Comparison of two semantic aware composition models for word embeddings and its relation to dependency type information

Student Research Paper submitted

to

Prof. Ulf Leser

Humboldt-Universität zu Berlin
Department of Computer Science
Knowledge Management in Bioinformatics

by

Arne Binder
(520999)

Berlin, May 4, 2018

Abstract

Comparison of two semantic aware composition models for word embeddings and its relation to dependency type information

by Arne Binder

Vector Space Models (VSMs) for textual data lead to success in many Natural Language Processing (NLP) tasks. Recently, prediction based word embedding models like word2vec gained attention. These models build upon Distributional Semantics, i.e. a word is defined by its contexts, and scale up to billions of training tokens resulting in robust embeddings for individual words. Compositional Distributional Semantics Models (CDSMs) intend to create vector representations for sequences of tokens by composing word embeddings in a meaningful manner. However, it is up to debate which composition functions perform well for semantic tasks.

In this work, we study the impact of order aware processing to token embedding composition at sentence level by implementing (1) an averaging model and (2) a Long Short-Term Memory (LSTM) based approach. Furthermore, we analyze the relation of order aware composition to syntactical information. We evaluate our models at the SICK relatedness prediction task.

Our results underpin the thesis, that order aware processing is useful for semantic aware composition and subsumes syntactical information in most cases. However, there are instances of linguistic constructions in which syntactical information seems to be superior to order aware processing, namely in the presence of passive.

Contents

List of Abbreviations

ANN	Artificial Neural Network
AVG	averaging
CDSM	Compositional Distributional Semantics Model
CNN	Convolutional Neural Network
DAN	Deep Averaging Network
DG	Dependency Grammar
DSM	Distributional Semantics Model
FNN	Feedforward Neural Network
FC	fully connected layer
GRU	Gated Recurrent Unit
LSA	Latent Semantic Indexing
LSTM	Long Short-Term Memory
IQR	interquartile range
IR	Information Retrieval
MSE	Mean Squared Error
NLP	Natural Language Processing
PMI	Pointwise Mutual Information
PPMI	Positive Pointwise Mutual Information
RecNN	Recursive Neural Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SGNS	Skip-Gram model with Negative Sampling
STD	standard deviation
STS	Semantic Textual Similarity
SVD	Singular Value Decomposition
TF-IDF	term frequency-inverse document frequency
VSM	Vector Space Model

1 Introduction

Representations of textual content that are processable by computer algorithms form the basis for many applications that require to understand human utterances up to a certain extend. A common way is to transform textual data into high dimensional vector space representations, known as VSM(Salton, Wong, and Yang 1975). These vector representations function as interface to a wide range of machine learning algorithms and leverage efficient approaches for many NLP tasks like document retrieval (Manning, Raghavan, and Schütze 2008), text classification (Sahami et al. 1998), machine translation (Wu et al. 2016) or argument clustering (Boltuzic and Šnajder 2015). However, the level of machine understanding is still not comparable to human performance. Recently, the Common Round[1] project (Uszkoreit et al. 2017) introduces a platform to facilitate large scale online debating with NLP technologies. The authors define aggregation of the semantic content of debates as one of their major objectives which they intend to achieve by clustering similar arguments. That approach requires a scalable, semantically consistent similarity measure for text, emphasizing the need for a well structured semantic vector space.

Improving upon VSMs, frameworks like word2vec (Mikolov, Sutskever, et al. 2013) or glove (Pennington, Socher, and Manning 2014) lead to success in many NLP tasks by producing dense vector representations for single words. The cosine distance of these so called *word embeddings* enables a semantically consistent similarity measure for word tokens, i.e. the distances between word vectors approximate human intuition. CDSMs introduced by Clark, Coecke, and Sadrzadeh (2008) exploit this by combining sequences of word embeddings in a functional manner to produce vector representations of arbitrary textual utterances that are, to a degree, semantical consistent. However, there is an ongoing debate how to effectively combine word embeddings for gaining better performance. Several approaches build on simple summation or averaging (Habernal and Gurevych 2015; Boltuzic and Šnajder 2015; Misra, Ecker, and Walker 2016). Then again, Misra, Ecker, and Walker (2016) state that averaging all word embeddings may lose too much information in long sentences. Wang and Zong (2017) present an overview of different embedding composition models for phrase representations. They conclude that the recurrent LSTM model (Hochreiter and Schmidhuber 1997) that processes tokens in order just slightly outperforms the additive, order unaware baseline model in this task. Tai, Socher, and Manning (2015) introduced an additional level of complexity by presenting the recursive TreeLSTM model. This model further reduces the vanishing gradient

[1]see http://commonround.dfki.de/

problem of recurrent models by shortening the average distance of entities in the computation graph. The authors use dependency parse trees to construct the neural model and present promising results for the SICK phrase relatedness task[2].

Recently, Mueller and Thyagarajan (2016) presented a simple LSTM based approach to combine word embeddings for phrase relatedness prediction. Although their model does not require any syntactical information as dependency structure, it still outperforms the TreeLSTM model by Tai, Socher, and Manning (2015). Furthermore, Iyyer et al. (2015) introduces the Deep Averaging Network (DAN) that performs well on several semantic tasks without exploiting any ordering information of the tokens. It seems to be an open question what kind of structural information impacts semantic aware processing to what degree.

1.1 Objective

In this work we examine what degree of semantic awareness is achievable with two different embedding composition models, when applied to sentences (Section 1.2). We regard a composition method for word embeddings as semantic aware if it yields a distance measure that matches human intuition. Furthermore, we study the impact of providing additional syntactical data for the respective architectures.

1.2 Approach

To achieve these goals we implement: (a) an averaging model, and (b) a neural sequence model. We evaluate the semantic awareness with the SICK Semantic Textual Similarity (STS) challenge (Marelli, Menini, et al. 2014) and compare the model performances against a TF-IDF baseline.

As neural sequence model, we use an approach based on Mueller and Thyagarajan (2016) due to its simplicity. It consists of one LSTM as composition model which is applied to two input sentences and uses the Manhattan metric as distance measure. Nevertheless, the authors report a Pearson's r of 0.8822 for the SICK challenge. In our averaging setting, every word embedding is mapped by one identical, non-linear transformation before their aggregation.[3]

We analyze the effect of syntactical information by optionally appending one-hot encoded dependency type data to the token embedding vectors.

[2]The SICK corpus (Marelli, Menini, et al. 2014) contains ∼10.000 similarity scored sentence pairs. The system by Tai, Socher, and Manning (2015) achieved a Pearson's r of 0.8676 when predicting similarities.

[3]At time of experiment conduct, we were not aware of the Deep Averaging Network model by Iyyer et al. (2015) which could give further insights about the impact of order aware semantic composition.

2 Theoretical Background

This chapter provides an overview of basic concepts regarding semantic aware vector space representations and neural machine learning. Furthermore, we briefly introduce linguistic dependency types.

2.1 Evaluation of Semantic Awareness

Following the Distributional Hypothesis (Harris 1954), that linguistic items with similar contexts have similar meanings, it should be possible to construct a conceptional space in a way, that distances of concepts follow human intuition. Distributional Semantics Models (DSMs) (Landauer and Dumais 1997; Schütze 1998) are approaches that implement this thesis.

With semantic awareness we describe the property of how much meaning a representational space is able to preserve with respect to the space of origin, textual natural language. Furthermore, the theoretical state of being semantic aware references the maximal achievable semantic awareness, i.e. conveying the same meaning as the origin of the representation.

With regard to the Distributional Hypothesis semantic awareness can be measured by the correlation of the distances between different concepts in the representational space and the distances between its representations in the original space. Although the concept of *distance* seems to be more intuitive in the context of representational spaces, its inverse, the concept of *similarity*, commonly serves as base of comparison.

Given similarity measures that map from both spaces into the same space $X \subseteq \mathbb{R}$, we can borrow the task of *similarity prediction* for evaluation of semantic awareness when using *semantic relatedness* as underlying similarity metric of X. With the concept of semantic relatedness we refer to the interpretation by Budanitsky and Hirst (2006): *linked by any kind of lexical or functional association.* Resnik (1999) visualizes this concept by contrasting it with *semantic similarity.* The authors state that "car" and "gasoline" are related, but not semantically similar, whereas "car" and "bicycle" are semantic related *and* similar. Thus, semantic similarity covers only a subset of semantic relatedness and captures relations like synonymy or hypernymy while semantic relatedness further captures antonomy, meronymy and others.

The capability of predicting this kind of similarity by using an internal representation requires by definition some degree of semantic awareness. We are going to evaluate to which extent that might be the case in different scenarios.

2.1.1 Similarity Prediction

The task of Similarity Prediction can be framed as to predict a score given two entities of a conceptional space, where higher scores represent higher degrees of similarity between them. In the context of NLP these entities are linguistic units such as words, phrases, sentences or documents. The score can include different granularities of similarity, ranging from binary classification, e.g. in the case of Paraphrase Detection[4], weighted aggregation of multi-label classification results, e.g. for instances of discrete questionnaire data, to regression with continuous scores.

We will focus on Similarity Prediction as regression task. Semantic Relatedness of arbitrary[5] composed textual entities is no binary phenomena. Otherwise the entity space would disintegrate into distinct clusters each containing entities interpreted as identical.

2.1.2 Evaluation Measures

Common measures for evaluation of regression estimators are *Mean Squared Error* (see Lehmann and Casella 1998), *Pearson correlation coefficient* (Pearson 1895) and *Spearman's rank correlation coefficient* (Spearman 1904).

The Mean Squared Error (MSE) measures the deviation of the predicted values to the correct ones. It incorporates the variance and the bias of the estimator. A value of zero denotes a perfect estimator and larger values indicate lower estimator performance. Given n predictions Y and corresponding correct values X, the MSE is defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2 \tag{1}$$

The Pearson correlation coefficient (Pearson's r) expresses the degree of linear correlation of two variables. It is the covariance of the variables normalized by their standard deviations. It ranges from -1, denoting perfect negative correlation, to 1, denoting perfect positive correlation. A value of zero expresses that the two variables do not correlate at all. Therefore, higher values above zero indicate better estimator performance. Given X and Y, Pearson's r is defined as:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \cdot \sum_{i=1}^{n}(Y_i - \overline{Y})^2}} = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}, \tag{2}$$

[4]Paraphrase Detection is the task of identifying if two phrases paraphrase each other.

[5]The composition is surely constrained by grammar, linguistic performance and other criteria, but this term is in favor of the tremendous variability of natural language.

where $\overline{X}$ is the empirical mean of X, $Cov(X,Y)$ is the empirical covariance of X and Y, and $\sigma(X)$ is the empirical standard deviation.

Spearman's rank correlation coefficient (Spearman's ρ) serves as another correlation based measure, which does not assume linear dependence between X and Y. Hence, it is more robust regarding outliers. Spearman's ρ can be interpreted as special case of Pearson's r where the data is converted into ranks before calculation of Pearson's r. Given X and Y, Spearman's ρ is defined as:

$$\rho = \frac{\sum_{i=1}^{n}(rg(X)_i - \overline{rg(X)})(rg(Y)_i - \overline{rg(Y)})}{\sqrt{\sum_{i=1}^{n}(rg(X)_i - \overline{rg(X)})^2 \cdot \sum_{i=1}^{n}(rg(Y)_i - \overline{rg(Y)})^2}} = \frac{Cov(rg(X), rg(Y))}{\sigma(rg(X))\sigma(rg(Y))}, \tag{3}$$

where $rg(X)$ are the ranks of X.

2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a machine learning model where input vectors are mapped to predictions by successive application of computational *layers* (David E. Rumelhart and McClelland 1986). Each layer is a differentiable function $l^{(i)} : \mathbb{R}^{n^{(i-1)}} \mapsto \mathbb{R}^{n^{(i)}}$, with $n^{(i)}$ denoting the size of the i-th layer. The arrangement of layers form the network graph, which is (in most cases) a directed graph defining the specific network architecture. Provided adequate training data, ANNs can be trained efficiently with gradient descent based optimization algorithms. In the next sections we will describe some popular architectures and important components of ANNs.

2.2.1 Feedforward Networks

The simplest class of ANNs are Feedforward Neural Networks (FNNs) that emerged from multilayer Perceptrons (Rosenblatt 1958). Their network architecture forms an acyclic graph and is defined by $f^{FNN}(v) = (l^{(N)} \circ ... \circ l^{(i)} \circ ... \circ l^{(1)})(v)$ where N is the total amount of layers. The N-th layer is called *output layer*, all other layers are *hidden layers*. In addition, a virtual *input layer* models the input. Figure 1 (Epelbaum 2017) shows such a structure.

There exists a diverse landscape of different types of layers. A common structure is the fully connected layer (FC). It connects every node from the previous layer with every node from the current one in the form of a weighted sum . Hence, it calculates an affine transformation of its input. Usually a layer $l^{(i)}$ is parametrized by $\theta^{(i)}$, e.g. a set of *weights* and *biases*. In this manner, a FC can be defined as:

$$l^{FC}(v; \theta^{(i)}) = w^{(i)}v + b^{(i)} \tag{4}$$

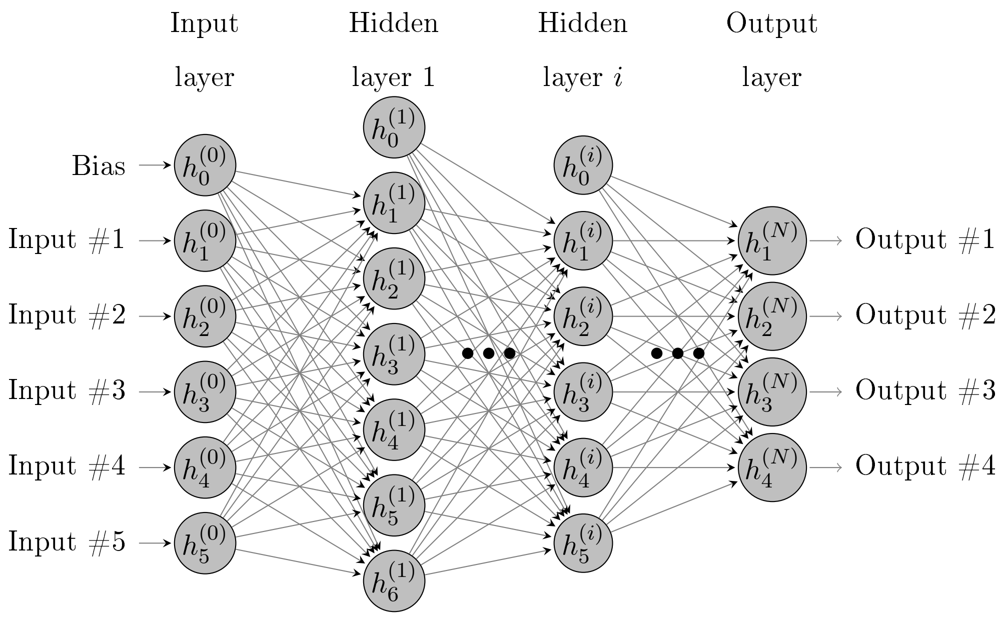


Figure 1: Neural Network with $N+1$ layers ($N-1$ hidden layers).

where $w^{(i)} \in \mathbb{R}^{n^{(i)} \times n^{(i-1)}}$, $b^{(i)} \in \mathbb{R}^{n^{(i)}}$ and $\theta^{(i)} = \{w^{(i)}, b^{(i)}\}$.

2.2.2 Activation Functions

The Feedforward Neural Network (FNN) described so far is capable of modeling linear mappings only. To enable non-linearity, different non-linear activation functions can be applied element-wise to the individual layer outputs. A prominent instance is the *sigmoid* function that takes a real-valued input and "squashes" it to range between 0 and 1. Therefore, it is used to model Bernoulli-Distributions. It is defined as:

$$\sigma(v_i) = \frac{1}{1+e^{-v_i}} \tag{5}$$

The *hyperbolic tangent* is another strictly increasing, non-linear activation function, but maps into the range between -1 and 1:

$$\tanh(v_i) = \frac{1-e^{-2v_i}}{1+e^{-2v_i}} = 2\sigma(2v_i) - 1 \tag{6}$$

Finally, the *softmax* function can be used to model discrete probability distributions, e.g. for classification tasks. In contrast to the previous activation functions it is not element wise independent, since it applies normalization across all input elements to guarantee that its result follow the probability distribution conditions. For an input vector x with entries x_i it is defined as:

$$softmax(v_i) = \frac{e^{v_i}}{\sum\limits_{v_j \in v} e^{v_j}} \tag{7}$$

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) introduced by Hopfield (1982) are ANNs that are capable of processing arbitrary long sequences when feeding them stepwise into the network. Additionally, they can access information from any previous step and, hence, process the new data conditioned by historical data. This is possible via a feedback loop in a *recurrent layer*. Given a sequence of τ-sized vectors $v^{(1)}, ..., v^{(\eta)}$ the t-th output $h^{(t)}$ of a recurrent layer l^{RNN} is defined as:

$$h^{(t)} = l^{RNN}(v^{(t)}; \theta) = g(v^{(t)}; h^{(t-1)}; \theta) \tag{8}$$

where $k, \vartheta \in \mathbb{N}$, k is the inner state size and g is the Recurrent Neural Network (RNN) cell function with $g : (\mathbb{R}^\tau; \mathbb{R}^k; \mathbb{R}^\vartheta) \mapsto \mathbb{R}^k$.

The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) is a prominent RNN. It is a *gated* RNN, meaning its cell function g^{LSTM} consists of multiple interconnected gating functions that model a dedicated cell state $\tilde{c}$ (Equation 9). LSTMs tackle one major drawback of the RNN architecture, namely the vanishing gradient problem that occurs when training with long sequences. Without gating, repeated application of RNN cell to the elements of a sequence often leads to exponential decrease of the calculated gradients because common activation functions produce gradients lesser 1 that are multiplied according to the chain rule. That effect practically hinders the network to use long-distance information. Section 2.3.2 explains the exact role of gradients in neural network training.

The different gates of the LSTM are responsible for deciding what information should be forgotten (g_f), which new information will be added (g_i) and, finally, which information from the cell state $\tilde{c}$ will be presented as output (g_o):

$$\begin{aligned}
g_f^{(t)} &= \sigma(l^{FC}([\tilde{h}^{(t-1)}, v^{(t)}]; \theta^{(f)})) \\
g_i^{(t)} &= \sigma(l^{FC}([\tilde{h}^{(t-1)}, v^{(t)}]; \theta^{(i)})) \\
g_o^{(t)} &= \sigma(l^{FC}([\tilde{h}^{(t-1)}, v^{(t)}]; \theta^{(o)})) \\
c^{(t)} &= tanh(l^{FC}([h^{(t-1)}, v^{(t)}]; \theta^{(c)})) \\
\tilde{c}^{(t)} &= g_f^{(t)} \odot \tilde{c}^{(t-1)} + g_i^{(t)} \odot \tilde{c}^{(t-1)} \\
\tilde{h}^{(t)} &= g_o^{(t)} \odot tanh(\tilde{c}^{(t)})
\end{aligned} \tag{9}$$

and finally

$$g^{LSTM}(v^{(t)};[\tilde{c}^{(t-1)},\tilde{h}^{(t-1)}]) = [\tilde{c}^{(t)},\tilde{h}^{(t)}] \tag{10}$$

where $[a,b]$ is the concatenation of the vectors a and b, and $a \odot b$ denotes their element-wise product (Hadamard product). We omit the parameter set $\theta^{LSTM} = \{\theta^{(f)},\theta^{(i)},\theta^{(o)},\theta^{(c)}\}$ for better readability.

2.3 Supervised Training of ANNs

Supervised training of an ANN model f parametrized by θ takes a set Ω of input-output training tuples (x,y) for granted and tries to find instantiations of θ in the manner that $f(x;\theta) = \hat{y} \approx y, \forall (x,y) \in \Omega$ (Mohri, Rostamizadeh, and Talwalkar 2012). In the context of ANNs, one usually uses a *cost function* J that quantifies the deviation of $\hat{y}$ from y, thus J is used to guide the parameter adjustment via gradient-based optimization to minimize this cost.

2.3.1 Cost Function

The cost function, or *loss function*, most commonly used for regression tasks is Mean Squared Error (MSE) as described in section 2.1.2, because it yields the maximum likelihood of y and $\hat{y}$ if the error is normally distributed and because it is efficient to calculate. We reformulate equation (1) to fit into our needs:

$$J_{\Omega}^{MSE}(\theta) = \frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{\iota-1}(y_{ij} - \hat{y}_{ij})^2 \tag{11}$$

where n is the amount of tuples (x_i, y_i) in Ω and $\hat{y}_i = f(x_i;\theta)$ with $y_i, \hat{y}_i \in \mathbb{R}^{\iota}$.

2.3.2 Gradient-based Optimization

If the loss function J_{Ω} is differentiable, in theory it would be possible to calculate its global minimum θ^* analytically. However, in practice this is infeasible due to the number of parameters in θ that can be in the order of millions. For instance, the VGG network (Simonyan and Zisserman 2014) uses up to 144 million trainable weights for image recognition tasks.

Gradient-based optimization tackles this problem by shifting the parameters θ stepwise in the negative direction of its gradients ∇J_{Ω}. By definition of gradients, an infinitesimal displacement of θ in this direction decreases $J_{\Omega}(\theta)$. Hence, we select the parameters for the next optimization step θ':

$$\theta' = \theta - \alpha \nabla J_{\Omega}(\theta) \tag{12}$$

where α is the chosen step size, or *learning rate*, that has to be sufficiently small. On the other hand, it is unfavorable to choose a step size too small because that increases convergence time, meaning it slows down training. Also different approaches exist that apply dynamical sized stepping like *Momentum* (Polyak 1964), which takes previous parameter updates into account, or *AdaDelta* (Zeiler 2012) that tracks individual learning rates for the parameters in θ. ADAM (Kingma and Ba 2014) is another robust method to guide the parameter optimization by keeping track of the gradients and its square roots.

As equation (11) states, one update step requires summation over all training examples in Ω, so it is an $O(n)$ operation. Hence, with large training sets calculation of $\nabla J_{\Omega}(\theta)$ for every step becomes infeasible. Stochastic Gradient Descent (SGD) addresses this problem by randomly partitioning Ω into subsets Ω_i^P that are used successively to calculate new parameter sets θ'. Exhausting all Ω_i^P, also referred to as *(mini-)batches*, of a certain partition Ω^P is called finishing an *epoch*. This approach stochastically approximates the calculation of $\nabla J_{\Omega}(\theta)$.

Finally, we describe the core algorithm that enables gradient-based training of ANNs, namely *Backpropagation* (David E Rumelhart, G. E. Hinton, and Williams 1988). It uses dynamic programming for efficient calculation of the gradients through the different layers by exploiting their interconnectivity as present in common ANNs. As stated before, an ANN f can be represented as a composition of layer functions $l^{(i)}$ with $f(v) = (l^{(N)} \circ \ldots \circ l^{(i)} \circ \ldots \circ l^{(1)})(v)$ and $l^{(i)}(v^{(i)}) = v^{(i+1)}$. Hence, the *chain rule* of gradient computation defines the derivatives for two successive layers as:

$$\frac{dv^{(i+2)}}{dv^{(i)}} = \frac{dv^{(i+2)}}{dv^{(i+1)}} \frac{dv^{(i+1)}}{dv^{(i)}} \tag{13}$$

$$= l^{(i+1)'}(v^{(i+1)}) l^{(i)'}(v^{(i)}) \tag{14}$$

$$= l^{(i+1)'}(l^{(i)}(v^{(i)})) l^{(i)'}(v^{(i)}) \tag{15}$$

$v^{(i)}$ occurs several times in equation (15). Furthermore, as shown in (13) $\frac{dv^{(i+2)}}{dv^{(i)}}$ depends on $\frac{dv^{(i+2)}}{dv^{(i+1)}}$ immediately. Backpropagation exploits these redundancies by successively calculating all $v^{(i)}$ in a *forward pass*. Then, it calculates all gradients in reverse order while reusing intermediate results in a *backwards pass*. By doing so, the algorithm reduces the complexity from exponential, considering a naive approach, to linear with regard to the edge count of the network graph.

2.4 A Model for Semantic Relatedness Prediction

We divide semantic relatedness prediction with regard to textual inputs into (1) transformation of the inputs into vector space representations and (2) similarity calculation in this embedding space.

2.4.1 Vector Space Embeddings

We define *embeddings* of documents, words or other linguistically motivated units as vector representations that are to a degree semantically aware. *Embedding models* are transformations that map these units into an *embedding space.*

Because of the relevance of embedding models for many NLP tasks, a diverse landscape of different approaches has evolved. In the following, we distinguish traditional document embedding approaches from composition based models. By traditional approaches we refer to models which process sequences of opaque tokens and directly produce embeddings for these sequences (see Turney and Pantel 2010, for an exhaustive study), whereby composition based models (Clark, Coecke, and Sadrzadeh 2008; Grefenstette and Sadrzadeh 2011) initially embed each token independently and compose these token embeddings into a final vector space representation.

2.4.2 Traditional Document Embeddings

Vector Space Models (VSMs) (Turney and Pantel 2010) build upon occurrences of terms T within certain contexts C. For example, a word can occur as term in a document, i.e. its context, or not. In VSMs contexts are expressed in means of terms that occur with these. Thus, these models are instances of DSMs (Section 2.1). We define $T = \{t_1, ..., t_m\}$ as finite set of opaque symbols with $m = |T|$ and the set of contexts $C \subseteq T^*$ with $n = |C|$, e.g. every context is a set of terms. The set of unique terms T is called a vocabulary.We define a *context embedding* as a function $f_C : C \mapsto \mathbb{R}^{\hat{n}}$, where $\hat{n} \in \mathbb{N}$ is the dimensionality of the embedding vectors.

One popular embedding model is based on the term frequency-inverse document frequency (TF-IDF) measure. TF-IDF was introduced as *term specificity* in Sparck Jones (1972). Given T and C, the term frequency tf of a certain term t with respect to a context c is defined as the count of occurrences of t in c normalized by the maximum of the term counts in this context:

$$tf(t,c) = \frac{\#(t,c)}{max_{t' \in c}\#(t',c)} \tag{16}$$

where $\#(t,c)$ is the frequency of term t in context c. The inverse document frequency *idf* of a term t regarding contexts C is defined as:

$$idf(t) = \frac{|C|}{|\{c \in C : t \in c\}|} \tag{17}$$

Finally, the term frequency-inverse document frequency *tf.idf* is calculated:

$$tf.idf(t,c) = tf(t,c) \cdot idf(t) \tag{18}$$

Hence, given a corpus containing documents as contexts C of terms T, it is possible to calculate a $|C| \times |T|$ TF-IDF matrix $M^{tf.idf}$ with $M^{tf.idf}_{ij} = tf.idf(t_j, c_i)$. The columns of $M^{tf.idf}$ represent embeddings for the contexts, respectively documents. The TF-IDF context embedding $f^{tf.idf}_C$, i.e. the TF-IDF document representation, is defined as $f^{tf.idf}_C(c_j) = M^{tf.idf} \cdot e_j$ where e_j is the j-th unit vector. In the following we call a matrix M^x whose entries M^x_{ij} quantify in any way the occurrences of term t_i in context c_i a *occurrence matrix*.[6]

Another common approach utilizes Pointwise Mutual Information (Church and Hanks 1990). Pointwise Mutual Information (PMI) quantifies the association of two outcomes of discrete random variables X and Y. Given outcomes x and y belonging to these variables, the PMI is defined as:

$$pmi(x;y) \equiv log \frac{p(x,y)}{p(x)p(y)} \tag{19}$$

By interpreting T and C as random variables X and Y an occurrence matrix M^{pmi} can be constructed as follows:

$$\begin{aligned} M^{pmi}_{ij} &= log \frac{\#(t_i, c_j)}{\sum\limits_{c \in C} \#(t_i, c) \cdot \sum\limits_{t \in T} \#(t, c_j)} \\ &= log\, \#(t_i, c_j) - b^{(T)}_i - b^{(C)}_j \end{aligned} \tag{20}$$

where $b^{(T)}_i$ and $b^{(C)}_j$ are term and context biases with

$$\begin{aligned} b^{(T)}_i &= log \sum_{c \in C} \#(t_i, c) \quad \text{and} \\ b^{(C)}_j &= log \sum_{t \in T} \#(t, c_j) \end{aligned} \tag{21}$$

Because $\#(T_i, C_j)$ is zero and, consequently, M^{pmi}_{ij} equals $-\infty$ for a lot of entries, often the Positive Pointwise Mutual Information (PPMI) (Niwa and Nitta 1994) is used:

$$M^{ppmi}_{ij} = max(M^{ppmi}_{ij}, 0) \tag{22}$$

[6]We chose this wording instead of *co-occurrence*, that would theoretically fit better, because co-occurrence is primary associated with the co-occurrence of aspects of the same or similar type, i.e. word-word events, in common literature and is not expected to cover document-word events, for instance.

By construction these document representations have as many dimensions as terms in T. But according to Zipf's word frequency law in natural language (Zipf 1935), a huge majority of words occurs very rarely, e.g., only in very few documents of a corpus. Thus, the majority of values in an occurrence matrix M is zero. In other words, the matrix is very *sparse* and has a low information density. Furthermore, some words occur in almost every document leading to high biases. Assuming they do not convey particular meaning and low frequency words are too rare to capture their meaning by containing documents, these extreme cases are commonly filtered out. However, occurrence matrices are still sparse by nature. For instance, one might imagine a corpus of 1 million documents containing words of a (pruned) vocabulary of size 100,000 resulting in 100 billion matrix entries requiring a huge amount of occurrence events to fill the majority of entries.

To combat the problem of sparsity, there are several methods to reduce the embedding dimensionality while maintaining semantic awareness. One common approach is to apply Singular Value Decomposition (SVD) (Beltrami 1873) to the occurrence matrix. Latent Semantic Indexing (LSA) (Deerwester et al. 1990) implements this in the context of Information Retrieval (IR) by decomposing the occurrence matrix M described above[7] in the following way:

$$M = U \cdot S \cdot V^T \tag{23}$$

where $U = MM^T$, $V = M^T M$ and S is a diagonal matrix that contains the singular values of M. Furthermore, U and V are orthogonal bases, whereby U spans a term based vector space and V a document based vector space. By this decomposition, a dimensionality reduction can be achieved by removing the smaller singular values from S resulting in the submatrix $S_{(\hat{n})}$ that contains only the $\hat{n}$ highest eigenvalues. This is a reasonable step because $S_{(\hat{n})}$ and the corresponding matrices $U_{(\hat{n})}$ and $V_{(\hat{n})}$ produce the rank $\hat{n}$ approximation to M with the smallest error[8](Hofmann 2001). Mapping a document representation d, where d consists of entries as in M, into the reduced dimensional space spanned by $V_{(\hat{n})}$ can be computed by:

$$\hat{d} = S^{-1}_{(\hat{n})} U^T_{(\hat{n})} d \tag{24}$$

where $\hat{d}$ is the $\hat{n}$-dimensional document representation.

Despite its clear theoretical motivation, VSMs for documents that build upon SVD have the major draw back that the decomposition has to be calculated every time a document is

[7]The original LSA makes use of the *term document matrix* M^{td}, a simplification of M^{pmi} without normalization, where $M^{td}_{ij} = \#(T_i, C_j)$.

[8]according to Frobenius norm

added to the corpus, which functions as base for the semantic space. SVD is computational expensive, it has a complexity of $O(min(\{mn^2, m^2n\}))$.

2.4.3 Term Embeddings

Term embeddings, in their common sense, are distributed representations of words (Bengio et al. 2003), thus forming DSMs (Section 2.1). Similar to context embeddings, we define a term embedding as a function $f_T : T \mapsto \mathbb{R}^{\hat{m}}$, where $\hat{m} \in \mathbb{N}$ is the dimensionality of the embedding vectors.

The methods described in the previous section are directly applicable to produce term embeddings by defining $f_T(t_i) = M^T \cdot e_i$ where $M \in \mathbb{R}^{m \times n}$ is an occurrence matrix with $m = |T|$ and $n = |C|$ as defined above. A common approach to produce term embeddings is to calculate M^{ppmi} and apply SVD. In analogy to $\hat{d} = S_k^{-1} U_k^T d$ (equation (24)), a term embedding $\hat{t}$ of the context based term vector t for term T_i can be calculated as $\hat{t} = S_k^{-1} V_k^T t$. In this means, it is not necessary to recalculate the SVD until a term T_x occurs that is not in T. However, in that case, there are probably not enough documents available to construct a consistent representation for T_x either, so it can and should be discarded.

Further ideas like the Hyperspace Analogue to Language model (Lund and Burgess 1996) build upon counts of word co-occurrences. They can be cast into the conceptional framework outlined by defining C via a co-occurrences relation R^{CO} as:

$$C_j = \{t_i | (t_i, t_j) \in R^{CO}\} \tag{25}$$

where $R^{CO} = \{(t_i, t_j) | t_i, t_j \in T;\ t_i \text{ and } t_j \text{ are successive tokens}\}$. This approach can be enhanced by relaxing the co-occurrence relation, e.g. defining R^{CO} via a fixed or soft sized co-occurrence window larger than two and optionally adding distance weighting.

In line with Levy, Goldberg, and Dagan (2015) we refer to the term embedding models presented so far as *count based* models and contrast them with *prediction based* models. Prediction based embedding models commonly build on ANNs whose intermediate low dimensional weights are exploited as term embeddings. They raised attention due to recently published methods that allow efficient embedding calculation with regard to billions of tokens. The Skip-Gram model with Negative Sampling (SGNS)[9] (Mikolov, Sutskever, et al. 2013) is a popular candidate that does not rely on decomposing any occurrence matrix, but trains a simple, one hidden layer ANN with the objective to predict the context of a term, e.g. the

[9] A SGNS implementation was released within the word2vec framework (Mikolov, K. Chen, et al. 2013).

neighbors of the tokens in a textual corpus. The core Skip-Gram idea can be formalized as:

$$
\begin{aligned}
f^{SG}(t_i)_j &= p(c_j|t_i) \\
&= (softmax \circ l^{FC}_{out} \circ l^{FC}_{hidden})(e_i) \cdot e_j \\
&= softmax(l^{FC}_{out}(l^{FC}_{hidden}(e_i; \{w_H; b_H\}); \{w_O; b_O\}) \cdot e_j \\
&= softmax(w_O(w_H \cdot e_i + b_H) + b_O) \cdot e_j \\
&= softmax([w_O, b_O^T] \cdot [w_H, b_H^T] \cdot e_i) \cdot e_j \\
&= softmax(M^{out} \cdot M^{hidden} \cdot e_i) \cdot e_j
\end{aligned}
\tag{26}
$$

with e_x being the x-th unit vector. Then, the Skip-Gram term embedding can be defined as $f^{SG}_T(t_i) = M^{hidden} \cdot e_i$. Although M^{hidden} does not immediately rely on occurrence counts, Levy and Goldberg (2014) showed that, in fact, Skip-Gram factorizes a shifted PMI matrix in the way that:

$$M^{hidden} \cdot M^{out} + log(k) = M^{pmi} \tag{27}$$

where k is a constant.

Glove (Pennington, Socher, and Manning 2014) is a prediction based model that explicitly uses M^{pmi}. Its cost function (see Section 2.3.1) J^{GLOVE} can be defined as:

$$J^{GLOVE}(\{w; b\}) = \sum_{t_i \in T} \sum_{c_j \in C} M^B_{ij}(w_i^T w_j + b_i + b_j - log\,\#(t_i, c_j))^2 \tag{28}$$

where M^B is a fixed matrix of biases and $|T| = |C|$, i.e. the number of possible terms equals the number of possible contexts. The later is feasible because Glove uses word co-occurrences to define C as described in equation (25), therefore contexts can be identified by terms. Obviously, the same holds for Skip-Gram.

If we recap the definition of M^{pmi} from equation (20), $M^{pmi}_{ij} = log\,\#(t_i, c_j) - b_i^{(T)} - b_j^{(C)}$, we notice that Glove's objective factorizes M^{pmi}:

$$(w^T w)_{ij} = log\,\#(t_i, c_j) - b_i - b_j \tag{29}$$

$$(w^T w)_{ij} + b_i - b_i^{(T)} + b_j - b_j^{(C)} = M^{pmi}_{ij} \tag{30}$$

Note, that, in contrast to Skip-Gram, the approach of Glove implements full symmetry with regard to T and C. By doing so, all information is accumulated in one mapping, context specific information is not outsourced like it does the default Skip-Gram model. Consider the Glove term embedding f^{GLOVE}_T with respect to its objective J^{GLOVE} (equation 28):

$$f^{GLOVE}_T(t_i) = f^{GLOVE}_C(c_i) = w \cdot e_i + b_i \tag{31}$$

and compare it with the Skip-Gram embedding function $f_T^{SG}(t_i) = M^{hidden} \cdot e_i$ with respect to its prediction function (equation 26) that is used to train the Skip-Gram model:

$$f^{SG}(t_i) = softmax(M^{out} \cdot M^{hidden} \cdot e_i) \tag{32}$$

Glove combines all learned information in w and b and use them to project into the embedding space, whereas Skip-Gram learns in addition to M^{hidden} the matrix M^{out} to project from the embedding space back into the space of terms. However, this ostensible drawback of Skip-Gram can be circumvented by using $(f_T^{SG} + f_C^{SG})$ or $[f_T^{SG}, f_C^{SG}]$ as term embedding, where $f_C^{SG}(c_i) = (M^{out})^T \cdot e_i$.

2.4.4 Composition Models

Compositional distributional models of meaning or Compositional Distributional Semantics Models (CDSMs) (Clark, Coecke, and Sadrzadeh 2008; Grefenstette and Sadrzadeh 2011) intend to produce document embeddings $\hat{d}$ from a sequence of term embeddings $\hat{s} = [\hat{t}_1, ..., \hat{t}_\kappa]$ with $\forall \hat{t} \in \hat{s} : \hat{t} = f_T(t), t \in T, \hat{t} \subseteq \mathbb{R}^{\hat{m}}$. They can be expressed recursively by:

$$\begin{aligned} f_{CM}(\hat{s}) &= f_N(f_{Rec}(\hat{s}; h_I); |\hat{s}|) \quad \text{and} \\ f_{Rec}(\hat{s}; h) &= \begin{cases} h & \text{if } |\hat{s}| = 0 \\ f_{Rec}([\hat{t}_2, ..., \hat{t}_\kappa]; f_R(f_M(\hat{t}_1); h)) & \text{else} \end{cases} \end{aligned} \tag{33}$$

where $f_M : \mathbb{R}^{\hat{m}} \mapsto \mathbb{R}^{\mathring{m}}$ is the term embedding *mapping function*, $f_R : (\mathbb{R}^{\mathring{m}}, \mathbb{R}^k) \mapsto \mathbb{R}^k$ is the internal *reduction function*, $f_N : (\mathbb{R}^k, \mathbb{N}) \mapsto \mathbb{R}^{\hat{n}}$ is a *normalization function* and $h_I \in \mathbb{R}^k$ is the initial internal state.

In the following we distinguish *order unaware* composition models from *order aware* models. Both types commonly use the identity function or $(htan \circ l^{FC})$ as mapping function f_M. There are also hybrid approaches based on n-grams and Convolutional Neural Networks[10], but these are out of scope for this work.

Order unaware composition

These models are known as Bag-of-Words models. They can be defined by constraining the reduction function f_R to be associative and commutative. Common instances are *summation* with:

$$f_{R_{sum}}(\mathring{t}; h) = h + \mathring{t} \quad \text{and} \quad h_{I_{sum}} = [0, ..., 0]^T \in \mathbb{R}^k, k = \mathring{m} \tag{34}$$

[10]They fit into the definition of f_{CM} by relaxing f_M, which is, in fact, a convolution kernel of size one, to $f_{M_{cnn}} : (\mathbb{R}^{\hat{m}}, ..., \mathbb{R}^{\hat{m}}) \mapsto \mathbb{R}^{\mathring{m}}$ and using the *max* function for $f_{R_{cnn}}$ to implement max pooling, for instance.

or *averaging* with:

$$f_{R_{avg}} = f_{R_{sum}}, \quad h_{I_{avg}} = h_{I_{sum}} \quad \text{and} \quad f_{N_{avg}}(\dot{d};\kappa) = \frac{1}{\kappa} \cdot \dot{d} \tag{35}$$

Order aware composition

These models take the ordering of the input tokens into account. Commonly they build upon Recurrent Neural Networks (RNNs). So, we can define the reduction function as:

$$f_{R_{rnn}} = g^{RNN} \tag{36}$$

where g^{RNN} is an RNN cell function (see equation (8), assuming θ is fixed) like g^{LSTM} (see equation (10)), for instance. As described in Section 2.2.3, using this kind of non-associative composition function enables processing conditioned by previously seen input. In other words, RNNs leverage contextualized processing of individual tokens.

2.4.5 Similarity measures

A similarity measure for embeddings $\hat{d}$ is a symmetric function $f_S : (\mathbb{R}^{\hat{n}}, \mathbb{R}^{\hat{n}}) \mapsto [0,1] \subset \mathbb{R}$ with:

$$f_S(\hat{d}_i, \hat{d}_i) = 1 \quad \text{and} \quad f_S(\hat{d}_i, \hat{d}_j) < 1 \Leftrightarrow i \neq j \tag{37}$$

The *cosine similarity* is a prominent similarity measure, that is defined by:

$$f_{S_{cos}}(\hat{d}_i, \hat{d}_j) = \frac{\sum_{k=1}^{\hat{n}} \hat{d}_{i_k} \hat{d}_{j_k}}{\|\hat{d}_i\|_2 \|\hat{d}_j\|_2} = \left(\frac{\hat{d}_i}{\|\hat{d}_i\|_2}\right)^T \cdot \frac{\hat{d}_j}{\|\hat{d}_j\|_2} \tag{38}$$

where $\|v\|_2 = \sqrt{\sum_{k=1}^{|v|}(v_k)^2}$ is the L_2 (euclidean) norm of v. As equation (38) shows this measure explicitly normalizes its input. This can be an advantage because it is independent of the *significance* of the embeddings which is, as Schakel and Wilson (2015) state, encoded by their norm.

An *euclidean* distance based similarity measure that relies directly on the euclidean norm can be constructed as:

$$f_{S_{eucl}}(\hat{d}_i, \hat{d}_j) = exp(-\|\hat{d}_i - \hat{d}_j\|_2) \tag{39}$$

Similarly, the *manhattan/taxicab* distance can be used as similarity measure as proposed by Mueller and Thyagarajan (2016):

$$f_{S_{manh}}(\hat{d}_i, \hat{d}_j) = exp(-\|\hat{d}_i - \hat{d}_j\|_1) \tag{40}$$

where $\|v\|_1 = \sum_{l=1}^{|v|} |v_l|$ is the L_1 norm of v. Compared with the other measures, $f_{S_{manh}}$ has the lowest calculation costs.

2.5 Linguistic Dependency Types

In the following, we present a very brief introduction into the theory of linguistic dependency grammar and typed dependency relations that was mainly distilled from Jurafsky and Martin (2014).

2.5.1 Dependency Grammar

Dependency Grammar (DG) is a class of syntactic theories rooted in the work of Lucien Tesnière (Tesnière 1976). Its formalism describes the structure of sentences by linking individual words in a sentence with each other by directed, binary relations forming a tree commonly rooted by the finite verb. Any word has zero to multiple *dependents* linked by *dependency* relation instances. With respect to its dependents, a word is referred to as *head*. Every word has maximal one head. That head governs the morpho-syntactical features of the dependents. A head and its dependents mark a linguistic unit in the hierarchical structure. Figure 2 shows an example dependency syntax tree.

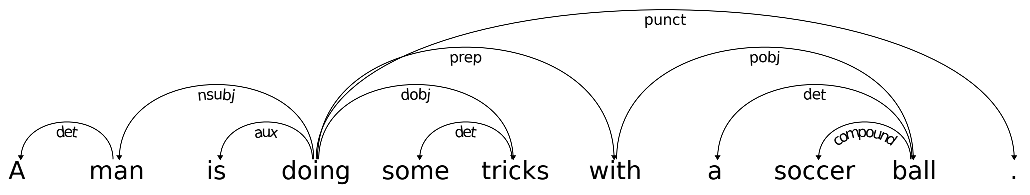


Figure 2: Dependency parse tree with typed relations. Arcs point from heads to their dependents. They are labeled with dependency types.

2.5.2 Dependency Types

In addition to structural information given by the dependence relation, dependency grammars classify the kinds of dependencies in terms of the grammatical role the dependent plays with respect to its head. For instance, *subject*, *direct object* or *indirect object* are commonly used and relate to the main finite verb. Because they link phrases of clauses, they are called *clausal argument relations*. To classify all types of dependencies found in dependence structures, the DG formalism expands the set of grammatical roles further to types of *nominal modifier relations*, *conjunctions* and many others. Languages vary in the set of dependency types and the means of expressing them (e.g. morphological marker, word order, function words). Nevertheless, efforts are underway to standardize the landscape of considered types of dependency relations. One approach that recently gained attention is the Universal Dependencies project (Nivre et al. 2016). It defines a common core of 40 grammatical relations while being flexible enough to incorporate language specific types, if needed. The project currently provides resources annotated with dependency structure (treebanks) for 60 different languages.

The English language is comparably restricted in its word order and has weak morphology. As Jurafsky and Martin (2014) states, in this case, the dependency type strongly correlates with the position of the individual words.

2.5.3 Dependency Parsing

Dependency parsing describes the process of constructing typed dependency structures for a given textual input. This can be done automatically by programs called dependency parsers.

State of the art systems achieve labeled attachment scores (LAS) and labeling scores (LS) of 89.0% / 93.3% (Honnibal and Johnson 2015)[11] or 90.7% / 94.7% (Choi, Tetreault, and Stent 2015) at the English OntoNotes corpus (Weischedel et al. 2011). The authors of the corpus name an inter-annotator agreement of 98.5% F1 for syntactical annotations on a sampled subset.

[11]This parser is part of the spaCy (`https://spacy.io/`) NLP framework.

3 Related work

Semantic relatedness prediction on sentence level was chosen as first task of SemEval 2014[12]. SemEval is an annual challenge intended to explore the ways natural language represents meaning. Like this work, the SemEval 2014 task utilizes the SICK (Marelli, Menini, et al. 2014) corpus to evaluate performance of composition models. Marelli, Bentivogli, et al. (2014) summarize the various approaches of submitted models. The majority of participants exploits compositionality features on different degrees of granularity (phrase or full sentence) plus other hand crafted features (e.g., word overlap and similarity, syntactical, alignments, topic modeling). Different learning approaches like Support Vector Machines (Cortes and Vapnik 1995) and kernel based methods, Random Forests (Breiman 2001) or ensembles (Opitz and Maclin 1999) were applied. The best performing system (J. Zhao, Zhu, and Lan 2014) exploits features related to sentence length, token and dependency type overlap, several distances (e.g., jaccard, cosine, manhatten) applied to TF-IDF and latent semantic representations, n-gram data at token and character levels, co-occurrence statistics and summed LSA data obtained from other corpora, among others, and applies an ensemble of machine learning methods. The system achieving the third place (Bjerva et al. 2014) builds on Formal Semantics and logical inference. In addition, it uses summed word2vec (Mikolov, Sutskever, et al. 2013) embeddings to produce sentence representations and calculates the cosine similarity between pairs, we reuse this idea (Section 4.1). Table 1 shows the top-5 scoring submissions and their relatedness prediction performance (Marelli, Bentivogli, et al. 2014).

	r	MSE
ECNU	0.828	0.325
StanfordNLP	0.827	0.323
The Meaning Factory	0.827	0.322
UNAL-NLP	0.804	0.359
Illinois-LH	0.799	0.369

Table 1: The top-5 submissions for the SemEval-2014 task 1 semantic relatedness prediction challenge. The performance was measured in Pearson correlation (r) and Mean Squared Error (MSE).

More recently, approaches that focus on Artificial Neural Networks gained increasing at-

[12]See http://alt.qcri.org/semeval2014/task1 for further information.

tention, leveraged by the rise of semantical rich token embeddings (Mikolov, Sutskever, et al. 2013; Pennington, Socher, and Manning 2014; Wieting, Bansal, Gimpel, Livescu, and Roth 2015). The Deep Averaging Network (DAN) (Iyyer et al. 2015) discards any syntactic structure, even ordering information, by averaging all token embeddings of a sentence. Afterwards, a deep FNN is applied. This model performs quite well on sentiment classification tasks, comparable to state of the art systems that rely on linear (Kim 2014) or hierarchical structure (Tai, Socher, and Manning 2015). Several other systems (Kim 2014; Kalchbrenner, Grefenstette, and Blunsom 2014; Hu et al. 2014; Yin and Schütze 2015; He, Gimpel, and Lin 2015) exploit Convolutional Neural Networks (CNNs) (LeCun and Bengio 1995) to compose pretrained token embeddings. He, Gimpel, and Lin (2015) uses convolutional filters to incorporate information of different granularities and directions, i.e. along token n-grams or dimensions of token embeddings, to produce sentence embeddings that are feed into a similarity measurement layer which compares selected subsets of sentence embeddings. They use the identical structure and set of weights to embed both sentences, thus their approach follows a *siamese* structure (Bromley et al. 1994).

Another line of research uses RNN architectures, especially Long Short-Term Memorys (LSTMs) (Wieting, Bansal, Gimpel, and Livescu 2015; Liu et al. 2015; Mueller and Thyagarajan 2016). As this work, Wieting, Bansal, Gimpel, and Livescu (2015) compares averaging of embeddings and LSTM based composition. The authors conclude, that simple averaging outperforms the LSTM model on the SICK dataset. In contrast to our system, they use embeddings trained on the PPDB (Ganitkevitch, Van Durme, and Callison-Burch 2013) dataset that is a very large paraphrase corpus containing pairs of short text spans. Mueller and Thyagarajan (2016) applies a simple LSTM in a siamese architecture and uses manhattan metric as similarity measure. In this means, their approach strongly resembles our LSTM setting (Section 4.1). Despite its simplicity, their model achieves high performance in the SICK relatedness prediction task. However, the system benefits strongly (+0.04 Pearson's r) from synonym augmentation that the authors used to double the size of the SICK dataset. The Skip-Thoughts (Kiros et al. 2015) model follows an encoder-decoder architecture to learn sequence embeddings that perform state of the art at the SICK relatedness task. The system is trained as generative language model, i.e., to produce a meaningful sentence complement given a sentence predecessor. As encoding and decoding units the model uses Gated Recurrent Units (GRUs) (Cho et al. 2014). GRUs are RNNs that are less complex then LSTMs despite performing comparable well.

Expanding the concept of RNNs to tree structures, several Recursive Neural Network (RecNN)[13] (Goller and Kuchler 1996) based architectures for semantic aware composition were proposed which rely on pre-calculated parse structure (Socher, Huang, et al. 2011; Socher, Huval, et al. 2012; Socher, Perelygin, et al. 2013; Irsoy and Cardie 2014; Tai, Socher, and Manning 2015; Wieting, Bansal, Gimpel, Livescu, and Roth 2015) or construct internal composition hierarchies on their own (H. Zhao, Lu, and Poupart 2015; X. Chen et al. 2015).

Another relevant work is the Paragraph Vector or doc2vec model (Le and Mikolov 2014). It builds directly on word2vec (Mikolov, Sutskever, et al. 2013) to learn distributed representations of individual, arbitrarily long token sequences (paragraphs) while training word embeddings. The approach lends from the word2vec continuous bag of words (CBOW) model, but modifies its optimization procedure as follows. A *virtual* word, that is shared along one paragraph, is added to every bag-of-word (BoW) context instance that belongs to this paragraph. Apart from that the training stays the same, i.e. using the average of all word embeddings related to one context of a target word, the system is optimized to predict this target. As soon as the model has converged, the vectors associated to the virtual words represent the respective paraphrase embedding. In fact, doc2vec is not a pure composition model as it requires to train the paragraph vectors along with the word embeddings. But it produces sequence embeddings that perform quite well for several semantic tasks (Lau and Baldwin 2016).

[13]RecNNs generalize RNNs in the way that their unfolded computation graph do not have to follow a linear chain, but rather is capable to model an arbitrary tree structure by allowing more than one predecessor.

4 Model architecture and training

This chapter describes the architecture of the neural models used to calculate a similarity score, i.e. the semantic relatedness, for a given sentence tuple and how these models are trained.

As we intend to analyze the impact of dependency type information and order aware processing, we introduce two boolean parameters, `dependency available` and `order aware`, to indicate if the respective information is exploited. After introducing the general model architecture, we explain how these parameters are realized. Then, we describe the baseline model and finally name implementation and training details.

4.1 Model architecture

Both models consist of a sentence embedding unit and a similarity function. The embedding unit translates a sequence of tokens into a single embedding vector and is applied to the two input sentences using identical weights, thus our systems follow a siamese structure (Bromley et al. 1994). They calculate the sentence embedding by gathering the individual embeddings for the contained tokens, eventually enhancing them with additional information, and applying a composition function. The two resulting embeddings are fed into the similarity function that produces the similarity score.

Figure 3 shows this architecture. In the following, we describe the respective modules.

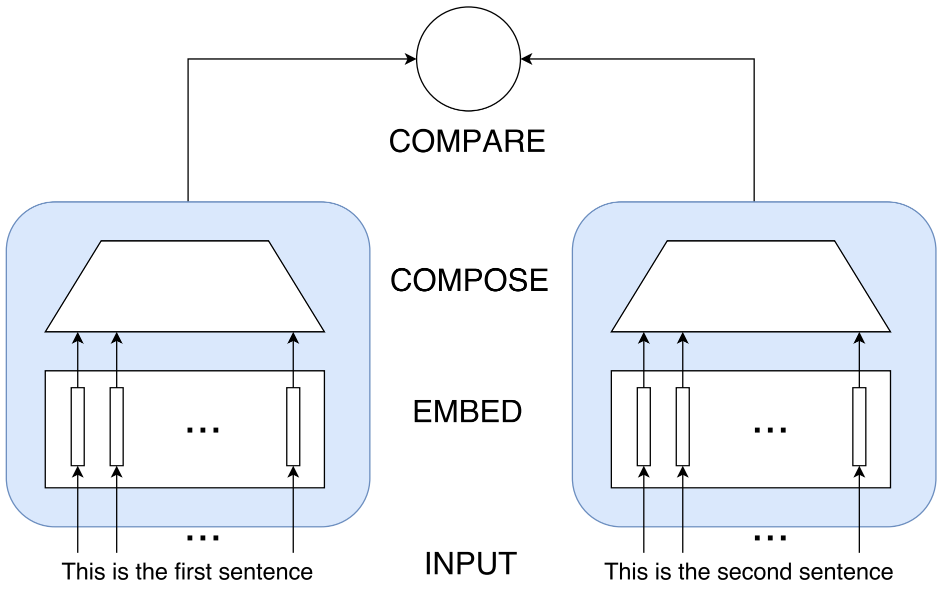


Figure 3: General architecture. The functionality marked by the blue boxes is identical for both input sentences, i.e. they share structure and weights.

4.1.1 Token Embeddings

We use 300 dimensional GloVe vectors (Pennington, Socher, and Manning 2014) pretrained on the Common Crawl[14] corpus to embed individual word tokens as they show good performance on several NLP tasks and perform comparable to its major competitor, Word2Vec (Levy, Goldberg, and Dagan 2015; Naili, Chaibi, and Ben Ghezala 2017). To evaluate the influence of dependency types, we concatenate the embedding with the one-hot encoded dependency type (Section 4.3). The dependency type tag set is based on the Universal Dependencies project (Nivre et al. 2016) and includes 40 different types, resulting in 340 dimensional vectors as input for the composition function. When disabling this feature, we set these 40 entries to zero. The token embeddings are not optimized during training of our models.

4.1.2 Composition function

We compare two composition functions: (1) averaging (AVG) and (2) Long Short-Term Memory (LSTM). In the AVG case, we apply one fully connected layer (FC) with *tanh* as activation function to every token embedding. Then, averaging the resulting vectors gives the sentence embedding. In the LSTM setting, the token embeddings are fed directly into a LSTM layer whose final state output is used as sentence embedding.

To achieve comparability between the two composition functions, we control the amount of effectively trainable parameters in each of them. Depending on whether dependency type information is used, we use different sizes for the *tanh* FC in the AVG case and for the inner state of the LSTM to fix the amount of effective parameters. Table 2 shows the exact numbers and sizes of the elements. The sentence representation dimension equals to the composition element (output) size.

	AVG (FC size)	LSTM (state size)
w/ dependency types	110840 (326)	111248 (68)
w/o dependency types	111000 (370)	111000 (74)

Table 2: Numbers of trainable parameters and (output) size of composition function elements

[14] see http://commoncrawl.org

4.1.3 Similarity calculation

We use the cosine similarity to calculate the similarity score between the two sentence embeddings as described in section 2.4.5.

4.2 Baseline model

As baseline we calculate TF-IDF based similarity scores in the following manner. We parse and lemmatize each sentence and filter for verbs, nouns and adjectives according to POS tags. Then, we embed each sentence with TF-IDF as described in section 2.4.2 and apply cosine measure.

4.3 Implementation and Training

The model is implemented with the TensorFlow framework (Abadi et al. 2016). TensorFlow allows to define and to execute arbitrary dataflow graphs efficiently on different devices as CPUs or GPUs. For tokenization, dependency parsing and POS-tagging we use spaCy[15] which is a fast and still accurate[16] NLP framework (labeling score of 93.3% for dependency parsing, see Section 2.5.3).

We train the model via batched back-propagation using MSE loss function as described in section 2.3.1 and apply ADAM (Kingma and Ba 2014) as optimizer. We use a 4:1:5 train/dev/test split. The training is terminated when the score on the development data set does not increase anymore regarding a smoothing window covering the last 25 epochs.

[15] see `https://spacy.io/`

[16] see Choi, Tetreault, and Stent (2015) for a comparative study

5 Experiments and Evaluation

We conduct several experiments to evaluate the influence of order awareness and availability of dependency type information to the similarity prediction performance. We test all combinations of the boolean parameters `dependency available` and `order aware` and the TF-IDF baseline. In the following, we describe the dataset and the hyperparameter setting. Finally, we present the results including a comparative analysis of errors.

5.1 Dataset

We use the SICK corpus[17] (Marelli, Menini, et al. 2014) for model training and evaluation. The corpus consists of about approximately 10.000 pairs of English sentences based on the 8K ImageFlickr data set[18] (Hodosh, Young, and Hockenmaier 2013) and the SemEval 2012 STS MSR-Video Description data set[19] (Agirre et al. 2012) that contain sentences describing the same picture or video. These sentences were linguistically normalized; i.e. Named Entities and complex verb constructions are replaced and subordinates are turned into coordinates. Pairs were manually labeled by 10 annotators with one to five point relatedness ratings that are averaged to produce a relatedness score. We rescaled the scores into the interval $[0.0, 1.0]$ to let them fit into our definition of a similarity measure. Table 3 shows some examples of the dataset. The mean of the score for both train and test set is approximately 0.63. The sentences contain 2408 different token types and have an average length of 9.6 words.

We chose the SICK corpus because it was used in the SemEval 2014 task 1[20], thus it is widely studied and various reference systems exist. Furthermore, as stated by Marelli, Bentivogli, et al. (2014), this corpus was created to evaluate semantical aware composition without relying on too many external tools and resources (e.g., named-entity recognizers, gazetteers, ontologies).

5.2 Training and Hyperparameters

We train the model in batches of 100 examples. The ADAM optimizer is initialized with a learning rate of 0.003. We apply gradient clipping by global norm as specified in Pascanu, Mikolov, and Bengio (2012) with a threshold of 5.0 and use dropout (Srivastava et al. 2014) with a keep probability of 0.8 at the LSTM and the FC in the AVG case to prevent from

[17] http://alt.qcri.org/semeval2014/task1/index.php?id=data-and-tools

[18] http://nlp.cs.illinois.edu/HockenmaierGroup/data.html

[19] https://www.cs.york.ac.uk/semeval-2012/task6/index.php%3Fid=data.html

[20] http://alt.qcri.org/semeval2014/task1/

sentence A	sentence B	score
A man in a shirt dyed purple is looking at a man in a black shirt who is doing a funny face	A man in a shirt dyed purple is looking at a man in a black shirt who is doing a face which looks funny	1.0
A group of kids is playing in a yard and an old man is standing in the background	A group of boys in a yard is playing and a man is standing in the background	0.875
A brown dog is attacking another animal in front of the man in pants	Two dogs are wrestling and hugging	0.55
A woman is chopping an onion	A woman is washing her feet	0.0

Table 3: Example sentence pairs from SICK corpus with rescaled relatedness score ranging from 0.0 (not related) to 1.0 (maximal related).

overfitting. We used a grid search at the train/dev set to find this parameter configuration. We applied 5-fold cross validation with regard to the train/dev split for every setting and repeated each experiment 10 times to achieve robust results.

5.3 Results

We measured performance using the Pearson correlation coefficient and MSE between the predictions and gold scores. The mean Pearson correlation of all 200 runs[21] is about 0.839 with a standard deviation (STD) of 0.002 and the mean MSE is 0.024 with STD of 0.004. The TF-IDF baseline produces a Pearson score of 0.619 and a MSE of 0.082. Thus, the regarded models achieve a mean MSE performance gain of +6.4 pph (parts per hundred) with respect to the baseline. Note, that the authors of the SICK corpus name an inter-annotator agreement of 0.036 in means of variance.[22]

The performance of the Order Aware model (MSE: 0.0206; Pearson's r: 0.838) lies in the range of the best system submitted for SemEval-2014 relatedness prediction task (MSE: 0.020, Pearson's r: 0.828), but stays below state of the art. Recent systems achieve MSE/Pearson's r scores of 0.016/0.87 (He, Gimpel, and Lin 2015)(CNN) or 0.014/0.88 (Mueller and Thyagarajan 2016)(LSTM) without exploiting dependency information (Section 3). We list the

[21]4 settings $\times$ 5-fold cross validation $\times$ 10 repetitions

[22]Precisely, the authors present a standard deviation of 0.76. Rescaling from the original scoring range $[1, 5]$ into the range $[0, 1]$ used in this work leads to the mentioned variance value.

individual performances of the models examined in this work averaged over identical parameter settings in Table 4. Figure 4 displays the MSE and Pearson's r results for all parameter combinations as box plots[23]. Note, that the y-axis is inverted for all box plots visualizing MSE scores to show better performing scores above worse ones and to provide comparability with plots of Pearson scores.

`dependency available`	`order aware`	mse	pearson
NO	NO	0.0284	0.8382
NO	YES	0.0206	0.8375
YES	NO	0.0274	0.8413
YES	YES	0.0205	0.8383
TFIDF		0.0823	0.6189

Table 4: MSE and Pearson scores aggregated by setting.

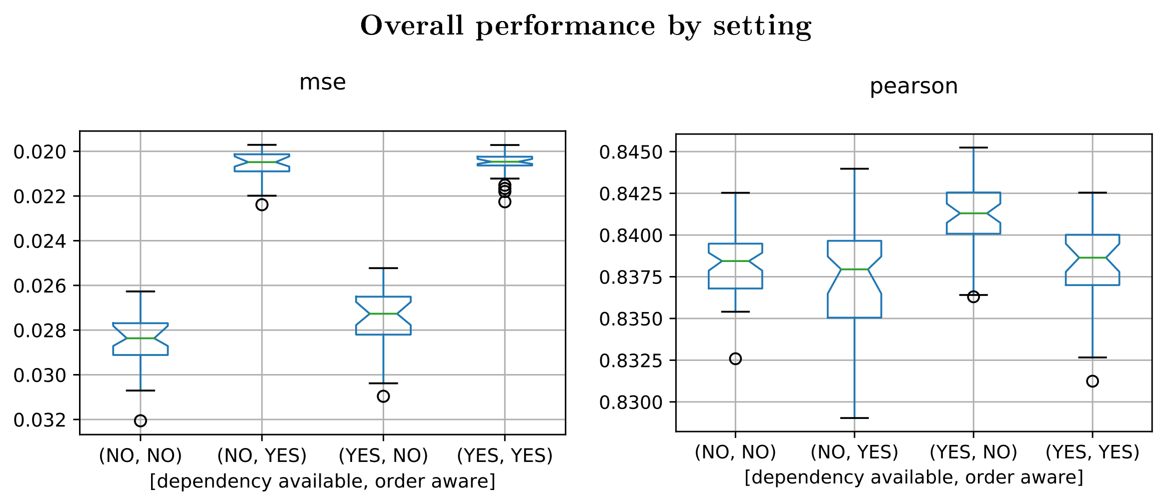


Figure 4: MSE (inverted y-axis) and Pearson's r for the different settings.

Adding **dependency types** improves on average MSE by 0.05% and Pearson's r increases by 0.24 pph, whereby the former is not significant (p=0.336), but the later is ($p < 0.0001$). Enabling the feature **order aware** results in a significant overall performance gain of 0.75 pph in means of MSE and a performance decrease of -0.21 pph with regard to Pearson's r ($p < 0.0001$ for two-tailed t-test in both cases). Table 5 shows the specific scores.

For further study these results that seem to contradict with respect to the applied measure,

[23]The boxes of all box plots show the interquartile range (IQR), whiskers mark the $[1.5 \times \text{IQR}, 3 \times \text{IQR}]$ range and notches the $10000\times$ bootstrapped confidence intervals.

parameter	enabled	mse		pearson	
dependency type	NO	0.0245		0.8378	
dependency type	YES	0.0240	+0.05 pph*	0.8398	+0.24 pph
order aware	NO	0.0279		0.8397	
order aware	YES	0.0206	+0.75 pph	0.8379	−0.21 pph

Table 5: Scores aggregated by individual parameter assignment and improvement when enabling the feature. The improvement marked with (*) is not significant.

we took a closer look at the deviations of the predicted relatedness scores from the gold scores. Figure 5 shows the deviations per setting ordered by gold score. It suggests that the averaging model (`order aware` = NO) overestimates the relatedness score. Indeed, the bias for all predictions in the averaging case (+8.5 pph) is significantly ($p < 0.0001$) higher when compared with non-averaging (+1.9 pph). Because such a bias distorts the Pearson's r, and MSE was used as cost function while training, we focus at the MSE measure for the rest of this work.

5.3.1 Relation of Order Awareness and Dependency Types

The specific performance in the examined settings suggests that the parameters `dependency available` and `order aware` are related. Figure 6 illustrates this by presenting the conditional impact of the parameters. The impact of `dependency available` seems to vary depending on activation of parameter `order aware`, whereas the opposite does not hold (see Table 4 for the actual values). This observation leads to the following question: What kind of useful information with respect to semantical awareness is encoded in dependency types? Precisely, we assume: Dependency types encode local context information that is useful to produce semantic aware compositional representations. The rationale is as follows. Order Awareness as implemented within LSTMs enable locally contextualized processing (see Section 2.4.4). Thus, these models encode local context. As explained in section 2.1, semantic awareness can be measured using the relatedness prediction task. Since enabling the parameter `order aware` significantly increases the prediction performance by +0.8 pph ($p < 0.0001$) the local context information seems to be important to determine the meaning of a word in a specific textual utterance and to handle semantic aware composition. Enabling the parameter `dependency available` significantly improves the model performance by +0.1 pph ($p < 0.001$) in the case of disabled order awareness, proving that dependency types indeed encode **useful** information.

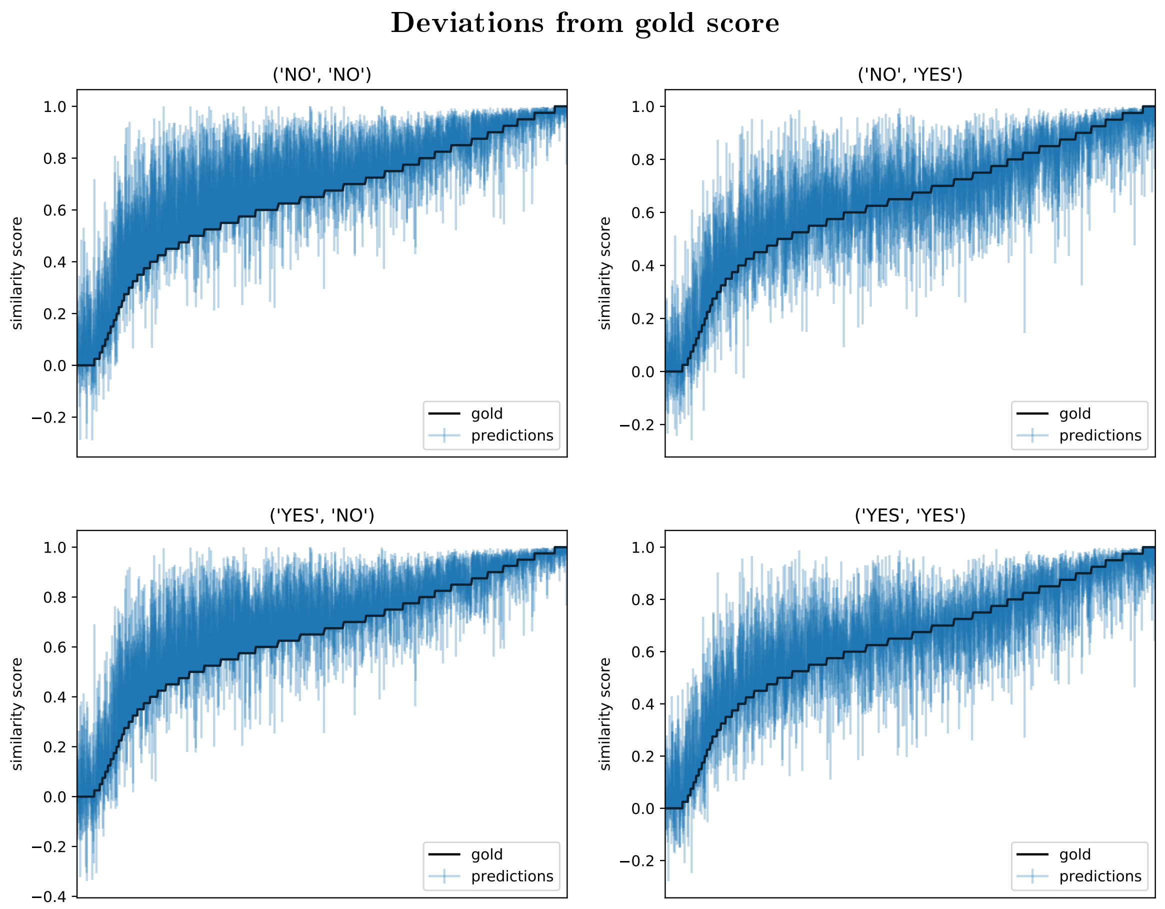


Figure 5: Deviations of the predictions from gold similarity scores by setting (<`dependency available`>, <`order aware`>). The averaging models (`order aware` = NO; plots at the left) clearly overestimate.

Since the order aware model does not perform significantly better with dependency type data ($t = 0.50$) it seems that this data does not encode extra useful information for order aware models. This suggests that dependency types indeed encode **local context**.

5.3.2 Value of Dependency Types

In order to further investigate the value of dependency type data for semantic aware composition we have taken a closer look at the sentence pairs in which only adding this data outperforms the setting in which parameter `order aware` is enabled exclusively. Table 6 lists the top 10 of these data points. Trying to examine the reasons behind cases of superior performance reveals two main classes of constructions (see column *remarks* of Table 6). First, half of the top 10 sentence pairs contains passive constructions. Second, three sentence pairs are related to negation.

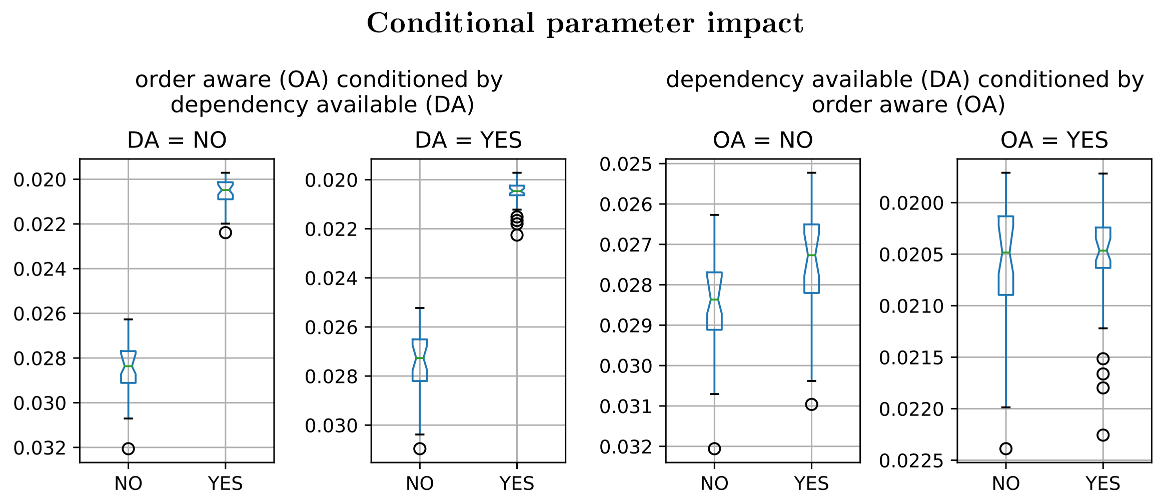


Figure 6: Conditional parameter impact measured with MSE.

As it turns out, dependency type data seems to be beneficial to handle composition of tokens involved in passive constructions. We filtered the test data set for dependency types indicating these constructions (*auxpass*: passive auxiliary; *nsubjpass*: passive nominal subject; *csubjpass*: passive clausal subject) and calculated MSE scores[24]. Table 7 presents the resulting scores and subset sizes. Figure 7a visualizes the results. The Dependency Available setting ([YES, NO]) significantly ($p < 0.0001$) outperforms the Order Aware setting ([NO, YES]) in the presence of passive constructions. Furthermore, negation seems to be hard in general. Similar to the passive check, we selected a subset of the test dataset, but filtered by dependency type *neg* and words like "no" or "nobody". For all four settings, the performance dropped significantly when switching from the subset without negation to the negation subset. Despite initial thoughts, negation favors Order Awareness (e.g., performance drop of -0.2 pph for [NO, YES] vs. -0.5 pph for [YES, NO]) as visualized in Figure 7b. Again, Table 7 displays the specific evaluation scores and, in addition, relative performance differences with respect to the considered data split.

Obviously, the benefit of dependency types decreases with increasing word overlap. Correlating the absolute differences of prediction errors of settings [NO, NO] and [YES, NO] with the Jaccard similarity $f_{S_{\text{Jacc}}}$[25] of the respective sentences results in a Pearson's r of -0.46 indicating a moderate negative correlation.

[24]We used the relatedness scores predicted by the best model for each setting with respect to train/test split and run.

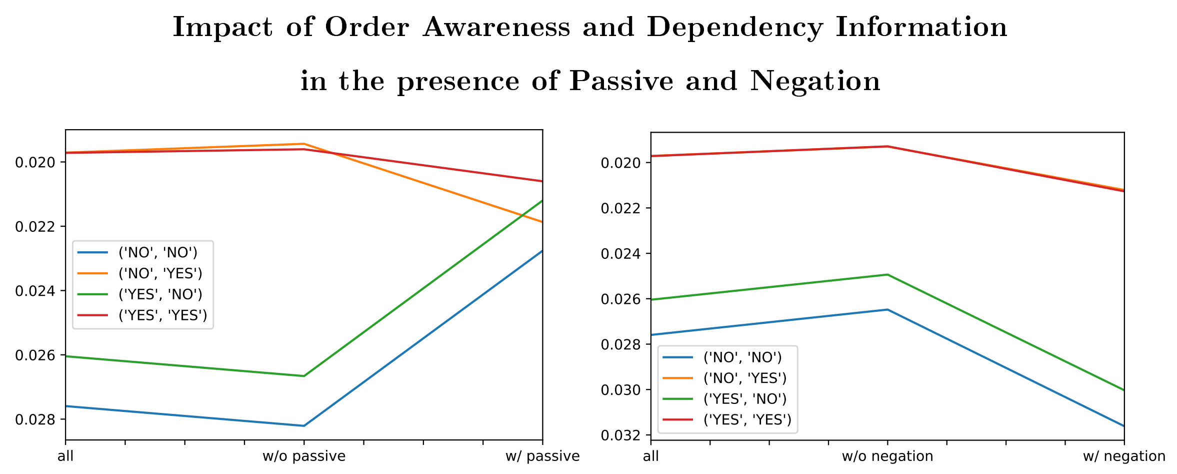
[25]$f_{S_{\text{Jacc}}}(s_1, s_2) := \frac{\#(\text{set of tokens in } s_1 \texttt{ AND } s_2)}{\#(\text{set of tokens in } s_1 \texttt{ OR } s_2)}$

sentence A	sentence B	errors	remarks
The patient is being helped by the doctor	The doctor is helping the patient	−.0; −.5 (1.0)	passive
The panda bear is not lying on the logs	A cute panda is lying down	−.2; −.7 (0.8)	negation
A man and a woman are sitting comfortably on the bench	Two large persons are sitting on a park bench and they have a bottle of soda between them	−.1; −.5 (0.7)	appended "noise"
A man is not playing a guitar	A man is playing a keyboard	−.1; −.4 (0.7)	negation
A dog is standing with its two front paws on a rock in a field	A rock is being climbed by the black dog	−.1; −.4 (0.6)	passive
Four middle eastern children, three girls and one boy, are climbing on the grotto with a pink interior	The grotto with a pink interior is being climbed by four middle eastern children, three girls and one boy	−.0; −.3 (1.0)	passive
The flute is being played by one man	A man is playing the guitar loudly	−.0; −.3 (0.6)	passive
Nobody is dangling from straps and kicking at each other	A blonde girl is hanging by gymnastic ropes	−.1; −.4 (0.4)	negation
A child is making a snow ball	A snow ball is being made by a child	−.0; −.3 (1.0)	passive
A black cat and a small white and black cat are looking up at a kitchen countertop	A large dog and a small dog are standing next to the kitchen counter and are investigating	−.0; −.3 (0.4)	

Table 6: The top 10 sentence pairs where enabling `dependency available` (setting: [YES, NO]) outperforms enabling `order aware` (setting: [NO, YES]). Rounded deviations from gold scores for both settings ([YES, NO]; [NO, YES]) are presented as *errors* with the actual gold score (rounded) in brackets. The *remarks* column names sources of potentially beneficial effects in favor of the Dependency Available setting.

test data	[NO, NO]		[NO, YES]		[YES, NO]		[YES, YES]		count
all	0.028		0.020		0.026		0.020		4927
w/o passive	0.028		0.019		0.027		0.020		4371
w/ passive	0.023	+0.5%	0.022	−0.3%	0.021	+0.6%	0.021	−0.1%	556
w/o negation	0.026		0.019		0.025		0.019		3856
w/ negation	0.032	−0.6%	0.021	−0.2%	0.030	−0.5%	0.021	−0.2%	1071

Table 7: MSE for selected subsets of SICK test data measured within all settings ([<`dependency available`>, <`order aware`>]) and relative performance gains/drops (+/-) in pph (%). *count* represents the number of sentence pairs in the respective subset.



(a) The Dependency Available setting ([YES, NO]) outperforms the Order Aware setting ([NO, YES]) on passive constructions.

(b) The performance decreases for all settings in the presence of negation, but favors Order Awareness ([NO, YES] and [YES, YES]).

Figure 7: MSE for specific test data subsets

6 Discussion and Future Work

In line with Mueller and Thyagarajan (2016) and Iyyer et al. (2015), our results demonstrate that simple neural models perform quite well for semantic aware composition. Furthermore, we investigated the relation of dependency type data and order aware composition. In this chapter we discuss our findings.

As demonstrated in Chapter 5.3.1 locally contextualized processing of individual tokens does matter. That capability is achieved by order aware RNN models holding previously processed information in an internal state which is considered when analyzing the next token. By keeping the internal state reasonable in size, bottleneck effects induce localization. It suggests, that this kind of contextualization leverages performance by filtering relevant information similar to processes. But it does not seem to be like sequential processing is a requirement. In addition to several well performing CNN models, one might imagine to process every individual token in a bag of words manner, while incorporating a vector representation previously created from this bag of words, or in a n-grams fashion, eventually. This mechanism builds upon *attention* (Bahdanau, Cho, and Bengio 2014; Vaswani et al. 2017) and is called self-attention or intra-attention (Cheng, Dong, and Mirella Lapata 2016). It seems to perform well on semantic NLP exploiting very little ordering information (Parikh et al. 2016). In fact, that approach sets one or multiple context frames for each token which could be considered as ordering information, too. Further experiments taking some of these insights into account could be arranged by simply shuffling the sentence tokens and repeating our experiments. By doing so, the context is artificially enlarged to cover the hole sentence and one might investigate, if RNN models still outperform averaging of independently mapped embeddings. Likely this experimental setting requires to use Bi-LSTMs (Graves, Mohamed, and G. Hinton 2013) or similar RNN architectures, that provide information regarding all previous and following tokens at every position.

Even though all models examined in this work conceptually have access at comparison level[26] to all information included in the respective embeddings, intermediate sentence representations functioning as bottleneck constrain their outreach, which underpins the benefit of local filtering. To proof this idea, one could extend our models by adding eventually deep networks on top of the sentence embedding layer similar to Iyyer et al. (2015) and increase the dimensionality of sentence representations itself. Thus the performance advantage of the order aware models should decrease.

[26]i.e. similarity measure application, see Section 4.1

Furthermore, we argued that locally contextualized processing as exploited by order aware models can be achieved by adding dependency type information. As mentioned in 2.5.2, order information strongly correlates with dependency type data for English language. Our findings underpin this thesis. However, the performance is still behind the order aware case in general, but exploiting dependency types significantly outperforms applying order awareness in the presence of passive constructions. This insight leads to the question if there are other structural cases in which using dependency type data can outperform order aware processing. Passive is just one kind of *syntactic alternation*[27], among many others. Another example is the dative alternation in the following example[28]:

(1) a. Who gave $\underline{\text{that wonderful watch}}$ $\underline{\text{to you}}$?

b. Who gave $\underline{\text{you}}$ $\underline{\text{that wonderful watch}}$?

Syntactical alternating phrases express the same meaning[29], but alternate at least partly the way their arguments are expressed (e.g. by the alternation between a prepositional indirect-object construction in Example 1a and a double-object construction in Example 1b). One might expect that processing these constructions would benefit from dependency type data, too. In the field of corpus linguistics the dative alternation is heavily studied (Maria Lapata 1999; Bresnan and Nikitina 2003; Bresnan, Cueni, et al. 2007; Kendall, Bresnan, and van Herk 2011). It should be straight forward to lend extraction patterns and create respective SICK corpus subsets to evaluate if dependency type data is also superior to ordering information in this case. Additionally, one might look into languages with less restrictive word order, like Turkish or Latin.

This work is based on the SICK corpus. As it is artificially created from image descriptions, it probably narrows the space of semantic and syntactic phenomena observed in nature. Furthermore, using a manually scored corpus requires a conceptional (pre-)determination of *relatedness* potentially holding a bias. For instance, having a second look at the sentence tuple presented in Table 3:

(2) A woman is chopping an onion
A woman is washing her feet

This tuple is scored with a relatedness of 0.0. However, one might expect that the following tuple scores even lower:

[27]See (Levin 1993) for a comprehensive study.

[28]The example was taken from (Kendall, Bresnan, and van Herk 2011).

[29]That is true in a broader sense of meaning. Apparently, there is a subtle difference in means of focus.

(3) A woman is chopping an onion

A man is washing his feet

Of course, different segments of a sentence, like the verbal parts, can be more important to the total meaning then other. But denying the fact that the actor in Example 2 is identical, seems to be too restrictive. This could be an instance of priming (Weingarten et al. 2016), i.e., that reading one sentence influences the perception of a following one unconsciously. It leads to the question if relatedness expressed at the beginning of the sentences is systematically underrated in the SICK corpus.

We discarded the evaluation result in means of Pearson's r because it shows same strange behavior that we traced back to a deviation bias along the parameter `order aware`. To determine the origin of these circumstances, further investigations are necessary. Especially, the impact of switching to a loss function based on Pearson correlation should be examined.

References

Abadi, Martín et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning". In: *OSDI*.

Agirre, Eneko et al. (2012). "Semeval-2012 Task 6: A Pilot on Semantic Textual Similarity". In: *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pp. 385–393.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: arXiv: 1409.0473 [cs, stat]. URL: http://arxiv.org/abs/1409.0473 (visited on 01/28/2018).

Beltrami, Edward (1873). "Sulle Funzioni Bilineari". In: Giornale di Matematiche ad Uso degli Studenti Delle Universita 11, pp. 98–106.

Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *Journal of machine learning research* 3 (Feb), pp. 1137–1155.

Bjerva, Johannes et al. (2014). "The Meaning Factory: Formal Semantics for Recognizing Textual Entailment and Determining Semantic Similarity." In: *SemEval@ COLING*, pp. 642–646. URL: http://www.aclweb.org/anthology/S14-2114.

Boltuzic, Filip and Jan Šnajder (2015). "Identifying Prominent Arguments in Online Debates Using Semantic Textual Similarity". In: *Proceedings of the 2nd Workshop on Argumentation Mining*, pp. 110–115. URL: http://www.aclweb.org/anthology/W/W15/W15-05.pdf#page=122 (visited on 06/12/2017).

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

Bresnan, Joan, Anna Cueni, et al. (2007). "Predicting the Dative Alternation". In: *Cognitive foundations of interpretation*, pp. 69–94.

Bresnan, Joan and Tatiana Nikitina (2003). *On the Gradience of the Dative Alternation*.

Bromley, Jane et al. (1994). "Signature Verification Using a" Siamese" Time Delay Neural Network". In: *Advances in Neural Information Processing Systems*, pp. 737–744.

Budanitsky, Alexander and Graeme Hirst (2006). "Evaluating Wordnet-Based Measures of Lexical Semantic Relatedness". In: *Computational Linguistics* 32.1, pp. 13–47. URL: http://www.mitpressjournals.org/doi/abs/10.1162/coli.2006.32.1.13 (visited on 08/24/2017).

Chen, Xinchi et al. (2015). “Sentence Modeling with Gated Recursive Neural Network.” In: *EMNLP*, pp. 793–798. URL: https://www.aclweb.org/anthology/D/D15/D15-1092.pdf (visited on 10/04/2017).

Cheng, Jianpeng, Li Dong, and Mirella Lapata (2016). “Long Short-Term Memory-Networks for Machine Reading”. In: arXiv: 1601.06733 [cs]. URL: http://arxiv.org/abs/1601.06733 (visited on 01/28/2018).

Cho, Kyunghyun et al. (2014). “Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A Meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. URL: http://aclweb.org/anthology/D/D14/D14-1179.pdf.

Choi, Jinho D., Joel R. Tetreault, and Amanda Stent (2015). “It Depends: Dependency Parser Comparison Using A Web-Based Evaluation Tool.” In: *ACL (1)*, pp. 387–396. URL: http://www.aclweb.org/anthology/P15-1038.

Church, Kenneth Ward and Patrick Hanks (1990). “Word Association Norms, Mutual Information, and Lexicography”. In: *Computational linguistics* 16.1, pp. 22–29.

Clark, Stephen, Bob Coecke, and Mehrnoosh Sadrzadeh (2008). “A Compositional Distributional Model of Meaning”. In: *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*. Oxford, pp. 133–140.

Cortes, Corinna and Vladimir Vapnik (1995). “Support-Vector Networks”. In: *Machine learning* 20.3, pp. 273–297.

Deerwester, Scott et al. (1990). “Indexing by Latent Semantic Analysis”. In: *Journal of the American society for information science* 41.6, p. 391.

Epelbaum, Thomas (2017). “Deep Learning: Technical Introduction”. In: arXiv: 1709.01412 [cs, stat]. URL: http://arxiv.org/abs/1709.01412 (visited on 09/10/2017).

Ganitkevitch, Juri, Benjamin Van Durme, and Chris Callison-Burch (2013). “PPDB: The Paraphrase Database.” In: *HLT-NAACL*, pp. 758–764. URL: http://www.aclweb.org/anthology/N13-1#page=796 (visited on 06/12/2017).

Goller, C. and A. Kuchler (1996). “Learning Task-Dependent Distributed Representations by Backpropagation through Structure”. In: vol. 1. IEEE, pp. 347–352. ISBN: 978-0-7803-3210-2. DOI: 10.1109/ICNN.1996.548916. URL: http://ieeexplore.ieee.org/document/548916/ (visited on 01/22/2018).

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). “Speech Recognition with Deep Recurrent Neural Networks”. In: arXiv: 1303.5778 [cs]. URL: http://arxiv.org/abs/1303.5778 (visited on 01/28/2018).

Grefenstette, Edward and Mehrnoosh Sadrzadeh (2011). “Experimental Support for a Categorical Compositional Distributional Model of Meaning”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1394–1404.

Habernal, Ivan and Iryna Gurevych (2015). “Exploiting Debate Portals for Semi-Supervised Argumentation Mining in User-Generated Web Discourse.” In: *EMNLP*, pp. 2127–2137. URL: http://www.aclweb.org/anthology/D/D15/D15-1255.pdf (visited on 06/21/2017).

Harris, Zellig S. (1954). “Distributional Structure”. In: WORD 10 (2-3), pp. 146–162. ISSN: 0043-7956, 2373-5112. DOI: 10.1080/00437956.1954.11659520. URL: http://www.tandfonline.com/doi/full/10.1080/00437956.1954.11659520 (visited on 09/05/2017).

He, Hua, Kevin Gimpel, and Jimmy J. Lin (2015). “Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks.” In: *EMNLP*, pp. 1576–1586.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural computation* 9 8, pp. 1735–80.

Hodosh, Micah, Peter Young, and Julia Hockenmaier (2013). “Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics”. In: *Journal of Artificial Intelligence Research* 47, pp. 853–899.

Hofmann, Thomas (2001). “Unsupervised Learning by Probabilistic Latent Semantic Analysis”. In: *Machine learning* 42 (1-2), pp. 177–196.

Honnibal, Matthew and Mark Johnson (2015). “An Improved Non-Monotonic Transition System for Dependency Parsing”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1373–1378. URL: https://aclweb.org/anthology/D/D15/D15-1162.

Hopfield, J. J. (1982). “Neural Networks and Physical Systems with Emergent Collective Computational Abilities.” In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.79.8.2554. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.79.8.2554 (visited on 11/27/2017).

Hu, Baotian et al. (2014). “Convolutional Neural Network Architectures for Matching Natural Language Sentences”. In: *Advances in Neural Information Processing Systems*, pp. 2042–

2050. URL: http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences (visited on 10/04/2017).

Irsoy, Ozan and Claire Cardie (2014). "Deep Recursive Neural Networks for Compositionality in Language". In: *Advances in Neural Information Processing Systems*, pp. 2096–2104. URL: http://papers.nips.cc/paper/5551-deep-recursive-neural-networks-for-compositionality-in-language (visited on 10/04/2017).

Iyyer, Mohit et al. (2015). "Deep Unordered Composition Rivals Syntactic Methods for Text Classification". 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)*. Vol. 1, pp. 1681–1691. URL: http://www.aclweb.org/anthology/P15-1162 (visited on 10/04/2017).

Jurafsky, Dan and James H. Martin (2014). "Dependency Parsing". In: *Speech and Language Processing*. Pearson London.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). "A Convolutional Neural Network for Modelling Sentences". In: *arXiv preprint arXiv:1404.2188*. URL: https://arxiv.org/abs/1404.2188 (visited on 10/02/2017).

Kendall, Tyler, Joan Bresnan, and Gerard van Herk (2011). "The Dative Alternation in African American English: Researching Syntactic Variation and Change across Sociolinguistic Datasets". In: *Corpus Linguistics and Linguistic Theory* 7.2. ISSN: 1613-7027, 1613-7035. DOI: 10.1515/cllt.2011.011.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *arXiv preprint arXiv:1408.5882*. URL: https://arxiv.org/abs/1408.5882 (visited on 10/04/2017).

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 11/21/2017).

Kiros, Ryan et al. (2015). "Skip-Thought Vectors". In: arXiv: 1506.06726 [cs]. URL: https://arxiv.org/pdf/1506.06726.pdf (visited on 09/28/2017).

Landauer, Thomas K. and Susan T. Dumais (1997). "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge". In: *Psychological Review*, pp. 211–240.

Lapata, Maria (1999). "Acquiring Lexical Generalizations from Corpora: A Case Study for Diathesis Alternations". In: *Proceedings of the 37th Annual Meeting of the Association for*

Computational Linguistics on Computational Linguistics. Association for Computational Linguistics, pp. 397–404.

Lau, Jey Han and Timothy Baldwin (2016). "An Empirical Evaluation of Doc2vec with Practical Insights into Document Embedding Generation". In: arXiv: 1607.05368 [cs]. URL: http://arxiv.org/abs/1607.05368 (visited on 10/02/2017).

Le, Quoc and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196. URL: http://www.jmlr.org/proceedings/papers/v32/le14.pdf (visited on 10/02/2017).

LeCun, Yann and Yoshua Bengio (1995). "Convolutional Networks for Images Speech and Time Series". In: URL: http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95a.pdf.

Lehmann, Erich L. and George Casella (1998). *Theory of Point Estimation*. 2nd ed. Springer Texts in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-98502-2. URL: //www.springer.com/de/book/9780387985022 (visited on 02/14/2018).

Levin, Beth (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago: University of Chicago Press. 348 pp. ISBN: 978-0-226-47532-5 978-0-226-47533-2.

Levy, Omer and Yoav Goldberg (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems*, pp. 2177–2185. URL: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization (visited on 09/05/2017).

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings". In: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225. URL: https://www.transacl.org/ojs/index.php/tacl/article/view/570 (visited on 06/22/2017).

Liu, Pengfei et al. (2015). "Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents." In: *EMNLP*, pp. 2326–2335. URL: https://www.aclweb.org/anthology/D/D15/D15-1280.pdf (visited on 10/04/2017).

Lund, Kevin and Curt Burgess (1996). "Producing High-Dimensional Semantic Spaces from Lexical Co-Occurrence". In: *Behavior Research Methods, Instruments, & Computers* 28.2, pp. 203–208. URL: http://link.springer.com/article/10.3758/BF03204766 (visited on 09/25/2017).

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. OCLC: ocn190786122. New York: Cambridge University Press. 482 pp. ISBN: 978-0-521-86571-5.

Marelli, Marco, Luisa Bentivogli, et al. (2014). "SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment." In: *SemEval@ COLING*, pp. 1–8. URL: https://www.aclweb.org/anthology/S/S14/S14-2.pdf#page=21 (visited on 08/14/2017).

Marelli, Marco, Stefano Menini, et al. (2014). "A SICK Cure for the Evaluation of Compositional Distributional Semantic Models." In: *LREC*, pp. 216–223. URL: http://clic.cimec.unitn.it/marco/publications/marelli-etal-sick-lrec2014.pdf (visited on 06/12/2017).

Mikolov, Tomas, Kai Chen, et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781*. URL: http://arxiv.org/abs/1301.3781 (visited on 12/12/2016).

Mikolov, Tomas, Ilya Sutskever, et al. (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Advances in Neural Information Processing Systems*, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality (visited on 10/02/2017).

Misra, Amita, Brian Ecker, and Marilyn A. Walker (2016). "Measuring the Similarity of Sentential Arguments in Dialog". In: *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 276. URL: http://www.aclweb.org/anthology/W/W16/W16-36.pdf#page=294 (visited on 06/12/2017).

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of Machine Learning*. Adaptive computation and machine learning. OCLC: 812407408. Cambridge, Mass. London: The MIT Press. 412 pp. ISBN: 978-0-262-01825-8.

Mueller, Jonas and Aditya Thyagarajan (2016). "Siamese Recurrent Architectures for Learning Sentence Similarity." In: *AAAI*, pp. 2786–2792. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12195/12023 (visited on 07/26/2017).

Naili, Marwa, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala (2017). "Comparative Study of Word Embedding Methods in Topic Segmentation". In: *Procedia Computer Science* 112, pp. 340–349. ISSN: 18770509. DOI: 10.1016/j.procs.2017.08.009. URL:

http://linkinghub.elsevier.com/retrieve/pii/S1877050917313480 (visited on 02/12/2018).

Nivre, Joakim et al. (2016). “Universal Dependencies v1: A Multilingual Treebank Collection.” In: *LREC*.

Niwa, Yoshiki and Yoshihiko Nitta (1994). “Co-Occurrence Vectors from Corpora vs. Distance Vectors from Dictionaries”. In: *Proceedings of the 15th Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pp. 304–309.

Opitz, David W. and Richard Maclin (1999). “Popular Ensemble Methods: An Empirical Study”. In: *J. Artif. Intell. Res.(JAIR)* 11, pp. 169–198.

Parikh, Ankur P. et al. (2016). “A Decomposable Attention Model for Natural Language Inference”. In: arXiv: 1606.01933 [cs]. URL: http://arxiv.org/abs/1606.01933 (visited on 12/10/2017).

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). “On the Difficulty of Training Recurrent Neural Networks”. In: arXiv: 1211.5063 [cs]. URL: http://arxiv.org/abs/1211.5063 (visited on 07/28/2017).

Pearson, Karl (1895). “Note on Regression and Inheritance in the Case of Two Parents”. In: *Proceedings of the Royal Society of London (1854-1905)* 58.-1, pp. 240–242. ISSN: 0370-1662. DOI: 10.1098/rspl.1895.0041. URL: http://rspl.royalsocietypublishing.org/cgi/doi/10.1098/rspl.1895.0041 (visited on 11/27/2017).

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “Glove: Global Vectors for Word Representation”. In: *EMNLP*.

Polyak, B. T. (1964). “Some Methods of Speeding up the Convergence of Iteration Methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5, pp. 1–17. ISSN: 0041-5553. DOI: 10.1016/0041-5553(64)90137-5. URL: http://www.sciencedirect.com/science/article/pii/0041555364901375 (visited on 09/12/2017).

Resnik, Philip (1999). “Semantic Similarity in a Taxonomy: An Information-Based Measure and Its Application to Problems of Ambiguity in Natural Language”. In: *J. Artif. Intell. Res.(JAIR)* 11, pp. 95–130. URL: https://www.jair.org/media/514/live-514-1722-jair.pdf.

Rosenblatt, F. (1958). “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” In: *Psychological Review* 65.6, pp. 386–408. ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519. URL: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519 (visited on 11/27/2017).

Rumelhart, David E. and James L. McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. In collab. with San Diego University of California. Computational models of cognition and perception. Cambridge, Mass: MIT Press. 2 pp. ISBN: 978-0-262-18120-4.

Rumelhart, David E, Geoffrey E Hinton, Ronald J Williams, et al. (1988). "Learning Representations by Back-Propagating Errors". In: *Cognitive modeling* 5.3, p. 1.

Sahami, Mehran et al. (1998). "A Bayesian Approach to Filtering Junk E-Mail". In: *Learning for Text Categorization: Papers from the 1998 Workshop*. Vol. 62, pp. 98–105.

Salton, Gerard, A. Wong, and Chung-Shu Yang (1975). "A Vector Space Model for Automatic Indexing". In: *Commun. ACM* 18, pp. 613–620.

Schakel, Adriaan M. J. and Benjamin J. Wilson (2015). "Measuring Word Significance Using Distributed Representations of Words". In: arXiv: 1508.02297 [cs]. URL: http://arxiv.org/abs/1508.02297 (visited on 06/14/2017).

Schütze, Hinrich (1998). "Automatic Word Sense Discrimination". In: *Comput. Linguist.* 24.1, pp. 97–123. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972719.972724 (visited on 02/14/2018).

Simonyan, Karen and Andrew Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: arXiv: 1409.1556 [cs]. URL: http://arxiv.org/abs/1409.1556 (visited on 02/14/2018).

Socher, Richard, Eric H. Huang, et al. (2011). "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection". In: *Advances in Neural Information Processing Systems*, pp. 801–809. URL: http://papers.nips.cc/paper/4204-dynamic-pooling-and-unfolding-recursive-autoencoders-for-paraphrase-detection.pdf (visited on 10/04/2017).

Socher, Richard, Brody Huval, et al. (2012). "Semantic Compositionality through Recursive Matrix-Vector Spaces". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 1201–1211. URL: http://dl.acm.org/citation.cfm?id=2391084 (visited on 10/04/2017).

Socher, Richard, Alex Perelygin, et al. (2013). "Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. URL: http://www.aclweb.org/anthology/D13-1170 (visited on 10/02/2017).

Sparck Jones, Karen (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Journal of documentation* 28.1, pp. 11–21. URL: http://www.emeraldinsight.com/doi/abs/10.1108/eb026526 (visited on 09/04/2017).

Spearman, C. (1904). "The Proof and Measurement of Association between Two Things". In: *The American Journal of Psychology* 15.1, pp. 72–101. ISSN: 00029556. DOI: 10.2307/1412159. JSTOR: 1412159.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958. URL: http://www.jmlr.org/papers/volume15/srivastava14a.old/source/srivastava14a.pdf (visited on 06/14/2017).

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning (2015). "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks". In: arXiv: 1503.00075 [cs]. URL: http://arxiv.org/abs/1503.00075 (visited on 06/12/2017).

Tesnière, Lucien (1976). *Éléments de Syntaxe Structurale*. 2e éd. revue et corrigée. Paris: Klincksieck. 674 pp. ISBN: 978-2-252-01861-3.

Turney, Peter D. and Patrick Pantel (2010). "From Frequency to Meaning: Vector Space Models of Semantics". In: DOI: 10.1613/jair.2934. arXiv: 1003.1141 [cs]. URL: http://arxiv.org/abs/1003.1141 (visited on 12/02/2017).

Uszkoreit, Hans et al. (2017). "Common Round: Application of Language Technologies to Large-Scale Web Debates". In: *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, pp. 5–8. URL: http://aclweb.org/anthology/E17-3002.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 08/17/2017).

Wang, Shaonan and Chengqing Zong (2017). "Comparison Study on Critical Components in Composition Model for Phrase Representation". In: *ACM Transactions on Asian and Low-Resource Language Information Processing* 16.3, pp. 1–25. ISSN: 23754699. DOI: 10.1145/3010088.

Weingarten, Evan et al. (2016). "From Primed Concepts to Action: A Meta-Analysis of the Behavioral Effects of Incidentally Presented Words". In: *Psychological Bulletin* 142.5, pp. 472–497. ISSN: 1939-1455. DOI: 10.1037/bul0000030. pmid: 26689090.

Weischedel, Ralph et al. (2011). “OntoNotes: A Large Training Corpus for Enhanced Processing”. In:

Wieting, John, Mohit Bansal, Kevin Gimpel, and Karen Livescu (2015). “Towards Universal Paraphrastic Sentence Embeddings”. In: arXiv: 1511.08198 [cs]. URL: https://arxiv.org/pdf/1511.08198.pdf (visited on 10/02/2017).

Wieting, John, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth (2015). “From Paraphrase Database to Compositional Paraphrase Model and Back”. In: arXiv: 1506.03487 [cs]. URL: http://arxiv.org/abs/1506.03487 (visited on 06/22/2017).

Wu, Yonghui et al. (2016). “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: arXiv: 1609.08144 [cs]. URL: http://arxiv.org/abs/1609.08144 (visited on 02/13/2018).

Yin, Wenpeng and Hinrich Schütze (2015). “Convolutional Neural Network for Paraphrase Identification”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 901–911. URL: https://aclanthology.info/pdf/N/N15/N15-1091.pdf (visited on 10/04/2017).

Zeiler, Matthew D. (2012). “ADADELTA: An Adaptive Learning Rate Method”. In: arXiv: 1212.5701 [cs]. URL: http://arxiv.org/abs/1212.5701 (visited on 09/12/2017).

Zhao, Han, Zhengdong Lu, and Pascal Poupart (2015). “Self-Adaptive Hierarchical Sentence Model”. In: arXiv: 1504.05070 [cs]. URL: http://arxiv.org/abs/1504.05070 (visited on 10/04/2017).

Zhao, Jiang, Tiantian Zhu, and Man Lan (2014). “ECNU: One Stone Two Birds: Ensemble of Heterogenous Measures for Semantic Relatedness and Textual Entailment.” In: *SemEval@ COLING*, pp. 271–277.

Zipf, George (1935). *The Psychobiology of Language: An Introduction to Dynamic Philology.* Cambridge, Mass.: M.I.T. Press.

Declaration of Authorship

I hereby confirm that I have authored this Student Research Paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, May 4, 2018

Arne Binder