation of the smoothed tree kernels which only use vectors for single words by exploiting reliable compositional models over chunks. CSTKs are defined over a chunk-based syntactic subtrees where terminal nodes are words or word sequences. I experimented with CSTKs on data from the recognizing textual entailment challenge (Dagan et al., 2006) and I compared my CSTKs with other standard tree kernels and standard recursive compositions. Experiments show that CSTKs perform better on this supervised task than basic compositional models recursively applied at the sentence level and better than syntactic tree kernels.

The rest of this chapter is organized as follows. Section 5.2 describes the CSTKs. Section 5.3 reports on the experimental setting and on the results. And, finally, Section 5.4 gives a summary of the chapter.

5.2 Chunk-based Smoothed Tree Kernels

This section describes the new class of kernels. I first introduce the notion of the chunk-based syntactic subtree. Then, I describe the recursive formulation of the class of kernels.

5.2.1 Notation and preliminaries

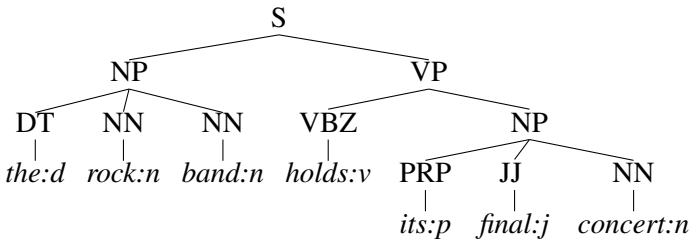


Figure 5-1: Sample Syntactic Tree

A *Chunk-based Syntactic Sub-Tree* is a subtree of a syntactic tree where each non-terminal node dominating a contiguous word sequence is collapsed into a chunk and, as

usual in chunks (Abney, 1996), the internal structure is disregarded. For example, Figure 5-2 reports some chunk-based syntactic subtrees of the tree in Figure 5-1. Chunks are represented with a pre-terminal node dominating a triangle that covers a word sequence. The first subtree represents the chunk covering the second NP and the node dominates the word sequence *its:d final:n concert:n*. The second subtree represents the structure of the whole sentence and one chunk, that is the first NP dominating the word sequence *the:d rock:n band:n*. The third subtree again represents the structure of the whole sentence split into two chunks without the verb.

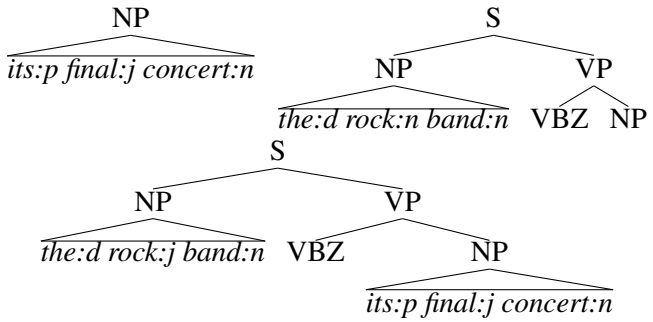


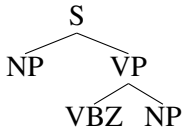
Figure 5-2: Some Chunk-based Syntactic Sub-Trees of the tree in Figure 5-1

In the following sections, generic trees are denoted with the letter t and $N(t)$ denotes the set of non-terminal nodes of tree t. Each non-terminal node $n \in N(t)$ has a label s_n representing its syntactic tag. As usual for constituency-based parse trees, pre-terminal nodes are nodes that have a single terminal node as child. Terminal nodes of trees are words denoted with *w:pos* where w is the actual token and pos is its postag. The structure of these trees is represented as follows. Given a tree t, $c_i(n)$ denotes i-th child of a node n in the set of nodes $N(t)$. The production rule headed in node n is $\mathsf{prod}(n)$, that is, given the node n with m children, $\mathsf{prod}(n)$ is:

$$\mathsf{prod}(n) = s_n \rightarrow s_{c_1(n)} \dots s_{c_m(n)}$$

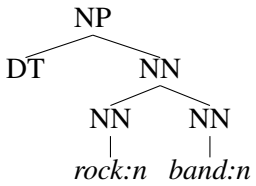
Finally, for a node n in $N(t)$, the function $d(n)$ generates the word sequence dominated by the non-terminal node n in the tree t. For example, *d(VP)* in Figure 5-1 is *holds:v its:p final:j concert:n*.

Chunk-based Syntactic Sub-Trees (CSSTs) are instead denoted with the letter τ. Differently from trees t, CSSTs have terminal nodes that can represent subsequences of words of the original sentence. The explicit syntactic structure of a CSST is the structure not falling in chunks and it is represented as $\mathtt{s}(\tau)$. For example, $\mathtt{s}(\tau_3)$ is:

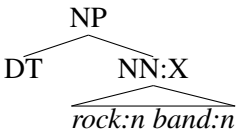


where τ_3 is the third subtree of Figure 5-2.

Given a tree t, the set $\mathcal{S}(t)$ is defined as the set containing all the relevant CSSTs of the tree t. As for the tree kernels (Collins and Duffy, 2002), the set $\mathcal{S}(t)$ contains all CSSTs derived from the subtrees of t such that if a node n belongs to a subtree t_s, all the siblings of n in t belongs to t_s. In other words, productions of the initial subtrees are complete. A CSST is obtained by collapsing in a single terminal nodes a contiguous sequence of words dominated by a single non-terminal node. For example:



is collapsed into:



Finally, $\vec{w_n} \in \mathbb{R}^m$ represent the *distributional* vectors for words w_n and $f(w_1 \dots w_k)$ represents a compositional distributional semantics model applied to the word sequence $w_1 \dots w_k$.

5.2.2 Smoothed Tree Kernels on Chunk-based Syntactic Trees

As usual, a tree kernel, although written in a recursive way, computes the following general equation:

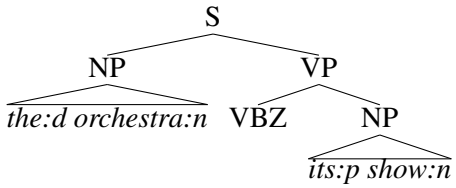
$$K(t_1, t_2) = \sum_{\substack{\tau_i \in \mathcal{S}(t_1) \\ \tau_j \in \mathcal{S}(t_2)}} \lambda^{|N(\tau_i)|+|N(\tau_j)|} K_F(\tau_i, \tau_j) \tag{5.1}$$

Here, the basic similarity $K_F(t_i, t_j)$ is defined to take into account the syntactic structure and the distributional semantic part. Thus, I define it as follows in line with what done with several other smoothed tree kernels:

$$K_F(\tau_i, \tau_j) = \delta(\mathsf{s}(\tau_i), \mathsf{s}(\tau_j)) \prod_{\substack{a \in PT(\tau_i) \\ b \in PT(\tau_j)}} \langle f(a), f(b) \rangle$$

where $\delta(\mathsf{s}(\tau_i), \mathsf{s}(\tau_j))$ is the Kroneker's delta function between the the structural part of two chunk-based syntactic subtrees, $PT(\tau)$ are the nodes in τ directly covering a chunk or a word, and $\langle \vec{x}, \vec{y} \rangle$ is the cosine similarity between the two vectors $\vec{x}$ and $\vec{y}$. For example, given the chunk-based subtree τ_3 in Figure 5-2 and

$\tau_4 =$



the similarity $K_F(\tau_3, \tau_4)$ is:

$\langle f(\textit{the:d orchestra:n}), f(\textit{the:d rock:n band:n}) \rangle \cdot \langle f(\textit{its:p show:n}), f(\textit{its:p final:j concert:n}) \rangle$

The recursive formulation of the Chunk-based Smoothed Tree Kernel (CSTK) is a bit more complex but very similar to the recursive formulation of the syntactic tree kernels:

$$K(t_1, t_2) = \sum_{\substack{n_1 \in N(t_1) \\ n_2 \in N(t_2)}} C(n_1, n_2) \tag{5.2}$$

where $C(n_1, n_2) =$

$$\begin{cases} \langle f(d(n_1)), f(d(n_2)) \rangle \text{ if } label(n_1) = label(n_2) \\ \quad \text{and } \mathsf{prod}(n_1) \neq \mathsf{prod}(n_2) \\ \langle f(d(n_1)), f(d(n_2)) \rangle \\ \quad + \prod_{j=1}^{nc(n_1)} (1 + C(c_j(n_1), c_j(n_2))) \\ \quad - \prod_{j=1}^{nc(n_1)} \langle f(d(c_j(n_1))), f(d(c_j(n_2))) \rangle \\ \quad \text{if } n_1, n_2 \text{ are not pre-terminals and} \\ \quad \mathsf{prod}(n_1) = \mathsf{prod}(n_2) \\ 0 \text{ otherwise} \end{cases}$$

where $nc(n_1)$ is the lenght of the production $\mathsf{prod}(n_1)$.

5.2.3 Compositional Distributional Semantic Models and two Specific CSTKs

To define specific CSTKs, we need to a specific composition model. Here I use two simple models: the additive model (Add) and the Full Additive model (FullAdd) for their simplicity (Mult is not selected as it is similar to Add but has lower performance, as shown in chapter 3). To keep the evaluation process simple, the PLF model discussed in the previous chapter is not included here since it requires optimizing the matrices for many words in the datasets. I thus define two specific CSTKs: the CSTK+Add that is based on the basic additive model and the CSTK+FA that is based on the full additive model. I describe again the two composition models in the following paragraphs and extend FullAdd to the recursive case.

The Add model (Mitchell and Lapata, 2008) computes the distibutional semantics vector of a pair of words $p = uv$ as:

$$f_{Add}(u, v) = \overrightarrow{u} + \overrightarrow{v}$$

The basic additive model for word sequences $s = w_1 \dots w_k$ is recursively defined as follows:

$$f_{Add}(s) = \begin{cases} \overrightarrow{w_1} & \text{if } k = 1 \\ \overrightarrow{w_1} + f_{Add}(w_2 \dots w_k) & \text{if } k > 1 \end{cases}$$

The Full Additive model (FullAdd) (used in Guevara (2010) for adjective-noun pairs and Zanzotto et al. (2010) for three different syntactic relations) computes the compositional vector $\overrightarrow{p}$ of a pair using two linear tranformations W_1 and W_2 respectively applied to the vectors of the first and the second word. These matrices generally only depends on the syntactic relation R that links those two words. The operation follows:

$$f_{FA}(u, v, R) = W_{1R}\overrightarrow{u} + W_{2R}\overrightarrow{v}$$

The full additive model for word sequences $s = w_1 \dots w_k$, whose node has a production rule $s \rightarrow s_{c_1} \dots s_{c_m}$ is also defined recursively:

$$f_{FA}(s) = \begin{cases} \overrightarrow{w_1} & \text{if } k = 1 \\ W_{1vn}\overrightarrow{V} + W_{2vn} f_{FA}(NP) & \\ \quad \text{if } s \rightarrow V\ NP & \\ W_{2an}\overrightarrow{A} + W_{2an} f_{FA}(N) & \\ \quad \text{if } s \rightarrow A\ N & \\ \sum f_{FA}(s_{c_i}) & \text{otherwise} \end{cases}$$

where W_{1vn}, W_{wvn} are matrices used for verb and noun phrase interaction, and W_{1an}, W_{2an} are used for adjective, noun interaction.

5.3 Experimental Investigation

5.3.1 Experimental set-up

Since Tree Kernels methods are good for supervised tasks, I experimented with the Recognizing Textual Entailment datasets (RTE) (Dagan et al., 2006). RTE is the task of deciding whether a long text T entails a shorter text, typically a single sentence, called hypothesis H. It has been often seen as a classification task (see Dagan et al. (2013)). I used four datasets: RTE1, RTE2, RTE3, and RTE5, with the standard split between training and testing. The dev/test distribution for RTE1-3, and RTE5 is respectively 567/800, 800/800, 800/800, and 600/600 T-H pairs.

Distributional vectors are derived from a corpus obtained by the concatenation of ukWaC (wacky.sslmit.unibo.it), a mid-2009 dump of the English Wikipedia (en.wikipedia.org) and the British National Corpus (www.natcorp.ox.ac.uk), for a total of about 2.8 billion words. I collected a 35K-by-35K matrix by counting co-occurrence of the 30K most frequent content lemmas in the corpus (nouns, adjectives and verbs) and all the content lemmas occurring in the datasets within a 3 word window. The raw count vectors were transformed into positive Pointwise Mutual Information scores and reduced to 300 dimensions by Singular Value Decomposition. This setup was picked without tuning, as I found it effective in previous, unrelated experiments and also elsewhere in the thesis.

I built the matrices for the full additive models using the procedure described in Guevara (2010). I considered only two relations: the Adjective-Noun and Verb-Noun. The full additive model falls back to the basic additional model when syntactic relations are different from these two.

To build the final kernel to learn the classifier, I followed standard approaches (Dagan et al., 2013), that is, I exploited two models: a model with only a rewrite rule feature space (RR) and a model with the previous space along with a token-level similarity feature (RRTWS). The two models use the CSTKs and the standard TKs in the following way as kernel functions: (1) $RR(p_1, p_2) = \kappa(t_1^a, t_2^a) + \kappa(t_1^b, t_2^b)$; (2) $RRTWS(p_1, p_2) = \kappa(t_1^a, t_2^a) + \kappa(t_1^b, t_2^b) + (TWS(a_1, b_1) \cdot TWS(a_2, b_2) + 1)^2$ where TWS is a weighted token similarity (as in Corley and Mihalcea (2005)).

	RR					RRTWS				
	RTE1	**RTE2**	**RTE3**	**RTE5**	**Average**	**RTE1**	**RTE2**	**RTE3**	**RTE5**	**Average**
Add	0.541	0.496	0.507	0.520	0.516	0.560	0.538	0.643	0.578	0.579
FullAdd	0.512	0.516	0.507	0.569	0.526	0.571	0.608	0.643	0.643	0.616
TK	0.561	0.552	0.531	0.54	0.546	0.608	0.627	0.648	0.630	0.628
CSTK+Add	0.553	0.545	0.562	0.568	$0.557^{\dagger}$	0.626	0.616	0.648	0.628	$0.629^{\dagger}$
CSTK+FA	0.543	0.550	0.574	0.576	$\mathbf{0.560}^{\dagger}$	0.628	0.616	0.652	0.630	$\mathbf{0.631}^{\dagger}$

Table 5.1: Task-based analysis: Accuracy on Recognizing Textual Entailment († is different from both ADD and FullADD with a stat.sig. of $p > 0.1$.)

5.3.2 Results

Table 5.1 shows the results of the experiments. The table is organized as follows: columns 2-6 report the accuracy of the RTE systems based on rewrite rules (RR) and columns 7-11 report the accuracies of RR systems along with token similarity (RRTS). I compare five differente models: Add is the additive model applied to the words of the sentence (without considering its tree structure), the same is done for the Full Additive (FullADD), defined as in 5.2.3. The Tree Kernel (TK) as defined in Collins and Duffy (2002) are applied to the constituency-based tree representation of the tree, without the intervening collapsing step described in 5.2.2. These three models are the baseline against which I compare the CSTK models where the collapsing procedure is done via Additive (CSTK + Add) and Full Additive (CSTK + FA), as described in section 5.2.2, again, with the aforementioned restriction on the relation considered. For RR models I have that CSTK+Add and CSTK+FA both achieve higher accuracy than Add and FullAdd, with a statistical significante greater than 93.7%, as computed with the sign test. Specifically I have that CSTK+Add has an average accuracy 7.94% higher than Add and 5.89% higher than FullADD, while CSTK+FA improves on Add and FullAdd by 8.52% and 6.46%, respectively. The same trend is visible for the RRTS model, again both models are statistically better than ADD and FullAdd, in this case I have that CSTK+Add is 8.63% more accurate then ADD and 2.11% more than FullAdd, CSTK+FA is respectively 8.98% and 2.43% more accurate than Add and FullAdd. As for the TK models I have that both CSTK models achieve again an higher average accuracy: for RR models CSTK+Add and CSTK+FA are respectively 2.01% and 0.15% better than TK, while for RRTS models the number are 2.54% and 0.47%. These

results though are not statistically significant, as is the difference between the two CSTK models themselves.

5.4 Summary

In this chapter, I introduced a novel sub-class of the convolution kernels in order to exploit reliable compositional distributional semantic models along with the syntactic structure of sentences.

The Chunk-based Smoothed Tree Kernels (CSTKs) are an alternative to PLF in obtaining sentence meaning representations. The main difference between the two approaches is that CSTKs are more suitable for supervised tasks, while PLF is a method to build general vector representations for sentences. Another difference lies in the way they use syntactic information. In PLF, syntax guides composition method, deciding the composing order whereas in CSTKs, syntax information is used directly in sentence meaning comparison.

Experiments show that this novel sub-class, namely, CSTKs, are a promising solution, performing significantly better than a naive recursive application of the simple compositional distributional semantic models on a supervised task, textual entailment recognition. I experimented with CSTKS equipped with the additive and the full additive models but these kernels are definitely open to be extended to all the basic composition models.

Chapter 6

C-PHRASE

6.1 Introduction

The recent evaluation of Baroni et al. (2014) suggests that the C-BOW model introduced by Mikolov et al. (2013a) is, consistently, the best across many tasks.[1] Interestingly, C-BOW vectors are estimated with a simple compositional approach: The weights of adjacent words are jointly optimized so that their sum will predict the distribution of their contexts. This is reminiscent of how the parameters of some *compositional* distributional semantic models are estimated by optimizing the prediction of the contexts in which phrases occur in corpora (Baroni and Zamparelli, 2010; Guevara, 2010), which is generalized to most compositional models in chapter 3. However, these compositional approaches assume that word vectors have already been constructed, and contextual evidence is only used to induce optimal combination rules to derive representations of phrases and sentences.

In this chapter, I follow through on this observation to propose the new C-PHRASE model, another method for building sentence meaning representations. Similarly to C-BOW, C-PHRASE learns word representations by optimizing their joint context prediction. However, unlike in flat, window-based C-BOW, C-PHRASE groups words according

[1] I refer here not only to the results reported in Baroni et al. (2014), but also to the more extensive evaluation that Baroni and colleagues present in the companion website (`http://clic.cimec.unitn.it/composes/semantic-vectors.html`). The experiments there suggest that only the Glove vectors of Pennington et al. (2014) are competitive with C-BOW, and only when trained on a corpus several orders of magnitude larger than the one used for C-BOW.

to their syntactic structure, and it simultaneously optimizes context-predictions at different levels of the syntactic hierarchy. For example, given training sentence *"A sad dog is howling in the park"*, C-PHRASE will optimize context prediction for *dog*, *sad dog*, *a sad dog*, *a sad dog is howling*, etc., but not, for example, for *howling in*, as these two words do not form a syntactic constituent by themselves.

Because they are estimated in a compositional way, C-PHRASE word vectors, when combined through simple addition, produce sentence representations that are better than those obtained when adding other kinds of vectors, and competitive against *ad-hoc* compositional methods on various sentence meaning benchmarks. Moreover, unlike compositional methods such as PLF and CSTKs in chapters 4 and 5, which use already-built word vectors, the C-PHRASE model is able to learn both sentence vectors and its own word vectors jointly. In addition, C-PHRASE word representations outperform C-BOW on several word-level benchmarks.

6.2 The C-PHRASE model

I start with reintroducing the Skip-gram and CBOW by Mikolov et al. (2013a), as C-PHRASE builds on them. As previously discussed in chapter 2, the **Skip-gram** model learns the vector of a target word by setting its weights to predict the words surrounding it in the corpus.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{6.1}$$

Skip-gram learns each word representation separately , the **C-BOW** model takes their combination into account. More precisely, it tries to predict a context word from the combination of the previous and following words, where the combination method is vector addition. The objective function is:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}..w_{t-1}, w_{t+1}..w_{t+c}) \tag{6.2}$$

While other distributional models consider sequences of words jointly as *context* when

estimating the parameters for a single word (Agirre et al., 2009; Melamud et al., 2014), C-BOW is unique in that it estimates the weights of a sequence of words jointly, based on their shared context. In this respect, C-BOW extends the distributional hypothesis (Harris, 1954) that words with similar context distributions should have similar meanings to longer sequences. However, the word combinations of C-BOW are not natural linguistic constituents, but arbitrary n-grams (e.g., sequences of 5 words with a gap in the middle). Moreover, the model does not attempt to capture the *hierarchical* nature of syntactic phrasing, such that *big brown dog* is a meaningful phrase, but so are its children *brown dog* and *dog*.

C-PHRASE aims at capturing the same intuition that word combinations with similar context distributions will have similar meaning, but it applies this intuition to syntactically motivated, potentially nested phrases. More precisely, I estimate word vectors such that they and their summed combinations are able to predict the contexts of words, phrases and sentences. The model is formalized as follows. I start from a parsed text corpus $\mathbb{T}$, composed of constituents $\mathcal{C}[w_l, \cdots, w_r]$, where $w_l, \cdots, w_r$ are the words spanned by the constituent, located in positions l to r in the corpus. I minimize an objective function analogous to equations (6.1) and (6.2), but instead of just using individual words or bags of words to predict context, I use summed vector representations of well-formed constituents at all levels in the syntactic tree to predict the context of these constituents. There are similarities with both CBOW and Skip-gram. At the leaf nodes, C-PHRASE acts like Skip-gram, whereas at higher node in the parse tree, it behaves like CBOW model. Concretely, I try to predict the words located within a window $c_{\mathcal{C}}$ from every constituent in the parse tree.[2] In order to do so, I learn vector representations for words v_w by maximizing the sum of the log probabilities of the words in the context window of the well-formed constituents with stochastic gradient descent:

[2]Although here I only use single words as context, the latter can be extended to encompass any sensible linguistic item, e.g., frequent n-grams or, as discussed below, syntactically-mediated expressions

$$\sum_{\mathcal{C}[w_l,\cdots,w_r]\in\mathbb{T}} \sum_{1\leq j\leq c_C} \Big(\log p(w_{l-j}|\mathcal{C}[w_l,\cdots,w_r]) + \log p(w_{r+j}|\mathcal{C}[w_l,\cdots,w_r]) \Big) \quad (6.3)$$

with p theoretically defined as:

$$p(w_O|\mathcal{C}[w_l,\cdots,w_r]) = \frac{\exp\left(v'^{\top}_{w_O} \frac{\sum_{i=l}^{r} v_{w_i}}{r-l+1}\right)}{\sum_{w=1}^{W} \exp\left(v'^{\top}_{w} \frac{\sum_{i=l}^{r} v_{w_i}}{r-l+1}\right)}$$

where W is the size of the vocabulary, v' and v denote output (context) and input vectors, respectively, and I take the input vectors to represent the words. In practice, since the normalization constant for the above probability is expensive to compute, I follow Mikolov et al. (2013b) and use negative sampling.

I let the context window size $c_{\mathcal{C}}$ vary as a function of the height of the constituent in the syntactic tree. The height $h(\mathcal{C})$ of a constituent is given by the maximum number of intermediate nodes separating it from any of the words it dominates (such that $h = 0$ for words, $h = 1$ for two-word phrases, etc.). Then, for a constituent of height $h(\mathcal{C})$, C-PHRASE considers $c_{\mathcal{C}} = c_1 + h(\mathcal{C})c_2$ context words to its left and right (the non-negative integers c_1 and c_2 are hyperparameters of the model; with $c_2 = 0$, context becomes constant across heights). The intuition for enlarging the window proportionally to height is that, for shorter phrases, narrower contexts are likely to be most informative (e.g., a modifying adjective for a noun), whereas for longer phrases and sentences it might be better to focus on broader "topical" information spread across larger windows (paragraphs containing sentences about weather might also contain the words *rain* and *sun*, but without any tendency for these words to be perfectly adjacent to the target sentences).

Figure 6-1 illustrates the prediction objective for a two-word phrase and its children. Since all constituents (except the topmost) form parts of larger constituents, their repre-

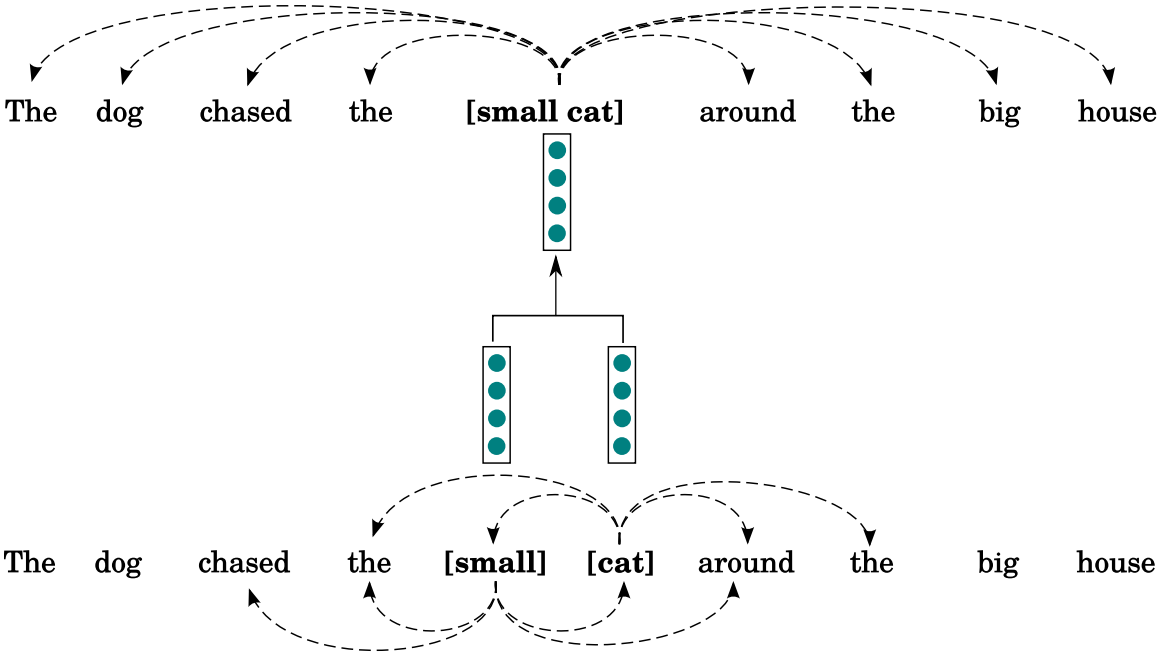


Figure 6-1: C-PHRASE context prediction objective for the phrase *small cat* and its children. The phrase vector is obtained by summing the word vectors. The predicted window is wider for the higher constituent (the phrase).

sentations will be learned both from the objective of predicting their own contexts, and from error propagation from the same objective applied higher in the tree. As a side effect, words, being lower in the syntactic tree, will have their vectors updated more often, and thus might have a greater impact on the learned parameters. This is another reason for varying window size with height, so that the latter effect will be counter-balanced by higher constituents having larger context windows to predict.

For lexical tasks, I directly use the vectors induced by C-PHRASE as word representations. For sentential tasks, I simply add the vectors of the words in a sentence to obtain its representation, exploiting the fact that C-PHRASE was trained to predict phrase contexts from the additive combination of their elements.

Joint optimization of word and phrase vectors The C-PHRASE hierarchical learning objective can capture, in parallel, generalizations about the contexts of words and phrases at different levels of complexity. This results, as we will see, in better word vectors, presumably because C-PHRASE is trained to predict how the contexts of a word change based

on its phrasal collocates (*cup* will have very different contexts in *world cup* vs. *coffee cup*). At the same time, because the vectors are optimized based on their occurrence in phrases of different syntactic complexity, they produce good sentence representations when they are combined. To the best of my knowledge, C-PHRASE is the first model that is jointly optimized for lexical and compositional tasks. C-BOW uses shallow composition information to learn word vectors. Conversely, some compositional models –e.g., Kalchbrenner et al. (2014), Socher et al. (2013)– induce word representations, that are only optimized for a compositional task and are not tested at the lexical level. Somewhat relatedly to C-PHRASE, Hill et al. (2014) evaluated representations learned in a sentence translation task on word-level benchmarks. Some *a priori* justification for treating word and sentence learning as joint problems comes from human language acquisition, as it is obvious that children learn word and phrase meanings in parallel and interactively, not sequentially (Tomasello, 2003).

Knowledge-leanness and simplicity For training, C-PHRASE requires a large, syntactically parsed corpus (more precisely, it only requires the constituent structure assigned by the parser, as it is blind to syntactic labels). Both large unannotated corpora and efficient pre-trained parsers are available for many languages, making the C-PHRASE knowledge demands feasible for practical purposes. There is no need to parse the sentences we want to build representations for at test time, since the component word vectors are simply added. The only parameters of the model are the word vectors; specifically, no extra parameters are needed for composition (composition models such as the one presented in Socher et al. (2012) require an extra parameter matrix for each word in the vocabulary, and even leaner models such as the one of Guevara (2010) must estimate a parameter matrix for each composition rule in the grammar). This makes C-PHRASE as simple as additive and multiplicative composition (Mitchell and Lapata, 2010),[3] but C-PHRASE is both more effective in compositional tasks (see evaluation below), and it has the further advantage that it learns its own word vectors, thus reducing the number of arbitrary choices to be made in modeling.

[3]I do not report results for component-wise multiplicative in the evaluation because it performed much worse than addition in all the tasks.

Supervision Unlike many recent composition models (Kalchbrenner and Blunsom, 2013; Kalchbrenner et al., 2014; Socher et al., 2012, 2013, among others), the context-prediction objective of C-PHRASE does not require annotated data, and it is meant to provide general-purpose representations that can serve in different tasks. C-PHRASE vectors can also be used as initialization parameters for fully supervised, task-specific systems. Alternatively, the current unsupervised objective could be combined with task-specific supervised objectives to fine-tune C-PHRASE to specific purposes.

Sensitivity to syntactic structure During training, C-PHRASE is sensitive to syntactic structure. To cite an extreme example, *boy flowers* will be joined in a context-predicting phrase in *"these are considered [boy flowers]"*, but not in *"he gave [the boy] [flowers]"*. A more common case is that of determiners, that will only occur in phrases that also contain the following word, but not necessarily the preceding one. Sentence composition at test time, on the other hand, is additive, and thus syntax-insensitive. Still, the vectors being combined will reflect syntactic generalizations learned in training. Even if C-PHRASE produces the same representation for *red*+*car* and *car*+*red*, this representation combines a *red* vector that, during training, has often occurred in the modifier position of adjective-noun phrases, whereas *car* will have often occurred in the corresponding head position. So, presumably, the *red*+*car*=*car*+*red* vector will encode the adjective-noun asymmetry induced in learning. While the model won't be able to distinguish the rare cases in which *car red* is genuinely used as a phrase, in realistic scenarios this won't be a problem, because only *red car* will be encountered. In this respect, the successes and failures of C-PHRASE can tell us to what extent word order information is truly distinctive in practice, and to what extent it can instead be reconstructed from the typical role that words play in sentences.

Comparison with traditional syntax-sensitive word representations Syntax has often been exploited in distributional semantics for a richer characterization of *context*. By relying on a syntactic parse of the input corpus, a distributional model can take more informative contexts such as *subject-of-eat* vs. *object-of-eat* into account (Baroni and Lenci, 2010; Curran and Moens, 2002; Grefenstette, 1994; Erk and Padó, 2008; Levy and Gold-

berg, 2014a; Padó and Lapata, 2007; Rothenhäusler and Schütze, 2009). In this approach, syntactic information serves to select and/or enrich the *contexts* that are used to build representations of target units. On the other hand, I use syntax to determine the *target units* that I build representations for (in the sense that I jointly learn representations of their constituents). The focus is thus on unrelated aspects of model induction, and I could indeed use syntax-mediated contexts together with this phrasing strategy. Currently, given *eat (red apples)*, I treat *eat* as window-based context of *red apples*, but I could also take the context to be *object-of-eat*.

6.3 Evaluation

Here I evaluate the quality of both word vectors and sentences vectors produced by C-PHRASE using a variety of lexical and sentential tasks.

6.3.1 Data sets

Semantic relatedness of words In this classic lexical task, the models are required to quantify the degree of semantic similarity or relatedness of pairs of words in terms of cosines between the corresponding vectors. These scores are then compared to human gold standards. Performance is assessed by computing the correlation between system and human scores (Spearman correlation in all tasks except rg, where it is customary to report Pearson). I used, first of all, the MEN (**men**) data set of Bruni et al. (2014), that is split into 1K pairs for training/development, and 1K pairs for testing. I used the training set to tune the hyperparameters of C-Phrase model, and report performance on the test set. The C-BOW model of Baroni et al. (2014) achieved state-of-the art performance on MEN test. I also evaluate on the widely used WordSim353 set introduced by Finkelstein et al. (2002), which consists of 353 word pairs. The WordSim353 data were split by Agirre et al. (2009) into similarity (**wss**) and relatedness (**wsr**) subsets, focusing on strictly taxonomic (*television/radio*) vs. broader topical cases (*Maradona/football*), respectively. State-of-the-art performance on both sets is reported by Baroni et al. (2014), with the C-BOW model. I further consider the classic data set of Rubenstein and Goodenough (1965) (**rg**), consisting

of 65 noun pairs. I report the state-of-the-art from Hassan and Mihalcea (2011), which exploited Wikipedia's linking structure.

Concept categorization Systems are asked to group a set of nominal concepts into broader categories (e.g. *arthritis* and *anthrax* into *illness*; *banana* and *grape* into *fruit*). As in previous work, I treat this as an unsupervised clustering task. I feed the similarity matrix produced by a model for all concepts in a test set to the CLUTO toolkit (Karypis, 2003), that clusters them into n groups, where n is the number of categories. I use standard CLUTO parameters from the literature, and quantify performance by cluster purity with respect to the gold categories. The Almuhareb-Poesio benchmark (Almuhareb, 2006) (**ap**) consists of 402 concepts belonging to 21 categories. A distributional model based on carefully chosen syntactic relations achieved top ap performance (Rothenhäusler and Schütze, 2009). The ESSLLI 2008 data set (Baroni et al., 2008) (**esslli**) consists of 6 categories and 42 concepts. State of the art was achieved by Katrenko and Adriaans (2008) by using full-Web queries and manually crafted patterns.

Semantic analogy The last lexical task I pick is analogy (**an**), introduced in Mikolov et al. (2013c). I focus on their *semantic* challenge, containing about 9K questions. In each question, the system is given a pair exemplifying a relation (*man*/*king*) and a test word (*woman*); it is then asked to find the word (*queen*) that instantiates the same relation with the test word as that of the example pair. Mikolov et al. (2013c) subtract the vector of the first word in a pair from the second, add the vector of the test word and look for the nearest neighbor of the resulting vector (e.g., find the word whose vector is closest to *king* - *man* + *woman*). I follow the method introduced by Levy and Goldberg (2014b), which returns the word x maximizing $\frac{\cos(king,x)\times\cos(woman,x)}{\cos(man,x)}$. This method yields better results for all models. Performance is measured by accuracy in retrieving the correct answer (in the search space of 180K words). The current state of the art on the semantic part and on the whole data set was reached by Pennington et al. (2014), who trained their word representations on a huge corpus consisting of 42B words.

Sentential semantic relatedness As the main purpose of C-PHRASE is to obtain good sentence meaning representations, I evaluate it also on several sentential tasks. As shown in chapter 4, composed sentence representations can be evaluated against benchmarks where humans provided relatedness/similarity scores for sentence pairs (sentences with high scores, such as *"A person in a black jacket is doing tricks on a motorbike"/"A man in a black jacket is doing tricks on a motorbike"* from the SICK data-set, tend to be near-paraphrases). Following previous work on these data sets, Pearson correlation is the figure of merit, and I report it between human scores and sentence vector cosine similarities computed by the models. SICK (Marelli et al., 2014) (**sick-r**) was created specifically for the purpose of evaluating compositional models, focusing on linguistic phenomena such as lexical variation and word order. Here I report performance of the systems on the test part of the data set, which contains 5K sentence pairs. The top performance (from the SICK SemEval shared task) was reached by Zhao et al. (2014) using a heterogeneous set of features that include WordNet and extra training corpora. Agirre et al. (2012) and Agirre et al. (2013) created two collections of sentential similarities consisting of subsets coming from different sources. From these, I pick the Microsoft Research video description dataset (**msrvid**), where near paraphrases are descriptions of the same short video, and the OnWN 2012 (**onwn1**) and OnWN 2013 (**onwn2**) data sets (each of these sets contains 750 pairs). The latter are quite different from other sentence relatedness benchmarks, since they compare definitions for the same or different words taken from WordNet and OntoNotes: these glosses often are syntactic fragments (*"cause something to pass or lead somewhere"*), rather than full sentences. I report top performance on these tasks from the respective shared challenges, as summarized by Agirre et al. (2012) and Agirre et al. (2013). Again, the top systems use feature-rich, supervised methods relying on distributional similarity as well as other sources, such as WordNet and named entity recognizers. Since here, I would like to focus on large dataset with arbitrary sentence structures, simple datasets in chapter 4 like **anvan** and **tfds** are excluded.

Sentential entailment Similar to chapter 5, I also evaluate the models on the textual entailment task since detecting the presence of entailment between sentences or longer pas-

sages is one of the most useful features that the computational analysis of text could provide (Dagan et al., 2009). I test the model on the SICK entailment task (**sick-e**) (Marelli et al., 2014). All SICK sentence pairs are labeled as ENTAILING (*"Two teams are competing in a football match"/"Two groups of people are playing football"*), CONTRADICTING (*"The brown horse is near a red barrel at the rodeo"/"The brown horse is far from a red barrel at the rodeo"*) or NEUTRAL (*"A man in a black jacket is doing tricks on a motorbike"/"A person is riding the bicycle on one wheel"*). For each model, I train a simple SVM classifier based on 2 features: cosine similarity between the two sentence vectors, as given by the models, and whether the sentence pair contains a negation word (the latter has been shown to be a very informative feature for SICK entailment). The current state-of-the-art is reached by Lai and Hockenmaier (2014), using a much richer set of features, that include WordNet, the denotation graph of Young et al. (2014) and extra training data from other resources.

Sentiment analysis Finally, as sentiment analysis has emerged as a popular area of application for compositional models, I test the methods on the Stanford Sentiment Treebank (Socher et al., 2013) (**sst**), consisting of 11,855 sentences from movie reviews, using the coarse annotation into 2 sentiment degrees (*negative/positive*). I follow the official split into train (8,544), development (1,101) and test (2,210) parts. I train an SVM classifier on the training set, using the sentence vectors composed by a model as features, and report accuracy on the test set. State of the art is obtained by Le and Mikolov (2014) with the Paragraph Vector approach I describe below.

6.3.2 Model implementation

The source corpus I use to build the lexical vectors is created by concatenating three sources: ukWaC,[4] a mid-2009 dump of the English Wikipedia[5] and the British National Corpus[6] (about 2.8B words in total). I build vectors for the 180K words occurring at least

[4]http://wacky.sslmit.unibo.it
[5]http://en.wikipedia.org
[6]http://www.natcorp.ox.ac.uk

100 times in the corpus. Since the training procedure requires parsed trees, I parse the corpus using the Stanford parser (Klein and Manning, 2003).

C-PHRASE has two hyperparameters (see Section 6.2 above), namely basic window size (c_1) and height-dependent window enlargement factor (c_2). Moreover, following Mikolov et al. (2013b), during training I sub-sample less informative, very frequent words: this option is controlled by a parameter t, resulting in aggressive subsampling of words with relative frequency above it. I tune on MEN-train, obtaining $c_1 = 5$, $c_2 = 2$ and $t = 10^{-5}$. As already mentioned, sentence vectors are built by summing the vectors of the words in them.

In lexical tasks, I compare the C-Phrase model to the best C-BOW model from Baroni et al. (2014),[7] and to a Skip-gram model built using the same hyperparameters as C-PHRASE (that also led to the best MEN-train results for Skip-gram).

In sentential tasks, I compare this model against adding the best C-BOW vectors pre-trained by Baroni and colleagues,[8] and adding the Skip-gram vectors. I compare the additive approaches to two sophisticated composition models. The first is the Practical Lexical Function (**PLF**) model in chapter 4. Here, I report PLF results on msrvid and onwn2 from chapter 4, noting that there I also used two simple but precious cues (word overlap and sentence length) I do not adopt here. I used the pre-trained vectors and matrices from the previous chapter also for the SICK challenges, while the number of new matrices to estimate made it too time-consuming to implement this model in the onwn1 and sst tasks.

Finally, I test the Paragraph Vector (**PV**) approach recently proposed by Le and Mikolov (2014). Under PV, sentence representations are learned by predicting the words that occur in them. This unsupervised method has been shown by the authors to outperform much more sophisticated, supervised neural-network-based composition models on the sst task. I use my own implementation for this approach. Unlike in the original experiments, I found the PV-DBOW variant of PV to consistently outperform PV-DM, and so I report results obtained with the former.

Note that PV-DBOW aims mainly at providing representations for sentences, not words.

[7] For fairness, I report their results when all tasks were evaluated with the same set of parameters, tuned on rg: this is row 8 of their Table 2.

[8] `http://clic.cimec.unitn.it/composes/semantic-vectors.html`

	men	wss	wsr	rg	ap	esslli	an
Skip-gram	78	77	66	80	65	82	63
C-BOW	**80**	78	68	**83**	**71**	77	68
C-PHRASE	79	**79**	**70**	**83**	65	**84**	**69**
SOA	80	80	70	86	79	91	82

Table 6.1: Lexical task performance. See Section 6.3.1 for figures of merit (all in percentage form) and state-of-the-art references. C-BOW results (tuned on rg) are taken from Baroni et al. 2014b.

	sick-r	sick-e	msrvid	onwn1	onwn2	sst
Skip-gram	70	72	74	66	62	78
C-BOW	70	74	74	69	63	**79**
C-PHRASE	**72**	**75**	**79**	**70**	65	**79**
PLF	57	72	**79**	NA	**67**	NA
PV	67	**75**	77	66	66	77
SOA	83	85	88	71	75	88

Table 6.2: Sentential task performance. See Section 6.3.1 for figures of merit (all in percentage form) and state-of-the-art references. The PLF results on msrvid and onwn2 are taken from Paperno et al. 2014.

When there is no need to induce vectors for sentences in the training corpus, i.e., only train to learn single words' representations and the softmax weights, PV-DBOW essentially reduces to Skip-gram. Therefore, I produce the PV-DBOW vectors for the sentences in the evaluation data sets using the softmax weights learned by Skip-gram. However, it is not clear that, if we were to train PV-DBOW jointly for words and sentences, we would get word vectors as good as those that Skip-gram induces.

Note that here I do not compare C-PHRASE to the CSTKs model since CTSKs are more suitable for supervised tasks than generic tasks like sentence relatedness.

6.4 Results

The results on the lexical tasks reported in Table 6.1 prove that C-PHRASE is providing excellent word representations, (nearly) as good or better than the C-BOW vectors of Baroni and colleagues in all cases, except for ap. Whenever C-PHRASE is not close to the state of the art results, the latter relied on richer knowledge sources and/or much larger corpora (ap, esslli, an).

Turning to the sentential tasks (Table 6.2), I first remark that using high-quality word vectors (such as C-BOW) and summing them leads to good results in all tasks, competitive with those obtained with more sophisticated composition models. This confirms the observation made by Blacoe and Lapata (2012) that simple-minded composition models are not necessarily worse than advanced approaches. Still, C-PHRASE is consistently better than C-BOW in all tasks, except sst, where the two models reach the same performance level.

C-PHRASE is outperforming PV on all tasks except sick-e, where the two models have the same performance, and onwn2, where PV is slightly better. C-PHRASE is outperforming PLF by a large margin on the SICK sets, whereas the two models are equal on msrvid, and PLF better on onwn2. Recall, however, that on the latter two benchmarks PLF used extra word overlap and sentence length features, but its word vectors have potentially lower quaility so the comparison is not entirely fair.

The fact that state-of-the-art performance is well above my models is not surprising, since the SOA systems are invariably based on a wealth of knowledge sources, and highly optimized for a task. To put some of the results in a broader perspective, C-PHRASE's sick-r performance is 1% better than the median result of systems that participated in the SICK SemEval challenge, and comparable to that of Beltagy et al. (2014), who entered the competition with a system combining distributional semantics with a supervised probabilistic soft logic system. For sick-e (the entailment task), C-PHRASE's performance is less than one point below the median of the SemEval systems, and slightly above that of the Stanford submission, that used a recursive neural network with a tensor layer.

Finally, the performance of all the models here, including PV, on sst is remarkably lower than the state-of-the-art performance of PV as reported by Le and Mikolov (2014). I believe that the crucial difference is that these authors estimated PV vectors *specifically on the sentiment treebank training data*, thus building *ad-hoc* vectors encoding the semantics of movie reviews. I leave it to further research to ascertain whether I could better fine-tune the models to sst by including the sentiment treebank training phrases in the source corpus.

Comparing vector lengths of C-BOW and C-PHRASE We can gather some insight into how the C-PHRASE objective might adjust word representations for composition with

respect to C-BOW by looking at how the length of word vectors changes across the two models.[9] While this is a very coarse measure, if a word vector is much longer/shorter (relative to the length of other word vectors of the same model) for C-PHRASE vs. C-BOW, it means that, when sentences are composed by addition, the effect of the word on the resulting sentence representation will be stronger/weaker.

The relative-length-difference test returns the following words as the ones that are most severely de-emphasized by C-PHRASE compared to C-BOW: *be, that, an, not, they, he, who, when, well, have*. Clearly, C-PHRASE is weighting down grammatical terms that tend to be context-agnostic, and will be accompanied, in phrases, by more context-informative content words. Indeed, the list of terms that are instead emphasized by C-PHRASE include such content-rich, monosemous words as *gnawing, smackdown, demographics*. This is confirmed by a POS-level analysis that indicates that the categories that are, on average, most de-emphasized by C-PHRASE are: determiners, modals, pronouns, prepositions and (more surprisingly) proper nouns. The ones that are, in relative terms, more emphasized are: *-ing* verb forms, plural and singular nouns, adjectives and their superlatives. While this reliance on content words to the detriment of grammatical terms is not always good for sentential tasks (*"not always good"* means something very different from *"always good"*!), the convincing comparative performance of C-PHRASE in such tasks suggests that the semantic effect of grammatical terms is in any case beyond the scope of current corpus-based models, and often not crucial to attain competitive results on typical benchmarks (think, e.g., of how little modals, one of the categories that C-PHRASE downplays the most, will matter when detecting paraphrases that are based on picture descriptions).

I also applied the length difference test to words in specific categories, finding similar patterns. For example, looking at *-ly* adverbs only, those that are de-emphasized the most by C-PHRASE are *recently, eventually, originally, notably* and *currently* – all adverbs denoting temporal factors or speaker attitude. On the other hand, the ones that C-PHRASE lengthens the most, relative to C-BOW, are *clinically, divinely, ecologically, noisily* and *theatrically*: all adverbs with more specific, content-word-like meanings, that are better

[9] I performed the same analysis for C-PHRASE and Skip-gram, finding similar general trends to the ones I report for C-PHRASE and C-BOW.

captured by distributional methods, and are likely to have a bigger impact on tasks such as paraphrasing or entailment.

C-BOW			C-PHRASE		
neural	*network*	*neural network*	*neural*	*network*	*neural network*
neuronal	networks	network	neuronal	networks	network
neurons	superjanet4	neural	cortical	internetwork	neural
hopfield	backhaul	networks	connectionist	wans	perceptron
cortical	fiber-optic	hopfield	neurophysiological	network.	networks
connectionist	point-to-multipoint	packet-switched	sensorimotor	multicasting	hebbian
feed-forward	nsfnet	small-world	sensorimotor	nsfnet	neurons
feedforward	multi-service	local-area	neocortex	networking	neocortex
neuron	circuit-switched	superjanet4	electrophysiological	tymnet	connectionist
backpropagation	wide-area	neuronal	neurobiological	x.25	neuronal

Table 6.3: Nearest neighbours of *neural*, *network* and *neural network* both for C-BOW and C-PHRASE

Effects of joint optimization at word and phrase levels As I have argued before, C-PHRASE is able to obtain good word representations presumably because it learns to predict how the context of a word changes in the presence of different collocates. To gain further insight into this claim, I looked at the nearest neighbours of some example terms, like *neural*, *network* and *neural network* (the latter, composed by addition) both in C-PHRASE and C-BOW. The results for this particular example can be appreciated in Table 6.3.

Interestingly, while for C-BOW we observe some confusion between the meaning of the individual words and the phrase, C-PHRASE seems to provide more orthogonal representations for the lexical items. For example, *neural* in C-BOW contains neighbours that fit well with *neural network*, like *hopfield*, *connectionist* and *feed-forward*. Conversely, *neural network* has neighbours that correspond to *network* like *local-area* and *packet-switched*. In contrast, C-PHRASE neighbours for *neural* are mostly related to the *brain* sense of the word, e.g., *cortical*, *neurophysiological*, etc. (with the only exception of *connectionist*). The first neighbour of *neural network*, excluding its own component words, quite sensibly, is *perceptron*.

6.5 Summary

In this chapter, I introduced C-PHRASE, a distributional semantic model that is trained on the task of predicting the contexts surrounding phrases at all levels of a hierarchical sen-

tence parse, from single words to full sentences. Consequently, word vectors are induced by taking into account not only their contexts, but also how co-occurrence with other words within a syntactic constituent is affecting these contexts.

Not only do C-PHRASE vectors outperform state-of-the-art C-BOW vectors in a wide range of lexical tasks, but also, because of the way they are induced, when C-PHRASE vectors are summed, they produce sentence representations that are as good or better than those obtained with sophisticated composition methods. Therefore, C-PHRASE can be a good model for sentence meaning, a superior alternative to the PLF model in term of simplicity.

C-PHRASE is a very parsimonious approach: The only major resource required, compared to a completely knowledge-free, unsupervised method, is an automated parse of the training corpus (but no syntactic labels are required, nor parsing at test time). C-PHRASE has only 3 hyperparameters and no composition-specific parameter to tune and store.

Having established a strong empirical baseline with this parsimonious approach, in future research I want to investigate the impact of possible extensions on both lexical and sentential tasks. When combining the vectors, either for induction or composition, I will try replacing plain addition with other operations, starting with something as simple as learning scalar weights for different words in a phrase (Mitchell and Lapata, 2010). I also intend to explore more systematic ways to incorporate supervised signals into learning, to fine-tune C-PHRASE vectors to specific tasks.

The C-PHRASE vectors described in this chapter are made publicly available at: `http://clic.cimec.unitn.it/composes/`.

Chapter 7

Conclusion

To conclude the thesis, I will begin by summarizing the work discussed in the previous chapters.

In chapter 2, I provided a literature overview of distributional semantics, surveying the two main approaches in producing the meaning representations for words and recently proposed models to combine them for simple phrasal representations. These models serve as a good foundation for further extensions in order to obtain meaning representations for more complex phrases and sentences.

Chapter 3 introduced a general approach for estimating the parameters for all the composition models discussed in chapter 2. This provided a fair method to compare these composition models, enabling us to see which of them are more promising for further development. The experiments in this chapter show that the Lexical Function (LF) model is the most promising model, achieving the most stable performance across datasets. The additive model, however, is also a good candidate due to its simplicity despite its shortcomings, such as word-order insensitivity.

Chapters 4, 5 and 6 describe some of my attempts in representing sentence meaning. The first model, the PLF model, is discussed in chapter 4. This model is based on the LF model, a linguistically motivated model with encouraging performance in various phrasal tasks as shown in chapter 3. The PLF model generalizes the LF model for different types of constructions and provides a way to combine them in order to obtain a single vector representing the sentence meaning. It uses multiple matrices instead of high order tensors,

which preserves the attractive features of LF while avoiding its downfalls such as data sparseness and time consuming computation. The vectors obtained by PLF are general and can be used in many different tasks requiring sentence comparisons.

Chapter 5 presented the Chunk-based Smoothed Tree Kernels (CSTKs) model, an alternative method for obtaining sentence vectors. This model directly uses syntactic information in comparing sentential meanings instead of using it as a guideline for composing sentence vectors. This model achieves good performance in recognizing textual entailment, a sentential supervised task.

The last model, C-PHRASE, is presented in chapter 6. It is a neural network-based model, inspired by the C-BOW model of Mikolov et al. (2013c). It uses the additive model to compute the phrase vectors and optimize both the word vectors and the phrase vectors jointly by using the context-predicting objective functions. This model not only obtains good word vectors outperforming the state-of-the-art C-BOW model but also produces high-quality sentence vectors achieving the best overall result compared to competing methods such as PLF in a set of sentential tasks.

Overall, the methods discussed in this thesis show promising results in representing sentence meaning. I also suggested various ways to extend these models, for example, using a different composition model in C-PHRASE. There are other methods of obtaining sentential representations developed by other researchers in parallel with the work presented in this thesis. Many of them using a supervised framework and focus on specific tasks such as sentiment analysis or question answering. Here I focus more on unsupervised methods that are able to obtain general sentence vectors which can be used across different tasks since I believe a good method for meaning representation should not be restricted to a specific type of task. An alternative to these two approaches and possibly a more promising one is the multi-task framework, where the sentence representations are obtained using a variety of supervised tasks because, after all, human acquires language skills through solving various interweaving problems at the same time. All in all, I believe meaning representation is an interesting area of research and I hope that this thesis will make a helpful contribution to this exciting field.

Appendix A

Practical tools

A.1 Introduction

Distributional methods, discussed in chapter 2, are appealing models of semantics. However, modeling words in isolation has limited applications and ideally we want to model semantics *beyond word level* by representing the meaning of phrases or sentences. These combinations are infinite and compositional methods are called for to derive the meaning of a larger construction from the meaning of its parts. This leads to the birth of a large number of compositional models, covered in chapter 2 and 3. This chapter introduces the DISSECT toolkit, which I developed during the course of this thesis, due to the needs of comparing and evaluating these different compositional models.

The DISSECT toolkit (`http://clic.cimec.unitn.it/composes/toolkit`) is, to the best of my knowledge, the first to provide an easy-to-use implementation of many compositional methods proposed in the literature. As such, I hope that it will foster further work on compositional distributional semantics, as well as making the relevant techniques easily available to those interested in their many potential applications, e.g., to context-based polysemy resolution, recognizing textual entailment or paraphrase detection. Moreover, the DISSECT tools to construct distributional semantic spaces from raw co-occurrence counts, to measure similarity and to evaluate these spaces might also be of use to researchers who are not interested in the compositional framework. DISSECT is freely available under the GNU General Public License.

A.2 Building and composing distributional semantic representations

The pipeline from corpora to compositional models of meaning can be roughly summarized as consisting of three stages:[1]

1. Extraction of co-occurrence counts from corpora In this stage, an input corpus is used to extract counts of target elements co-occurring with some contextual features. The target elements can vary from words (for lexical similarity), to pairs of words (e.g., for relation categorization), to paths in syntactic trees (for unsupervised paraphrasing). Context features can also vary from shallow window-based collocates to syntactic dependencies.

2. Transformation of the raw counts This stage may involve the application of weighting schemes such as Pointwise Mutual Information, feature selection, dimensionality reduction methods such as Singular Value Decomposition, etc. The goal is to eliminate the biases that typically affect raw counts and to produce vectors which better approximate similarity in meaning.

3. Application of composition functions Once meaningful representations have been constructed for the atomic target elements of interest (typically, words), various methods, such as vector addition or multiplication, can be used for combining them to derive context-sensitive representations or for constructing representations for larger phrases or even entire sentences.

DISSECT can be used for the second and third stages of this pipeline, as well as to measure similarity among the resulting word or phrase vectors. The first step depends heavily on the language, task and corpus-annotation. I do not attempt to implement all the corpus pre-processing and co-occurrence extraction routines that it would require to be of gen-

[1] See Turney and Pantel (2010) for a technical overview of distributional methods for semantics.

eral use, and expect instead as input a matrix of raw target-context co-occurrence counts.[2] DISSECT provides various methods to re-weight the counts with association measures, dimensionality reduction methods as well as the discussed composition functions proposed by Mitchell and Lapata (2010) (*Additive*, *Multiplicative* and *Dilation*), Baroni and Zamparelli (2010)/Coecke et al. (2010) (*LF*) and Guevara (2010)/Zanzotto et al. (2010) (*FullAdd*). In DISSECT I define and implement these in a unified framework and in a computationally efficient manner described in chapter 3. The focus of DISSECT is to provide an intuitive interface for researchers and to allow easy extension by adding other composition methods.

A.3 DISSECT overview

DISSECT is written in Python. I provide many standard functionalities through a set of powerful command-line tools, however users with basic Python familiarity are encouraged to use the Python interface that DISSECT provides. This section focuses on this interface (see the online documentation on how to perform the same operations with the command-line tools), that consists of the following top-level packages:

```
#DISSECT packages
composes.matrix
composes.semantic_space
composes.transformation
composes.similarity
composes.composition
composes.utils
```

Semantic spaces and transformations The concept of a semantic space (implemented as `composes.semantic_space`) is at the core of the DISSECT toolkit. A semantic

[2]These counts can be read from a text file containing two strings (the target and context items) and a number (the corresponding count) on each line (e.g., `maggot food 15`) or from a matrix in format `word freq1 freq2 ...`

```
#create a semantic space from counts in
#dense format("dm"): word freq1 freq2 ..
ss = Space.build(data="counts.txt",
                 format="dm")

#apply transformations
ss = ss.apply(PpmiWeighting())
ss = ss.apply(Svd(300))

#retrieve the vector of a target element
print ss.get_row("car")
```

Figure A-1: Creating a semantic space.

space consists of co-occurrence values, stored as a matrix, together with strings associated to the rows of this matrix (by design, the target linguistic elements) and a (potentially empty) list of strings associated to the columns (the context features). A number of transformations (`composes.transformation`) can be applied to semantic spaces. I implement weighting schemes such as positive Pointwise Mutual Information (*ppmi*) and Local Mutual Information, feature selection methods, dimensionality reduction (Singular Value Decomposition (*SVD*) and Nonnegative Matrix Factorization (*NMF*)), and new methods can be easily added.[3] Going from raw counts to a transformed space is accomplished in just a few lines of code (Figure A-1).

Furthermore DISSECT allows the possibility of adding new data to a semantic space in an online manner (using the `semantic_space.peripheral_space` functionality). This can be used as a way to efficiently expand a co-occurrence matrix with new rows, without re-applying the transformations to the entire space. In some other cases, the user may want to represent phrases that are specialization of words already existing in the space (e.g., *slimy maggot* and *maggot*), without distorting the computation of association measures by counting the same context twice. In this case, adding *slimy maggot* as a "peripheral" row to a semantic space that already contains *maggot* implements the desired behaviour.

[3]The complete list of transformations currently supported can be found at `http://clic.cimec.unitn.it/composes/toolkit/spacetrans.html#spacetrans`.

```
#load a previously saved space
ss = io_utils.load("ss.pkl")

#compute cosine similarity
print ss.get_sim("car", "book",
                 CosSimilarity())

#the two nearest neighbours of "car"
print ss.get_neighbours("car", 2,
                        CosSimilarity())
```

Figure A-2: Similarity queries in a semantic space.

Similarity queries Semantic spaces are used for the computation of similarity scores. DISSECT provides a series of similarity measures such as cosine, inverse Euclidean distance and Lin similarity, implemented in the `composes.similarity` package. Similarity of two elements can be computed within one semantic space or across two spaces that have the same dimensionality. Figure A-2 exemplifies (word) similarity computations with DISSECT.

Composition functions Composition functions (`composes.composition`) in DISSECT take as arguments a list of element pairs to be composed, and one or two spaces where the elements to be composed are represented. They return a semantic space containing the distributional representations of the composed items, which can be further transformed, used for similarity queries, or used as inputs to another round of composition, thus scaling up beyond binary composition. Figure A-3 shows a Multiplicative composition example. See Table A.1 for the currently available composition models, their definitions and parameters.

Parameter estimation All composition models except (simple) Multiplicative have parameters to be estimated. For simple models with few parameters, such as as Additive, the parameters can be passed by hand. However, DISSECT supports automated parameter estimation from training examples. As discussed in chapter 3, I extend to all composition methods the idea originally proposed by Baroni and Zamparelli (2010) for LF and Guevara

```
#instantiate a multiplicative model
mult_model = Multiplicative()

#use the model to compose words from input space input_space
comp_space = mult_model.compose([("red", "book", "my_red_book"),
                                 ("red", "car", "my_red_car")],
                                 input_space)

#compute similarity of: 1) two composed phrases and 2) a composed
   phrase and a word
print comp_space.get_sim("my_red_book", "my_red_car",
   CosSimilarity())
print comp_space.get_sim("my_red_book", "book", CosSimilarity(),
   input_space)
```

Figure A-3: Creating and using Multiplicative phrase vectors.

(2010) for FullAdd, namely to use corpus-extracted example vectors of both the input (typically, words) and output elements (typically, phrases) in order to optimize the composition operation parameters. The problem can be generally stated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||P - fcomp_{\boldsymbol{\theta}}(U, V)||_F$$

where U, V and P are matrices containing input and output vectors respectively. For example U may contain adjective vectors such as *red*, *blue*, V noun vectors such as *car*, *sky* and P corpus-extracted vectors for the corresponding phrases *red car*, *blue sky*. $fcomp_{\boldsymbol{\theta}}$ is a composition function and $\boldsymbol{\theta}$ stands for a list of parameters that this composition function is associated with. I implement standard least-squares estimation methods as well as Ridge regression with the option for generalized cross-validation, but other methods such as partial least-squares regression can be easily added. Figure A-4 exemplifies the FullAdd model.

Composition output examples DISSECT provides functions to evaluate (compositional) distributional semantic spaces against benchmarks in the `composes.utils` package. However, as a more qualitatively interesting example of what can be done with DISSECT, Table A.2 shows the nearest neighbours of *false belief* obtained through composition with

```
#training data for learning an adjective-noun phrase model
train_data = [("red","book","red_book"),
   ("blue","car","blue_car")]

#train a fulladd model
fa_model = FullAdditive()
fa_model.train(train_data, input_space, phrase_space)

#use the model to compose a phrase from new words and retrieve
   its nearest neighb.
comp_space = fa_model.compose([("yellow", "table",
   "my_yellow_table")], input_space)
print comp_space.get_neighbours("my_yellow_table", 10,
   CosSimilarity())
```

Figure A-4: Estimating a FullAdd model and using it to create phrase vectors.

Model	Composition function	Parameters
Add.	$w_1\overrightarrow{u} + w_2\overrightarrow{v}$	$w_1(=1), w_2(=1)$
Mult.	$\overrightarrow{u} \odot \overrightarrow{v}$	-
Dilation	$\|\|\overrightarrow{u}\|\|_2^2\overrightarrow{v} + (\lambda-1)\langle\overrightarrow{u},\overrightarrow{v}\rangle\overrightarrow{u}$	$\lambda(=2)$
FullAdd	$W_1\overrightarrow{u} + W_2\overrightarrow{v}$	$W_1, W_2 \in \mathbf{R}^{m\times m}$
LF	$A_u\overrightarrow{v}$	$A_u \in \mathbf{R}^{m\times m}$

Table A.1: Currently implemented composition functions of inputs (u, v) together with parameters and their default values in parenthesis, where defined. Note that in LF each functor word corresponds to a separate matrix or tensor A_u (Baroni and Zamparelli, 2010).

Target	Method	Neighbours
belief	Corpus	moral, dogma, worldview, religion, world-view, morality, theism, tenet, agnosticism, dogmatic
false belief	FullAdd	pantheist, belief, agnosticism, religiosity, dogmatism, pantheism, theist, fatalism, deism, mind-set
	LF	self-deception, untruth, credulity, obfuscation, misapprehension, deceiver, disservice, falsehood
	Add	belief, assertion, falsity, falsehood, truth, credence, dogma, supposition, hearsay, denial

Table A.2: Top nearest neighbours of *belief* and of *false belief* obtained through composition with the FullAdd, LF and Add models.

Target	Method	Neighbours
florist	Corpus	Harrod, wholesaler, stockist
flora + -ist	FullAdd	flora, fauna, ecologist
	LF	ornithologist, naturalist, botanist
	Add	flora, fauna, ecosystem

Table A.3: Compositional models for morphology. Top 3 neighbours of *florist* using its (low-frequency) corpus-extracted vector, and when the vector is obtained through composition of *flora* and *-ist* with FullAdd, LF and Add.

the FullAdd, LF and Add models. In Table A.3, I exemplify a less typical application of compositional models to derivational morphology, namely obtaining a representation of *florist* compositionally from distributional representations of *flora* and *-ist* (Lazaridou et al., 2013).

A.4 Summary

The DISSECT toolkit, introduced here, provides a set of tools to obtain vector representations for words and phrases. It can be useful for practical applications that require similarities between words and phrases or for further researches on compositional models, where comparisons between models are called for.

DISSECT is designed to maximize usability and extensibility, therefore it can be easily extended. Some possible extensions include: adding further compositional methods or integrating modules for composing entire sentences, taking syntactic trees as input and returning composed representations for each node in the input trees.

List of publications

The contents described in this thesis have appeared in the following publications:

- Chapter 3: Dinu, G., Pham, N., and Baroni, M. (2013b). General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria. Dinu and Pham are equally contributing first authors

- Chapter 4: Paperno, D., Pham, N. T., and Baroni, M. (2014). A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*, pages 90–99, Baltimore, MD. Paperno and Pham are equally contributing first authors

- Chapter 5: Pham, N. T., Ferrone, L., and Zanzotto, F. M. (2014). Compositional distributional semantics models in chunk-based smoothed tree kernels. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 93–98, Dublin, Ireland. Association for Computational Linguistics and Dublin City University

- Chapter 6: Pham, N. T., Kruszewski, G., Lazaridou, A., and Baroni, M. (2015). Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings 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)*, pages 971–981, Beijing, China. Association for Computational Linguistics

- Appendix A: Dinu, G., Pham, N., and Baroni, M. (2013a). DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of ACL (System Demonstrations)*, pages 31–36, Sofia, Bulgaria. Dinu and Pham are equally contributing first authors

Bibliography

Abdi, H. and Williams, L. (2010). Newman-Keuls and Tukey test. In Salkind, N., Frey, B., and Dougherty, D., editors, *Encyclopedia of Research Design*, pages 897–904. Sage, Thousand Oaks, CA.

Abney, S. (1996). Part-of-speech tagging and partial parsing. In K.Church, S.Young, G., editor, *Corpus-based methods in language and speech*. Kluwer academic publishers, Dordrecht.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of HLT-NAACL*, pages 19–27, Boulder, CO.

Agirre, E., Cer, D., Diab, M., and Gonzalez-Agirre, A. (2012). SemEval-2012 Task 6: a pilot on semantic textual similarity. In *Proceedings of *SEM*, pages 385–393, Montreal, Canada.

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., and Guo, W. (2013). *SEM 2013 shared task: Semantic Textual Similarity. In *Proceedings of *SEM*, pages 32–43, Atlanta, GA.

Almuhareb, A. (2006). *Attributes in Lexical Acquisition*. Phd thesis, University of Essex.

Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of *SEM*, pages 435–440, Montreal, Canada.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247, Baltimore, MD.

Baroni, M., Evert, S., and Lenci, A., editors (2008). *Bridging the Gap between Semantic Theory and Computational Simulations: Proceedings of the ESSLLI Workshop on Distributional Lexical Semantic*. FOLLI, Hamburg.

Baroni, M. and Lenci, A. (2010). Distributional Memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Beltagy, I., Roller, S., Boleda, G., Erk, K., and Mooney, R. (2014). UTexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of SemEval*, pages 796–801, Dublin, Ireland.

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

Bernardi, R., Dinu, G., Marelli, M., and Baroni, M. (2013). A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of ACL (Short Papers)*, pages 53–57, Sofia, Bulgaria.

Blacoe, W. and Lapata, M. (2012). A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

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

Bullinaria, J. and Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

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

Clark, S. (2015). Vector space models of lexical meaning. In Lappin, S. and Fox, C., editors, *Handbook of Contemporary Semantics, 2nd ed.*, pages 493–522. Blackwell, Malden, MA.

Clark, S. and Curran, J. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Coecke, B., Sadrzadeh, M., and Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, Helsinki, Finland.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Corley, C. and Mihalcea, R. (2005). Measuring the semantic similarity of texts. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18. Association for Computational Linguistics, Ann Arbor, Michigan.

Croce, D., Moschitti, A., and Basili, R. (2011). Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1034–1046, Stroudsburg, PA, USA. Association for Computational Linguistics.

Curran, J. and Moens, M. (2002). Improvements in automatic thesaurus extraction. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 59–66, Philadelphia, PA.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. (2009). Recognizing textual entailment: rationale, evaluation and approaches. *Natural Language Engineering*, 15:459–476.

Dagan, I., Glickman, O., and Magnini, B. (2006). The pascal recognising textual entailment challenge. In et al., Q.-C., editor, *LNAI 3944: MLCW 2005*, pages 177–190. Springer-Verlag, Milan, Italy.

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

Dinu, G., Pham, N., and Baroni, M. (2013a). DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of ACL (System Demonstrations)*, pages 31–36, Sofia, Bulgaria.

Dinu, G., Pham, N., and Baroni, M. (2013b). General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Dumais, S. (2003). Data-driven approaches to information access. *Cognitive Science*, 27:491–524.

Erk, K. (2012). Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Erk, K. and Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, HI.

Evert, S. (2005). *The Statistics of Word Cooccurrences*. Ph.D dissertation, Stuttgart University.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Golub, G. and Van Loan, C. (1996). *Matrix Computations (3rd ed.)*. JHU Press, Baltimore, MD.

Grefenstette, E. (2013). *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. PhD thesis, University of Oxford Essex.

Grefenstette, E., Dinu, G., Zhang, Y.-Z., Sadrzadeh, M., and Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. In *Proceedings of IWCS*, pages 131–142, Potsdam, Germany.

Grefenstette, E. and Sadrzadeh, M. (2011a). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, pages 1394–1404, Edinburgh, UK.

Grefenstette, E. and Sadrzadeh, M. (2011b). Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the EMNLP GEMS Workshop*, pages 62–66, Edinburgh, UK.

Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Discovery*. Kluwer, Boston, MA.

Guevara, E. (2010). A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the EMNLP GEMS Workshop*, pages 33–37, Uppsala, Sweden.

Han, L., Kashyap, A., Finin, T., Mayfield, J., and Weese, J. (2013). UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of *SEM*, pages 44–52, Atlanta, GA.

Harris, Z. (1954). Distributional structure. *Word*, 10(2-3):1456–1162.

Hassan, S. and Mihalcea, R. (2011). Semantic relatedness using salient semantic analysis. In *Proceedings of AAAI*, pages 884–889, San Francisco, CA.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning, 2nd edition*. Springer, New York.

Hill, F., Cho, K., Jean, S., Devin, C., and Bengio, Y. (2014). Not all neural embeddings are born equal. In *Proceedings of the NIPS Learning Semantics Workshop*, Montreal, Canada. Published online: `https://sites.google.com/site/learningsemantics2014/`.

Huang, E., Socher, R., Manning, C., and Ng, A. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665, Baltimore, MD.

Kartsaklis, D. and Sadrzadeh, M. (2013). Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of EMNLP*, pages 1590–1601, Seattle, WA.

Kartsaklis, D., Sadrzadeh, M., and Pulman, S. (2013). Separating disambiguation from composition in distributional semantics. In *Proceedings of CoNLL*, pages 114–123, Sofia, Bulgaria.

Karypis, G. (2003). CLUTO: A clustering toolkit. Technical Report 02-017, University of Minnesota Department of Computer Science.

Katrenko, S. and Adriaans, P. (2008). Qualia structures and their impact on the concrete noun categorization task. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*, pages 17–24, Hamburg, Germany.

Klein, D. and Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430, Sapporo, Japan.

Lai, A. and Hockenmaier, J. (2014). Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Lazaridou, A., Marelli, M., Zamparelli, R., and Baroni, M. (2013). Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of ACL*, pages 1517–1526, Sofia, Bulgaria.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Lee, D. and Seung, S. (2000). Algorithms for Non-negative Matrix Factorization. In *Proceedings of NIPS*, pages 556–562.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Levy, O. and Goldberg, Y. (2014a). Dependency-based word embeddings. In *Proceedings of ACL (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland.

Levy, O. and Goldberg, Y. (2014b). Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL*, pages 171–180, Ann Arbor, MI.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval*, pages 1–8, Dublin, Ireland.

Mehdad, Y., Moschitti, A., and Zanzotto, F. M. (2010). Syntactic/semantic structures for textual entailment recognition. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1020–1028, Stroudsburg, PA, USA. Association for Computational Linguistics.

Melamud, O., Dagan, I., Goldberger, J., Szpektor, I., and Yuret, D. (2014). Probabilistic modeling of joint-context in distributional similarity. In *Proceedings of CoNLL*, pages 181–190, Ann Arbor, MI.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. `http://arxiv.org/abs/1301.3781/`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, Lake Tahoe, NV.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, pages 746–751, Atlanta, Georgia.

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

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Montague, R. (1970). English as a Formal Language. In et al., B. V., editor, *Linguaggi nella Societaà e nella Tecnica*, pages 189–224. Edizioni di Comunità. Reprinted in Montague (1974), 188–221.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. *Computional Linguistics*, 34(2):193–224.

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

Paperno, D., Pham, N. T., and Baroni, M. (2014). A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*, pages 90–99, Baltimore, MD.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, Doha, Qatar.

Pham, N. T., Bernardi, R., Zhang, Y.-Z., and Baroni, M. (2013). Sentence paraphrase detection: When determiners and word order make the difference. In *Proceedings of the Towards a Formal Distributional Semantics Workshop at IWCS 2013*, pages 21–29, Potsdam, Germany.

Pham, N. T., Ferrone, L., and Zanzotto, F. M. (2014). Compositional distributional semantics models in chunk-based smoothed tree kernels. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 93–98, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Pham, N. T., Kruszewski, G., Lazaridou, A., and Baroni, M. (2015). Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings 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)*, pages 971–981, Beijing, China. Association for Computational Linguistics.

Rothenhäusler, K. and Schütze, H. (2009). Unsupervised classification with dependency based word spaces. In *Proceedings of the EACL GEMS Workshop*, pages 17–24, Athens, Greece.

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

Salton, G. (1979). Mathematics and information retrieval. *Journal of Documentation*, 20(1):33–53.

Socher, R., Huval, B., Manning, C., and Ng, A. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642, Seattle, WA.

Tomasello, M. (2003). *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Harvard University Press, Cambridge, MA.

Turian, J., Ratinov, L.-A., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394, Uppsala, Sweden.

Turney, P. (2012). Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

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

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Zanzotto, F., Korkontzelos, I., Falucchi, F., and Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.

Zhao, J., Zhu, T., and Lan, M. (2014). Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.