Institut für Computerlinguistik
Universität Heidelberg

Dimensionality Reduction in Semantic Vector Spaces Using a Derivational Resource

Bachelor's thesis
July 16, 2014

Julia Kreutzer
kreutzer@cl.uni-heidelberg.de

Supervisor	Prof. Dr. Sebastian Padó
Reviewers	Prof. Dr. Anette Frank
	Prof. Dr. Sebastian Padó

Abstract

Lexical semantic vector spaces model the meaning of words by representing the co-occurrence statistics of words in certain contexts. Words are considered similar if they occur in similar contexts, so that word similarity can get predicted by comparing their representation in the semantic vector space.

Two major problems in those vector spaces are their size and their sparsity. Due to the characteristics of language, semantic vector space models built from large corpora tend to grow immensely in dimensionality and therefore in size. They capture co-occurrence information in a sparse way. On the one hand sparse co-occurrence matrices can be represented and stored more efficiently than dense, completely filled matrices, but on the other hand they limit the vector space's ability to make semantic predictions. Dimensionality reduction techniques aim to solve this problem by reducing the vector spaces' number of dimensions.

With this thesis, we introduce a linguistically motivated approach for dimensionality reduction: We abstract from the meaning of single words to the meaning of derivational families building on DErivBase [Zeller *et al.*, 2013], a resource for German which groups words into derivational families. DErivBase allows us to partition the original dimensions into equivalence classes corresponding to derivational families. The result of this transformation is a dimensionality-reduced vector space with derivational families representing the dimensions.

In two experiments (looking at word-based and dependency-based semantic vector spaces, respectively), we evaluate on two benchmark data sets for two standard tasks, namely semantic relatedness scoring and synonym detection, with a varying range of transformation parameters. We analyse the following aspects: dimensionality and sparsity reduction, size, and quality of semantic relatedness predictions. We compare our approach against competitors and baselines, including singular value decomposition (*SVD*) and a dimension selection approach (*top-n*).

We find that our novel technique brings about a substantial reduction of dimensions, and allows highly compact representation and storage of the vector space model. It performs better on word-based than on dependency-based vector space models, and better on semantic relatedness than on synonym choice tasks. It is superior to *SVD* in terms of accuracy and superior to *top-n* in terms of coverage. Nevertheless, we record losses in accuracy compared to the original model. Our analysis suggests that future work should attempt to reduce the influence of the noise inherent in large DErivBase clusters.

Zusammenfassung

Lexikalische semantische Vektorräume modellieren die Bedeutung von Wörtern, indem sie die Kookkurrenzstatistiken von Wörten in bestimmten Kontexten repräsentieren. Wörter sind einander ähnlich, wenn sie in ähnlichen Kontexten auftreten, sodass man Wortähnlichkeit über den Vergleich ihrer Repräsentationen im semantischen Vektorraum vorhersagen kann.
Diese Vektorräume haben vor allem mit folgenden zwei Problemen zu kämpfen: mit der Größe und der Spärlichkeit („sparsity"), die eine geringe Dichte in der zugehörigen Matrix beschreibt. Aufgrund der spezifischen Eigenschaften der natürlichen Sprache und ihrer Verwendung tendieren semantische Vektorraummodelle, die auf Häufigkeitszählungen innerhalb großer Korpora basieren, zu einer sehr hohen Anzahl von Dimensionen. Sie erfassen Kookkurrenzstatistiken auf spärliche Art und Weise. Einerseits ermöglicht das eine effiziente Repräsentation und Speicherung der Matrizen, andererseits schränken sie aber auch den Rahmen semantischer Vorhersagen ein. Methoden zur Dimensionsreduktion versuchen diese Probleme zu lösen, indem sie die Anzahl der Dimensionen des Vektorraums verringern.
In dieser Bachelorarbeit stellen wir einen neuen, linguistisch motivierten Ansatz zur Reduktion von Dimensionen in lexikalischen semantischen Vektorräumen vor: Wir abstrahieren von der Bedeutung einzelner Wörter auf die Bedeutung von Wortfamilien, wozu wir DErivBase [Zeller *et al.*, 2013] nutzen, eine Ressource für die deutsche Sprache, die Wörter in ihre derivationellen Familien einteilt. Mithilfe von DErivBase partitionieren wir die ursprünglichen Dimensionen entsprechend ihrer derviationellen Familien in Äquivalenzklassen. Das Ergebnis dieser Transformation ist ein dimensionsreduzierter Vektorraum, dessen Dimensionen von derivationellen Familien repräsentiert werden.
In zwei Experimenten (mit jeweils wortbasierten und syntaktischen Vektorräumen) evaluieren wir auf zwei Testdatensätzen für zwei Standardaufgaben, nämlich die Bewertung von semantischer Verwandtheit und die Erkennung von Synonymen. Dabei analysieren wir die Aspekte Dimensions- und Sparsity-Reduktion, Größe des Vektorraumes, und Qualität von Vorhersagen zu semantischer Verwandtheit. Wir führen Vergleiche mit Vektorräumen der gleichen Größe durch, die durch Singulärwertzerlegung (Singular Value Decomposition, *SVD*) und die Auswahl der häufigsten Dimensionen (*top-n*) reduziert wurden.

Es stellt sich heraus, dass unser Ansatz eine erheblich Reduktion der Anzahl der Dimensionen bewirkt und gleichzeitig eine kompakte Repräsentation und Speicherung des Vektorraummodells erlaubt. Die Transformation erzielt auf dem wort-basierten Raum bessere Ergebnisse als auf dem syntaktischen. In der Evaluationsaufgabe zur Bestimmung von semantischer Verwandtheit von Wörtern schneiden die transformierten Modelle zudem besser ab als in der Synonymauswahl. Sie übertreffen *SVD* in der Genauigkeit („accuracy") und *top-n* in der Abdeckung („coverage"). Trotzdem verzeichnen wir Verluste in der Genauigkeit im Vergleich mit dem ursprünglichen Vektorraum. Deshalb schlagen wir zur Verbesserung von zukünftigen Experimenten vor, den Einfluss des Rauschens von großen DErivBase-Cluster zu beschränken.

Contents

List of Figures

List of Tables

1 Introduction

Semantic vector space models are widely used in language technology to capture the meaning of words, phrases and documents. The crucial hypothesis, which lays the foundation for distributional and statistical semantics, was first expressed by [Harris, 1954] and then elaborated by many others (e.g. [Deerwester *et al.*, 1990]). Named the distributional hypothesis, it states that words occurring in similar contexts tend to have similar meanings. In order to subsume multiple proposed and experimentally supported hypotheses in the field of distributional semantics, [Turney & Pantel, 2010] later formulated the statistical semantics hypothesis stating that "statistical patterns of human word usage" [Turney & Pantel, 2010] can be used to examine the meaning of words.
These distributional statistics are obtained by observing co-occurrence frequencies of targets in predefined contexts [Turney & Pantel, 2010] in natural language corpora. Targets are represented by vectors in a co-occurrence matrix $M_{n \times m}$ with n targets $T = \{t_1, t_2, t_3, ..., t_n\}$ and m contexts $C = \{c_1, c_2, c_3, ..., c_m\}$: The jth value of the ith vector tells how often the target t_i is observed in the context c_j. Measuring their vectors' similarity in the constructed vector space is hence a way to estimate the distributional similarity between arbitrary targets. By choosing single words as targets, we can thus obtain a model to predict semantic similarity between words.
The choice of the context influences the resulting matrix and its reliability to a high extent. Essentially, the number of distinct contexts, that is the number of dimensions in the co-occurrence matrix, and the contexts' complexity have to be considered carefully. In the field of lexical semantics, where the focus is on the meaning of words, it is common to use single words or words combined with syntactical information as contexts (word space models) instead of e.g., documents as in document space models [Deerwester *et al.*, 1990]. Lexical semantic vector space models have successfully been applied to various tasks, such as synonymy detection [Landauer & Dumais, 1997] (e.g. TOEFL multiple choice tasks), word sense discrimination [Schütze, 1998], and similarity judgements [McDonald, 2000].
As Zipf's law [Zipf, 1949] states, there are many infrequent words in natural language utterances and only a few frequent ones, and each of the infrequent word co-occurs with many other infrequent words, which increases the number of distinct contexts immensely. This is why lexical semantic vector spaces tend to have high-dimensional matrices. On

the one hand, they heavily rely on the high number of dimensions in order to capture the characteristics of the underlying corpus adequately. On the other, the high number of dimensions in turn raises the problem of impaired coverage through sparsity[1]. As we introduced in the abstract, sparsity has the positive effect of enabling an efficient storage and representation for the model, but the influence on the model's coverage is clearly an undesirable consequence: Although a model might contain a high number of dimensions and therefore represent the underlying language source precisely, the model will predict zero similarities for a high number of word pairs. This is due to the lack of common co-occurrences for these word pairs.

Dimensionality reduction techniques tackle this problem and aim to reduce the number of dimensions in the vector space without losing information and accuracy. Truncated Singular Value Decomposition (SVD), a common technique used for the reduction of matrices in various fields of applications, performs quite well on semantic vector spaces, but does not allow an intuitive linguistic interpretation of the reduced dimensions as co-occurrence contexts.

With this thesis we introduce a novel approach of reducing the dimensions in semantic vector spaces by transforming them with the help of a derivational resource. It results in dimensionality-reduced vector spaces with dimensions that can be justified by intuition and linguistic knowledge. The dimensions are formed by derivational families, that are clusters of words derived from the same morphological root. Thus the transformed vector space represents the co-occurrence statistics of words with derivational clusters, assuming that an abstraction to the core meaning of a group of derivationally-related words is still precise enough. The resource for the clustering is the German DErivBase resource developed by [Zeller *et al.*, 2013]. In this thesis we focus on investigating the potential of this dimensionality reduction technique applied to two types of lexical semantic vector spaces, word-based and dependency-based spaces.

In the following we first introduce the reader to the basics of semantic vector spaces and related work on this field in chapter 2, then establish the theoretical principles and methods of our approach in chapter 3, and subsequently conduct some experiments in chapter 4 including an application of word-based and dependency-based models to the tasks of semantic relatedness scoring and synonym detection. The experimental results will be evaluated, discussed, and explained in chapter 5 with regard to the reduction's efficiency, the transformed models' size and their semantic reliability. Finally we conclude

1 The term "sparsity" describes the fraction of zero values in a matrix: if this is low, a matrix is regarded as "sparse", the opposite is a "dense" matrix.

in chapter 6 with an estimation of its applicability in the field of distributional semantics and suggestions for further research.

2 Related Work

In this section, we provide detailed background information about the aspects that were mentioned in the introduction. First, the development and the methods of semantic vector spaces in the field of lexical semantics are further elaborated, since they lay the foundations for our work. Subsequently, we explain the principles of truncated SVD in detail, and then present DErivBase, the resource that our approach for dimension reduction is based on.

2.1 Semantic Vector Spaces in Lexical Semantics

Formally, a semantic vector space is defined as a quadruple of $\langle A, B, S, M \rangle$ [Lowe, 2001], where A is a lexical association function, B is a set of basis elements representing the dimensions, S a similarity measure, and M a transformation that maps one space onto another[1]. In the following we describe how a semantic vector space is constructed, and at which construction steps these elements of the vector space are relevant.

1. Linguistic pre-processing:
   As explained in the introduction, semantic vector spaces serve the purpose of capturing the distributional statistics of language. These statistics need to be extracted from a corpus, a collection of spoken or written text. This usually includes various linguistic pre-processing steps, such as tokenization[2], normalization[3], and annotation, for instance with Part-of-Speech (POS) tags, dependency paths or word senses. The output of the linguistic pre-processing are word tokens or types, potentially

1 There are several definitions described in literature: For instance, [Padó & Lapata, 2007] define a semantic vector space as a tuple with eight elements in order to find an adequate definition that optimally suits both word-based and dependency-based spaces. Our focus is not on the parametrization and creation of semantic vector spaces, but on the usage. This is why we adhere to a simpler definition that is sufficient for our purposes.

2 Tokenization: split the text into single-word tokens

3 Normalization: normalize the text to word types, e.g., by a stemmer reducing tokens to their word stems

extended by annotations.

2. Recording co-occurrence events:
   The linguistically processed texts are further processed mathematically. First, co-occurrence events are recorded in a matrix with a vector for targets T and contexts C as dimensions. It must be defined which parts of the linguistic environment are considered as relevant context to each target. The relevant contexts are collected in B, the set of basis elements for the vector space. The definition of contexts and the subsequent categorization of semantic vector spaces is particularized later in this section.

3. Weighting:
   In order to normalize the frequencies (more frequent words should not generally be more similar), a lexical association or weighting function A is applied to the matrix. Often-used weighting functions are Pointwise Mutual Information (PMI, [Fano, 1961, Church & Hanks, 1990] and Local Mutual Information (LMI, [Evert, 2005]). They are based on single and joint probabilities, computed from the co-occurrence frequencies $f(t,c)$ of a target $t \in T$ with a context $c \in C$. $f(*,*)$ is the marginal frequency, i.e., the sum of all values in the co-occurrence matrix. $f(*,c)$ is the marginal frequency for each column ("How often does a context occur with any target?"), and $f(t,*)$ the marginal frequency for each row ("How often does a target occur in any context?").

   PMI(t,c) $\equiv log\frac{p(c|t)}{p(c)} = log\frac{p(t,c)}{p(t)\,p(c)}$
   LMI(t,c) $\equiv p(t,c)\; log\frac{p(t,c)}{p(t)\,p(c)}$
   where $p(t,c) = \frac{f(t,c)}{f(*,*)}$, $p(t) = \frac{f(t,*)}{f(*,*)}$, and $p(c) = \frac{f(*,c)}{f(*,*)}$

4. Transformation:
   The weighted matrix may also get transformed (defined by M) by external smoothing methods which usually aim to increase the coverage for given tasks (section 2.3 presents a smoothing technique using DErivBase).

5. Measuring Similarity:
   The weighted and smoothed co-occurrence matrix can now be used to assess word similarity. For two given target words P and Q, the similarity score is calculated by applying a distance or similarity measure S to their according vectors p and q in the multi-dimensional space. The most popular similarity measure is the cosine, which produces scores between -1 and 1, -1 expressing minimal semantic relatedness, 1 accordingly the maximal relatedness. From their similarity score we can thus tell how semantically similar the targets P and Q are. Note that the cosine predicts a similarity of zero if there is no common context that both targets co-occur with.

$$\text{sim}_{cos} = \frac{p \cdot q}{\|p\| \|q\|} = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} (p_i)^2} \times \sqrt{\sum_{i=1}^{n} (q_i)^2}}$$

The semantic similarity that we analyse is the attributional similarity: We measure the correspondence of the targets' properties or attributes, which are defined by the vector space's contexts. The definition of the contexts is thus crucial to the nature of the similarity estimates that we obtain.
By intuition we understand the context of a word as its neighbouring words.
[Salton *et al.*, 1975] coined the term of "bag of words" that describes a method to capture these neighbouring words. It is defined as a multiset, a set that allows duplicates. The bag of words for each target is constructed by collecting the words that occur around the target word in a certain context window. A window of ± 5 for instance means that we consider the five words before the target and the five words after the target. Usually, function words and very frequent and therefore uninformative words are filtered out. Vector space models that use this bag-of-words representation are referred to as word-based models.
Word-based models are simple to construct as they do rely on the mere counting of words. However, the drawback is that they do not contain any information about the linguistic structure that connects the target and the context words. It might for example be important to make a distinction in the concept of two words if one of them only occurs in subject position of the verb "eat", and the other as direct object. The "eater", for instance a horse, and the "eaten", for instance an apple, would otherwise get assigned a very high similarity, even though they represent distinct concepts (an animal and a fruit, respectively). This example illustrates that the use of unstructured context information leads to a rather coarse representation of semantic relatedness.

[Padó & Lapata, 2007] therefore introduced another category of semantic vector spaces that makes use of syntactic dependency information between the targets and their context words. They are referred to as “dependency-based” vector space models. Contexts can be defined as pairs of syntactic dependency links and context words. For instance, “Horse” and “apple” in this representation would only get assigned a minor similarity, as the former often co-occurs with the context “subj-eat” and the latter with “obj-eat”. This more structured notion of context allows a finer-grained and deeper estimation of the semantic similarity between words.
[Baroni & Lenci, 2010] developed the *Distributional Memory* (*DM*), a framework for distributional semantics that represents distributional information not in the form of a traditional multidimensional vector space, but in a three-dimensional tensor. The tensor contains weighted word-link-word triples, where links are dependency links that connect both words. The weights for the triples are LMI-weighted corpus counts (“How often does each word-link-word combination occur in a given corpus?”). The tensor representation allows a more flexible application compared to traditional vector spaces: Instead of constructing one vector space for each purpose, several vector spaces can be derived through matricization from the same single tensor. One can for example build a word by word-link vector space, where targets are words and dimensions are pairs of links and words, but also a word by word (i.e., word-based) vector space, or a word-word by link vector space. Depending on their type, the resulting vector spaces model thus model different aspects of distributional semantics, e.g., word similarity (word by word-link or word by word spaces) or semantic relations (word-word by link space).
[Padó & Utt, 2012] implemented *DM* for German and constructed *DM.de* through parallel induction from English to German with words, links and weights extracted from the SdeWaC corpus (see 4.1.1 for more details about SdeWaC). We use vector spaces built from *DM.de* in our experiments, where the dependency links essentially influence the transformation. Therefore we take a closer look at the characteristics of the dependency links used in *DM.de*. The links include unlexicalized syntactic patterns like “subj-tr”, “iobj”, or “verb”, and lexicalized patterns, like “n_1 prep n_2” or “n_1 verb n_2”, where n_1 and n_2 represent the words that occur in the specific syntactic dependency. From a sentence like e.g., “Der Taxifahrer parkt das Auto im Halteverbot”, the following triples are collected: “Taxifahrer, subj-tr, parken”, “parken, in, Halteverbot”, “Auto, obj, parken”, “Taxifahrer, parken, Auto”. Also inverse links, marked with “-1”, are processed accordingly: “parken, subj-tr-1, Taxifahrer”, “Halteverbot, in-1, parken”, “parken, obj-1, Auto”, “Auto, parken-1, Taxifahrer”.

2.2 Dimension Reduction with SVD

We explained in the introduction that semantic vector spaces, especially dependency-based vector spaces for their higher number of distinct contexts, are prone to the problem of sparsity. Truncated SVD (see e.g. [Cline & Dhillon, 2006]) is a dimension reduction technique that was applied to semantics[4] by [Deerwester *et al.*, 1990] in their Latent Semantic Analysis (LSA), inspired by Latent Semantic Indexing (LSI) that is used in the field of information retrieval [Hofmann, 1999]. LSA discovers latent meaning components of the original matrix that form the new dimensions. This is achieved by applying truncated SVD on a word-context (e.g. paragraphs) matrix. SVD performs a matrix factorization by decomposing the original matrix into its singular values and right- and left-singular vectors. By selecting the top k eigenvalues (=truncating the matrix) and recomposing a matrix from it, it subsequently approximates the original matrix in a lower dimensional space: The original matrix $A_{n\times m}$ is decomposed into the product of the three matrices U, Σ, and V^T.

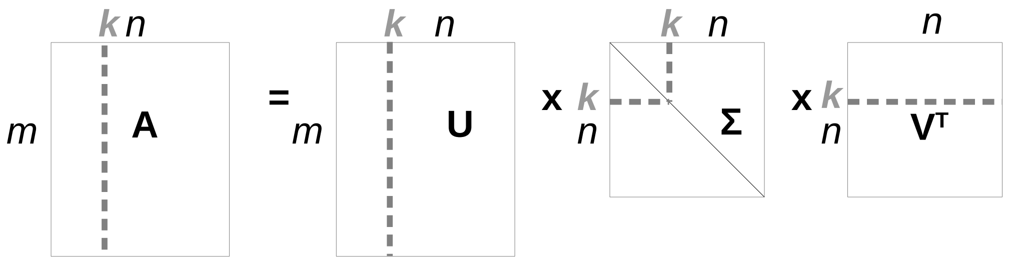

Figure 2.1: Truncated Singular Value Decomposition: The original matrix $A_{n\times m}$ is approximated in a lower-dimensional (here: k-dimensional) space by the product of the truncated matrices U, Σ and V^T.

U contains derived left-singular vectors, V^T contains right-singular vectors. In both matrices all vectors are linear independent. Σ is a square diagonal matrix (non-zero values only

4 SVD is a highly interdisciplinary used method for dimension reduction and particularly established in signal processing and pattern recognition.

in the cells along the diagonal) with sorted (descending) singular values. They represent the amount of variance in the original matrix that is captured by each dimension. By selecting the top *k* dimensions (highlighted in grey in figure 2.1) we ignore the least *n-k* singular values and linked left-singular and right-singular vectors. From these dimensions a matrix with lower dimensionality $A_{n \times k}$ is constructed by the product of the three truncated matrices.
The most variant dimensions are selected since they bear the highest degree of relevant information, i.e., they contribute best to a meaningful representation of words in the semantic vector space. [Deerwester *et al.*, 1990] found that LSA performed on a synonym tests equally well as the average of sample students which allowed them to draw parallels to human learning.

2.3 DErivBase: Construction and Application

Our dimension reduction technique is based on derivational information. This information is extracted from DErivBase, a derivational resource for German. As introduction to DErivBase, we will first report how it was built and then present one approach that uses DErivBase for smoothing of semantic vector spaces.

The motivation behind DErivBase was to build a high-coverage derivational knowledge for German, a morphologically rich language. [Zeller *et al.*, 2013] proposed a rule-based framework for inducing derivational families and applied it to create the German resource DErivBase.
Surface-based derivation rules for nouns, adjectives and verbs were defined with the help of linguistic textbooks. A rule might for example describe how the noun "Tag" ("day") is adjectivized to "täglich" ("daily"). The rules were applied to lemmas extracted from a POS-tagged version of SdeWaC ([Faaß & Eckart, 2013], see 4.1.1 for more details) to induce derivational families, each family containing all lemmas that were derived from an input lemma. Each pair of lemmas in a family is thus connected by a derivational rule path of a certain length.
For example the input lemma "Gespenst_Nn" ("ghost") was adjectivized to "gespenstisch_A" and "gespenstig_A" by suffixation (DErivBase rules "dNA05" and "dNA02").

"Gespenstische_N" was furthermore formed by nominalization through suffixation (rule "dAN01") of the adjective, such that the whole derivational family for the input lemma "Gespenst_Nn" is: "Gespenstische_N Gespenst_Nn gespenstisch_A gespenstig_A".
[Zeller *et al.*, 2013] provide their resource in two different formats: "DErivBase-v1.4.1-families.txt", where all families are listed as clusters, and "DErivBase-v1.4.1-rulePaths.txt", where each two lemmas are connected with their derivational rule path. Also the derivational rules, such as "dNA01", are listed and explained. The current version 1.4.1[5] covers 280,336 lemmas which are grouped into 17,314 non-singleton families and 210,899 singleton families. In a quantitative evaluation against manual annotations, the current version achieves a precision of 85%, and a recall of 91%.

Derivationally related words are assumed to generally show a high degree of semantic relatedness. They share the same morphological root, for the above example this would be "gespenst", which is the foundation of their individual meaning. To the basis meaning of "gespenst" describing a ghost, the derived words add some word-class specific meaning. The adjectives "gespenstig" and "gespenstisch" for instance describe the essence of being a ghost or the attributes of a ghost.
Derivational relatedness was therefore employed in several semantic applications as in question answering [Jacquemin, 2010] and textual entailment [Szpektor & Dagan, 2008]. [Padó *et al.*, 2013] furthermore used the derivational information from DErivBase for smoothing semantic vector spaces, in order to increase the coverage of sparse dependency-based vector spaces. In a given vector space, the vector for the target "Gespenst_Nn" might for instance be quite dense, whereas the vector for "gespenstig_A" might be rather sparse due to its infrequent use in the underlying corpus. Since those two targets are semantically closely related, their vectors should be similar, too. Derivational smoothing makes use from this connection and re-calculates the vectors for words under the influence of the vectors representing derivationally related words. [Padó *et al.*, 2013] therefore implemented and evaluated several manners of operationalizing this idea. On the semantic similarity prediction task (see 4.1.4.2 for details about the task), they achieved an improvement in correlation with manual annotations, and coverage (nearly 7% and 50%, respectively). On a synonym choice task (see 4.1.4.3 for details about the task) derivational smoothing increased the coverage by 7% but impaired the accuracy slightly.

5 For details about version-dependent changes refer to `http://www.cl.uni-heidelberg.de/~zeller/res/derivbase/derivbase_doc.txt`.

Consequently, their work has demonstrated that derivational knowledge can be used to improve the coverage of semantic similarity estimates in syntactic vector spaces.

3 Methods for Dimensionality Reduction with DErivBase

The motivation for this work is similar to the approach of [Padó *et al.*, 2013] to perform derivational smoothing: We tackle the issue of sparsity in semantic vector space models by profiting from derivational information.
However, our approach does not fall into the category of smoothing, since it addresses the general characteristics of the vector space as a whole. Smoothing addresses the vector space's targets and aims to back-off for single poorly-covered items. Dimensionality reduction techniques directly affect the origin of sparseness: the number of dimensions. They can be applied before actually filling the vector space's matrix with co-occurrence counts, whilst smoothing works on a ready-made vector space with a filled matrix and uses already computed vectors. Hence, both smoothing and transformation can potentially be combined and are not mutually exclusive.
Our vector space transformation is based on the fact that derivationally related words tend to show a high degree of semantic similarity. This allows us to generalize over derivational families and reduce the meaning of words to the elements of meaning that all family members share. Specific characteristics are not important for the group of words, e.g., the information about their word classes. Nonetheless, the abstraction should not lead to overgeneralization that could result from removing essentially distinctive properties of the words' co-occurrence statistics.
Another advantage of our DErivBase transformation is that the dimensions itself remain intuitively understandable, which is not the case for SVD-transformed dimensions representing latent dimensions. In addition to that, it is less costly and less mathematically complex than SVD and related mechanisms in the case of extremely large matrices.

In the following sections we provide formal definitions for the vector space transformation with DErivBase. Since dimensions are differently defined in word-based and dependency-based models, distinctions between their transformations are required. First we will address the transformation of word-based vector spaces and then continue with the transformation of dependency-based spaces.

3.1 Transformation of Word-based Vector Spaces

We start with a given word-based vector space V_w with target words $T = \{t_1, t_2, t_3, ..., t_n\}$ and basis elements (or dimensions, also words) $W = \{w_1, w_2, w_3, ..., w_m\}$ and a derivational resource DR containing derivational clusters $D = \{d_1, d_2, d_3, ..., d_p\}$.
A DErivBase transformation partitions the original dimensions W into equivalence classes. Formally, we define a DErivBase transformation as a function dt that assigns each dimension $w \in W$ an equivalence class $w' \in W'$. It features a transformation condition and a transformation scheme. The transformation condition defines how the new dimensions W' are formed, and the transformation scheme defines how the co-occurrence weights for the new dimensions W' are computed. We combine these two parameters to clearly name and distinguish between several transformations, *add-words* for instance stands for a DErivBase transformation that uses the *words* condition and *add* scheme. In the following we define several transformation conditions and schemes for word-based vector spaces.

words: $dt_w : w \mapsto \mathcal{P}(w)$

$$\text{such that } dt_w(w) = \begin{cases} \{v \,|\, v \in d\,\} & \text{if } \exists d \in D : w \in d \\ \{w\} & \text{else} \end{cases}$$

The *words* transformation condition is characterized by: $w \in d$ and $v \in d$, which indicates that both original dimensions w and v must be found in the same family to be assigned the same equivalence class.
We name the new dimensions after their source, such that a new dimension representing a derivational family is labelled with the derivational families index in the DErivBase data set, and a dimension that was not affected by the transformation (the "else"-case) keeps its label. Given for instance the dimensions $W = \{$*ffahren, Fahrer, Straße, Motor, motorisiert, Radweg*$\}$, the *words*-transformation creates the set of new dimensions $W' = \{$*18, Straße, 886, Radweg*$\}$, as DErivBase family no. 18 contains both "fahren" and "Fahrer", and no. 886 contains "Motor" and "motorisiert".

The new dimensions are represented by sets of words, and their meaning somehow needs to get composed from the meaning of the single words. We propose three transformation schemes for this purpose:

- *add*: The co-occurrence weight for the target t with new context w' is the sum of all co-occurrence weights for the target t with all contexts w, where $dt_w(w) = w'$.
  Adding up co-occurrence weights implies that we understand the meaning of a cluster as the sum of the meanings of its components, such that the cluster holds all characteristics of its components.
- *multiply*: The co-occurrence weight for the target t with new context w' is the product of all co-occurrence weights for the target t with all contexts w, where $dt_w(w) = w'$.
  When we multiply weights to produce the weight for the whole cluster, we create a cluster that is represented by the features that all of its components share, since multiplication with a zero component weight leads to a zero target weight.
- *avg*: The co-occurrence weight for the target t with new context w' is the arithmetic mean of all co-occurrence weights for the target t with all contexts w, where $dt_w(w) = w'$.
  By averaging the components′ weights in the cluster we understand the cluster′s meaning as an average of the components′ meaning, such that the cluster still holds all their characteristics, but not as strongly as each single component.

The effects of the above transformation schemes are illustrated in the example vector spaces in tables 3.1 to 3.7 and the resulting similarity scores (cosine similarity). The vector space referred to as "plain" describes the vector space before the transformation. Changes due to the transformation scheme are marked in bold. In the lists of similarity predictions, the highest values within each list are marked: The word pairs obtaining the highest semantic similarity score are predicted to be most semantically similar.

Table 3.1: Plain: co-occurrence frequency vector space

	fahren	*Fahrer*	*Straße*	*Motor*	*motorisiert*	*Radweg*
Auto	20	1	5	4	0	1
Bus	2	13	5	1	9	0
Fahrrad	14	5	8	0	1	8
Fußgänger	1	0	17	0	0	2

Table 3.2: Plain: similarity predictions

lemma1	lemma2	sim_{cos}
Auto	*Bus*	.23
Bus	*Fahrrad*	.45
Fahrrad	*Fußgänger*	.52
Auto	*Fahrrad*	**.85**
Bus	*Fußgänger*	.30
Auto	*Fußgänger*	.30

Table 3.3: *Add-words*-transformed vector space

	18	*Straße*	***886***	*Radweg*
Auto	**21**	5	**4**	1
Bus	**15**	5	**10**	0
Fahrrad	**19**	8	**1**	8
Fußgänger	**1**	17	**0**	2

Table 3.4: *Add-words*: similarity predictions

lemma1	lemma2	sim_{cos}
Auto	*Bus*	**.87**
Bus	*Fahrrad*	.73
Fahrrad	*Fußgänger*	.30
Auto	*Fahrrad*	.85
Bus	*Fußgänger*	.21
Auto	*Fußgänger*	.20

Table 3.5: *Multiply-words*-transformed vector space

	18	*Straße*	***886***	*Radweg*
Auto	**20**	5	**0**	1
Bus	**26**	5	**9**	0
Fahrrad	**70**	8	**0**	8
Fußgänger	**0**	17	**0**	2

Table 3.6: *Multiply-words*: similarity predictions

lemma1	lemma2	sim_{cos}
Auto	*Bus*	.90
Bus	*Fahrrad*	.92
Fahrrad	*Fußgänger*	.09
Auto	*Fahrrad*	**.96**
Bus	*Fußgänger*	.12
Auto	*Fußgänger*	.17

Table 3.7: *Avg-words*-transformed vector space

	18	*Straße*	***886***	*Radweg*
Auto	**10.5**	5	**2**	1
Bus	**7.5**	5	**5**	0
Fahrrad	**9.5**	8	**0.5**	8
Fußgänger	**0.5**	17	**0**	2

Table 3.8: *Avg-words*: similarity predictions

lemma1	lemma2	sim_{cos}
Auto	*Bus*	**.77**
Bus	*Fahrrad*	.59
Fahrrad	*Fußgänger*	.39
Auto	*Fahrrad*	.69
Bus	*Fußgänger*	.32
Auto	*Fußgänger*	.30

We need to take into account that both transformation condition and scheme influence both coverage and accuracy of the transformed vector space's performance:

The transformation condition controls the extent of the dimensionality reduction and thereby the number of dimensions in the transformed vector space. A high number of dimensions generally increases the risk of higher sparsity and a consequently lower coverage. But a low number of dimensions increases the risk of overgeneralization resulting in lower accuracy. The transformation scheme affects the accuracy by defining the magnitude of the co-occurrence weights, but also the coverage when producing zero values.
Therefore it is not a trivial task to find the optimal combination of transformation condition and scheme and the best trade-off between coverage and accuracy for a given application or task.

3.2 Transformation of Dependency-based Vector Spaces

Given a dependency-based vector space V_d with target words $T = \{t_1, t_2, t_3, ..., t_n\}$ and basis elements, here pairs of words $w \in W$ and dependency links $l \in L$, and a derivational resource DR containing derivational clusters $D = \{d_1, d_2, d_3, ..., d_p\}$, we have more options to define transformation conditions than for a word-based vector space.
The transformation with the *words* condition is defined analogously to the word-based transformation described above (section 3.1) with slight changes due to the dimensions' ⟨word, link⟩-pair structure:

words:

$dt_{d1} : (LxW) \mapsto \mathcal{P}(LxW)$

such that $dt_{d1}(\langle l, w\rangle) = \begin{cases} \{\langle l', v\rangle \mid v \in d\} & \text{if } \exists d \in D : w \in d \\ \{\langle l', w\rangle\} & \text{else} \end{cases}$

This transformation is well-suited for word-based models, but when applying it to dependency-models we are forced to ignore the dependency links completely. In order to integrate this information into our transformed vector space, we define another two transformations:

wordlinks:

$dt_{d2} : (LxW) \mapsto \mathcal{P}(LxW)$

such that $dt_{d2}(\langle l, w\rangle) = \begin{cases} \{\langle l', v\rangle \mid v \in d \;\wedge l' = l\} & \text{if } \exists d \in D : w \in d \\ \{\langle l', w\rangle\} & \text{else} \end{cases}$

wordsimlinks:

$dt_{d3} : (LxW) \mapsto \mathcal{P}(LxW)$

such that $dt_{d3}(\langle l, w\rangle) = \begin{cases} \{\langle l', v\rangle \mid v \in d \;\wedge l' \in s(l)\} & \text{if } \exists d \in D : w \in d \\ \{\langle l', w\rangle\} & \text{else} \end{cases}$

where s defines “similar” link types $s : L \mapsto \mathcal{P}(L)$

such that $s(l) = \begin{cases} \{\text{"obj", "iobj"}\} & \text{if } l = \text{"obj"} \;\vee l = \text{"iobj"} \\ \{l\} & \text{else} \end{cases}$

It is rather difficult to define which types of links can be considered as similar in dt_{d3} and definitively needs further exploration, but we decide to follow a conservative approach, only defining “obj” and “iobj” to be similar.
We expect *words* to behave similar in dependency-based models and word-based models, as its effect can be understood as a reduction of the dependency-based vector space to its word-based contents. This will allow a comparison between both models, though constructed differently, but will make the dependency-based model lose the information about syntactic relations. So we expect to find a potential drop in accuracy, but a high coverage, as the transformation will reduce the numbers of dimensions drastically.
Wordlinks, by contrast, is expected to behave in a converse manner: Since all syntactic information is kept, but the transformation’s impact on the dimensionality is not that strong, we estimate the accuracy to be higher than for *words*, and the coverage to be lower.
Wordsimlinks can be regarded as trade-off between both: It is assumed to provide an increased accuracy compared to *words*, since the dependency information is not completely

ignored, but we expect it to have a slightly higher coverage than *wordlinks*, as it abstracts over a smaller number of link types.
Given for instance the dimensions W=$\{\langle$ *subj-1, fahren*$\rangle$, $\langle$*iobj, Fahrer*$\rangle$, $\langle$*obj, Fahrer*$\rangle$, $\langle$*auf, Straße*$\rangle$, $\langle$*iobj, Motor*$\rangle$, $\langle$*Fahrzeug-1, motorisiert*$\rangle$, $\langle$*auf, Radweg*$\rangle\}$[1], the *words*-transformation creates the set of new dimensions $W'_{words} = \{$*18, Straße, 886, Radweg*$\}$, the same as for the word-based model (see 3.1). *Wordlinks* leaves the dimensions unchanged as there are no dimensions that occur in the same derivational family and have the same link. *Wordsimlinks* creates the dimensions $W'_{wordsimlinks} = \{\langle$*subj-1, fahren*$\rangle$, $\langle\{$*obj, iobj*$\}$, *Fahrer*$\rangle$, $\langle$*auf, Straße*$\rangle$, $\langle$*iobj, Motor*$\rangle$, $\langle$*Fahrzeug-1, motorisiert*$\rangle$, $\langle$*auf, Radweg*$\rangle\}$, hence reduces the number of dimensions by one by aggregating "$\langle$iobj, Fahrer$\rangle$" and "$\langle$obj, Fahrer$\rangle$" as we defined "iobj" and "obj" as similar.
The transformations themselves are computed just as described for the word-based model (see 3.1). We change the *multiply* transformation scheme slightly, such that only non-zero weights contribute to the weight for a new dimension. This modification was made because we found in experiment 1 that *multiply* models suffer from their high number of zero values.

1 In the notation for dependency links, we use "-1" to represent inverse links (see section 2.1)

4 Experiments

In this chapter we report on experiments with first word-based, and then dependency-based vector space models.
For each transformed model, we subsequently characterize the effects caused by the transformation, evaluate the new models on two semantic tasks, and finally present and summarize the results.

4.1 Experiment 1: Word-based Vector Spaces

4.1.1 Data

For our experiments with word-based vector spaces, we use a word-by-word co-occurrence matrix extracted from the SdeWaC corpus. SdeWaC [Faaß & Eckart, 2013] contains a large collection of sentences from websites from the .de top-level domain [1]. The corpus was lemmatised, POS-tagged with TreeTagger [Schmid, 1994] and dependency-parsed with the MST Parser [McDonald *et al.*, 2006][2] including a MATE-Backup [Bohnet, 2010] for unknown lemmas. The 10,000 most frequent lemmas form the dimensions of the vector space, and all lemmas with a frequency over three form the target words, resulting in a matrix of the size 280,266$\times$10,000. Hereinafter it is referred to as *plain*.
A context window of $\pm$5 words within a sentence was used when extracting the co-occurrence counts. From the frequency counts we additionally constructed a matrix weighted with Positive PMI since this weighting function is considered to perform well in combination with the cosine similarity measure especially in word-context matrices, as examined by [Bullinaria & Levy, 2007]. Positive PMI (PPMI) is the same as PMI, but all negative values are set to zero. Altogether we obtain two vector spaces, each having a differently weighted matrix but identical dimensions and targets.
The matrix with frequency counts is 96.22% sparse, i.e., 96.22% of the contained values are zero. Even more sparse is the PPMI weighted matrix with 97.24% zeros.

1 Available at `http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/sdewac.html`
2 Available at `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`

For the transformation of the two vector spaces we use the collection of derivational families of DErivBase version 1.4.1[3] without the information about derivational rules.

4.1.2 Transformation

The transformed vector spaces have new basis elements, which are the same for all transformation schemes, and new weights, which depend on the transformation scheme. The targets remain unchanged.
In case of the word-based vector spaces we applied the three transformation schemes *add*, *multiply* and *avg* (see 3.1 for definitions) to the frequency-based and to the PPMI weighted matrix. Since PPMI works with marginal probabilities, we applied the transformation schemes *add* and *avg* before and *multiply* after the computation of PPMI values to the frequency-count matrix.
In word-based vector spaces no syntactic information is accessible, so the only transformation condition that can be used here is *words*.

4.1.3 Baselines

We compare our novel approach of dimension reduction to common and well-established approaches. The simplest and least costly way of dimension reduction is the technique to only consider the *top n* dimensions of a given matrix. The *top n* dimensions are the *n* most frequent dimensions (or the ones with the highest marginal sum of LMI weights, see [Padó & Lapata, 2007]). By selecting them, it is ensured that they produce a low number of zeros in the co-occurrence matrix, because they have the maximal number of co-occurrences with the given targets. Although this approach seems attractive for its simplicity, it is in fact not very sophisticated, as it discards information, that is the co-occurrence information from the *least (d-n)* dimensions (if *d* is the number of dimensions). Truncated Singular Value Decomposition (see e.g. [Cline & Dhillon, 2006]), by contrast, clearly avoids to discard any information and instead tries to approximate the original matrix and the inherent components as closely as possible. This is the reason why using (truncated) SVD for dimension reduction in distributional semantic vector spaces is a

3 Available at `http://www.cl.uni-heidelberg.de/~zeller/res/derivbase/`

common and well-proven procedure, as described in section 2.2.
We compare the characteristics of dimensionality-reduced vector spaces and their performance on semantic evaluation tasks in order to identify strong and weak points of our novel approach and to help to estimate in which cases it is can be used preferably to truncated SVD or top-n selection for dimension reduction.
For a fair comparison, all three reduction techniques should reduce the given vector space to the same size, such that *n* of *top-n* is equal to the number of dimensions in the DErivBase-transformed vector space and the parameter *k* in truncated SVD. For readability reasons the models transformed with these techniques are referred to as *top-n* and *SVD*.[4]
The following section reports on the methods and results of a quantitative and qualitative comparison with the two baselines.

4.1.4 Evaluation

4.1.4.1 Sparsity and Dimension Reduction

Table 4.1 documents the impact of DErivBase transformations and the baseline transformations on the matrix. For our research purpose it is important to consider not only the transformed matrices' size, but also their sparsity, as it allows inferences about the models' coverage. Furthermore it is interesting to consider the absolute number of non-zero values, as it allows to estimate the models' efficiency in storage. A low number of non-zero values indicates that the information captured by the model is storable in a very compact manner, whereas a high number indicates the opposite.

The DErivBase transformation causes a reduction of dimensions by over a half. All the co-occurrence information is now captured in approximately one third of the number of non-zero values of the original matrix. In comparison to the baseline models, the DErivBase-transformed models provide the most compact representation.
The original model with frequency counts is 96.22% sparse, PPMI weights raise the sparsity to 97.24%. We generally observe that PPMI weighted matrices are sparser than according matrices with raw frequency weights, which is due to the inherent characteristics of the PPMI measure: Zeros that already occurred in the frequency matrix stay zeros in the

4 *SVD* was computed with MATLAB's (version 8.3) built-in function svd(X).

Model			Frequencies		PPMI Weights	
		Dimensions	Sparsity	Non-zero values	Sparsity	Non-zero values
plain		10,000	96.22%	105,844,861	97.24%	77,422,002
words	*add*	5,291	95.16%	71,771,750	96.05%	58,574,053
	multiply	5,291	99.17%	12,307,965	99.37%	9,342,191
	avg	5,291	95.16%	71,771,750	96.05%	58,574,053
SVD		5,291	0.00%	1,482,887,406	0.02%	1,453,229,658
top-n		5,291	94.44%	82,448,540	96.04%	82,411,138

Table 4.1: Dimensionality, sparsity reduction, and non-zero values of word-based models

PPMI weighted matrix, since the joint probability is zero in these cases. Additional zeros originate from those cases where the joint probability of a context and a target is smaller than the product of their individual probabilities, such that the log of the quotient, which is <1, turns <0 and is set to zero in PPMI. Joint probabilities generally tend to get extremely small in vector spaces with high sparsity.

For both weighting schemes, *add* and *avg* reduce the models' sparsity by approximately 1%. *Multiply* by contrast increases the sparsity up to over 99%, due to the fact that a product is zero as soon as one factor is zero. Here the factors are co-occurrence counts for a target and a word from a certain derivational family. If only one member of the derivational family is assigned a zero co-occurrence count, the count for the whole derivational family in the transformed matrix is also zero. For *add* and *avg*, zero counts are easily "repaired" and a target only needs to co-occur with one member of a derivational family to get assigned a non-zero count for the whole derivational family.

Note that the SVD baseline does not permit a perfectly proper comparison here, as it was performed on a matrix containing only the targets needed for our evaluation tasks due to computational efficiency reasons. Nevertheless, the zero percent sparsity of the SVD-transformed model is striking. Assuming an equal distribution of zeros over the targets, a projected sparsity to the numbers of all targets would still not exceed 5% in both models. A detailed analysis and explanation is provided in the discussion in chapter 5.

The top-n reduction technique does not reduce the sparsity to such a high extent as SVD does, but the reduction is still remarkably higher than the ones caused by the DErivBase transformations. This effect is easily explained, as selecting the dimensions with the highest marginal frequencies (or PPMI weights) decreases the likelihood that these dimensions are filled with zero values.

4.1.4.2 Word Similarity

Lexical semantic vector spaces are most widely applied for indicating semantic similarity between words, by measuring the similarity of the corresponding vectors.
For German, the Gur350 dataset [Gurevych, 2005][5] is well-suited for evaluating the ability of a vector space to model word similarity. The data set is a collection of 350 German word pairs with a human annotated [6] discrete relatedness score between 0 (meaning fully unrelated) and 4 (fully related) and the standard deviation of the annotated scores. The pairs cover all types of semantic relatedness, also across word classes. The following lines are extracted from the data set and each contain a word pair, their annotated similarity score and the standard deviation.

```
leben Tod 3.25 1.38873015
Linguistik Wissenschaft 3.5 0.534522484
Studium studieren 4 0
viel schreiben 0.375 0.51754917
```

By evaluating our models on this data set, we aim to detect how well the DErivBase-transformed models, in comparison to the *SVD* and *top-n* baseline models, are able to retain and predict semantic relatedness.
For this task we measure the Pearson correlation of the models' predicted similarity scores with the human gold annotation for each word pair. Pearson's r is a correlation coefficient that measures linear correlation or dependence between two variables, giving values in the (inclusive) interval between +1 and -1 (+1 stands for fully positive correlation, -1 for fully negative correlation). In our case we compute the Pearson correlation r for the pairs (X_i,Y_i) where X_i is the gold annotation score for the ith pair in the data set, and Y_i is the

5 Available at `www.ukp.tu-darmstadt.de/fileadmin/user_upload/Group_UKP/data/datasets.zip`
6 Eight annotators, inter-annotator agreement: 0.69

model's cosine similarity score for the same pair, as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

$$\text{with sample means } \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \text{ and } \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \quad (4.1)$$

Coverage is measured by the proportion of word pairs that is assigned a similarity score > 0 by the examined model, as zero scores imply that the model is not able to make a prediction for this pair.

For our word-based setting with three transformed models, the plain model and the two baselines, we observe the results listed in table 4.2. The highest figures each are marked in bold.

Model		**Pearson's** *r*	**Coverage**
plain		.656	.840
words	*add*	.660	.840
	multiply	.630	.840
	avg	.663	.840
SVD		.661	.840
top-n		**.666**	.840

Table 4.2: Correlation and coverage of word-based models for the word similarity task

The plain model obtains a correlation of 0.656. All other models maintain the plain model's coverage, and simultaneously manage to increase the correlation (except for the multiply-words-transformed model). Of the DErivBase-transformed models, the *avg-words*-transformation causes the strongest rise in correlation up till 0.663.
SVD scores minimally lower (-0.002) than the *avg-words*-transformed model, but *top-n* yields best overall results with a correlation coefficient of 0.666.
According to the method of measuring significance introduced by [Fisher, 1925], none of the transformed vector space produces significantly different similarity scores for the word similarity task at p = 0.05 compared to the *plain* model.

4.1.4.3 Synonym Choice

The second semantic evaluation task deals with synonyms. Synonymy is a specific type of semantic similarity: if two words are maximally similar, they are synonyms to each other. In a way, the ability of predicting synonymy is already tested in the word similarity task in the preceding section, since synonymy is included in the data set's various types of semantic relatedness. However, it is not only important for applications in natural language processing to correctly predict a degree of similarity for a isolated pair of words, but also to properly classify and rank the similarity of several word pairs in relation. For example, a semantic model should not only detect that *house* and *garage* are quite related and similar, but also that *villa* and *house* are even more similar in their semantics.
For English, this ability is often evaluated on the TOEFL synonym choice task [Landauer & Dumais, 1997]. In the same manner, we use the German equivalent, the "Reader's Digest Word Choice Problems for German"[7] [Wallace & Wallace, 2005]. The data set contains 984 tasks, with one target and four candidates each, for instance:

```
Flora
a) kleines Insekt
b) Tierreich
c) Pflanzenwelt
d) Blütenpracht
```

The task is to select the synonym for a given target from the four possible candidates (here candidate c)). Many tasks involve phrases instead of single words as targets and answer candidates. There are two alternative ways of evaluating a system's performance on the tasks introduced by [Mohammad *et al.*, 2007]:

1. Alpha: The system only processes tasks, if the scores for all four candidates are greater than zero, assuming that the system is not able to make a proper prediction with a solid base of information if not all four answers are covered.

7 Available at
`http://www.ukp.tu-darmstadt.de/fileadmin/user_upload/Group_UKP/data/germanWordChoiceProblems.txt`

2. Beta: The system processes a task as soon as one of the candidates' score is greater than zero, assuming that a single score greater than zero provides already enough information to make a choice between the candidates.

To put it in other words, the alpha version allows merely safe decisions with a high threshold of information needed to support the chosen candidate, whereas the beta version already goes with "riskier" decisions based on less information to enable a wider coverage. Coverage is measured as the proportion of tasks that gets processed (depending on alpha or beta version). Each correctly answered task contributes to a score for each system (+1), where ties give partial scores: A 4-way tie contributes one quarter to the score (+.25), a 3-way tie one third to the score (+.$\overline{3}$), a 2-way tie one half (+.5). On the basis of this score, the accuracy is calculated (=score/covered). Thus, the accuracy represents the share of the covered tasks that are answered correctly, that is when the correct answer of the four possible answers is chosen.
As synonyms are characterized by maximal semantic similarity and we can assess semantic similarity in semantic vector spaces by vector similarity, we can detect synonyms from a range of pairs by choosing the pair with maximal vector similarity. Like that, we try to identify the correct candidate for a given target from the word choice data set by choosing the candidate that has the highest target-candidate similarity score.
It is somewhat problematic that candidates and targets occur as phrases of multiple words, for our model is not designed for assessing the similarity of phrases. Following the approach of [Padó & Utt, 2012], we solve this problem by systematically choosing the maximal similarity score from all the words contained in the phrases.

The plain model, the transformed models, and the baselines yield the results documented in Table 4.3.

Alpha evaluation: The *plain* model achieves a score of 252, resulting an accuracy of 70% and a coverage of 36.6%. Our best DErivBase transformations manage to increase the score and the number of covered tasks by one each, raising accuracy and coverage by approximately 0.1%. *add* and *avg* score equally well, whereas *multiply* performs significantly worse. SVD loses two points of the score whilst covering the same number of tasks, so in comparison with the *plain* model, it suffers a slight loss in accuracy. The *top-n* baseline has the lowest accuracy and coverage of all models (except *multiply*).
It is worth noticing that our *add-words*-transformed model performs best in accuracy, even

Model		Alpha				Beta			
		Score	Covered	Acc.	Cov.	Score	Covered	Acc.	Cov.
plain		252	360	.700	.366	**379**	**587**	**.646**	**.597**
words	*add*	**253**	**361**	**.701**	**.367**	344	**587**	.635	**.597**
	multiply	170	281	.605	.286	320	572	.559	.581
	avg	**253**	**361**	**.701**	**.367**	375	**587**	.639	**.597**
SVD		250	360	.694	.366	368	**587**	.627	**.597**
top-n		246	358	.687	.364	366	**587**	.624	**.597**

Table 4.3: Accuracy (Acc.) and coverage (Cov.) of word-based models on the word choice task

better than the double-sized *plain* model.
Beta evaluation: As expected the number of items and scores are generally higher in the beta evaluation than in the alpha version. Simultaneously, the accuracy for all models is lower, as it makes "riskier" predictions than before (see 4.1.4.3). The *plain* model covers 59.7% of all tasks, with an accuracy of 64.6%. The *add-words-* transformed model's accuracy is now, in contrast to the alpha evaluation, lower (-1.7%) than the *plain* model's accuracy. The *multiply* performs even worse. Now it is *avg* to perform best of all DErivBase-transformed models, but it fails in improving the *plain* models accuracy decreasing it by 1.1% to 63.9%. Still it scores better than both *SVD* and *top-n* baseline models, which achieve an accuracy of 62.7% and 62.4% respectively.
Again, a significance test revealed that none of the transformed vector space produces significantly different similarity scores for the synonym choice task at p = 0.05 compared to the *plain* model.

To summarize the experiments for the word-based vector spaces: The DErivBase transformation reduces the plain vector space's dimensionality by almost half, yet the observed reduction in sparsity is only marginal. However, the DErivBase-transformed models stand out for their compact storage of co-occurrence information.
For the word similarity task, the correlation is found to be highest for the *top-n* baseline, but still the *avg*/*add*-transformation models performs better than *plain*. For the synonym choice task (alpha evaluation), the DErivBase-transformed models actually perform best.

4.2 Experiment 2: Dependency-based Vector Spaces

4.2.1 Data

For the dependency-based vector space, we use a matricization of *DM.de* [Padó & Utt, 2012][8], the German equivalent of the Distributional Memory proposed by [Baroni & Lenci, 2010] (see section 2.1).
From the *DM.de* tensor, we build a word-$\langle$link,word$\rangle$ matrix with lemmas and their POS-tags as targets and pairs of dependency links and words as dimensions. To increase the readability of this section, we call lemmas with their POS-tag "words". The resulting matrix has 3.5M targets and 16M dimensions. As we only need a limited number of targets in our evaluation tasks, we cut our matrix down to these specific target words, such that the size is reduced to 7,946$\times$5,766,661. This matrix, in the following called *plain*, is 99.96% sparse.
In a preliminary experiment, we found the best performance when using the top 50k-200k dimensions and will therefore use a 50k dimension matrix, hereinafter referred to as *top50k*, as our starting point (compare tables A.1 and A.2 for the results of a test series with varying dimensionality). The matrix was built by selecting the 50k word-link-pairs with the highest marginal sum of LMI scores in the *DM.de* tensor during the matricization. The reduction of dimensions by roughly 99% involve only a tiny reduction of sparsity to 98.95%.
The higher a co-occurrence matrix's dimensionality, the more prone to sparsity the matrix is. With a large number of dimensions, we expect dimensionality reduction to have an even greater effect on the vector space as it had on our word-based models.

4.2.2 Transformation

The same variation of transformation schemes as in experiment 1 are used for the dependency-based approach: *add*, *multiply*, and *avg*. In contrast to the word-based transformation experiments, a wider range of transformation conditions is accessible: *words*, *wordlinks*, and *wordsimlinks* (see 3.2 for definitions and examples).

8 Available at `http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/dmde.html`

4.2.3 Baselines

We simply use the same baseline techniques for dimension reduction as for the word-based experiments: truncated SVD with *k* equal to the numbers of the DErivBase-transformed vector space and *top-n* with *n* equal to *k*.
Note that we need distinct baselines for each transformation condition we inspect, since they produce differently-sized vector spaces.

4.2.4 Evaluation

4.2.4.1 Sparsity and Dimension Reduction

A list of the transformed models and their dimensionality, sparsity and number of non-zero values is presented in table 4.4.

Model		**Dimensions**	**Sparsity**	**Non-zero values**
plain		5M	99.96%	14,634,613
top50k		50k	98.95%	3,685,608
words	*add/multiply/avg*	16,316	98.41%	1,823,167
SVD		16,316	0.00%	0
top-n		16,316	98.05%	2,157,467
wordlinks	*add/multiply/avg*	39,569	98.85%	3,188,866
SVD		39,569	0.00%	0
top-n		39,569	98.80%	3,325,995
wordsimlinks	*add/multiply/avg*	38,738	98.85%	3,136,376
SVD		38,738	0.00%	0
top-n		38,738	98.80%	3,293,472

Table 4.4: Dimensionality, sparsity reduction, and non-zero values of dependency-based models

The *top50k* model's dimensionality was reduced to a third by the *words* transformation and to c. 80% by *wordlinks* and *wordsimlinks*. As we observed in the word-based experiment, the DErivBase-transformed models represent the data in a most compact form.
The *plain* model has an extremely high sparsity of 99.96%. This figure indicates that barely 14,6M values in the full 7946x5766661 matrix are not zero. Selecting the top 50k dimensions,

thereby reducing it to 1% of the plain vector space's size, produces a matrix which is still 98.95% sparse. The reduction of 1% is explained by the fact that those selected 50k dimensions have the highest marginal LMI weights in the vector space, thus the probability to produce zero values is minimized.
Of all DErivBase-transformed models, the *words*-transformations cause the highest reduction of both dimensions (by over 67%) and sparsity (by 0.5%) from the original 50k vector space. As the *wordlinks* and the *wordsimlinks* transformations conditions induce a more selective clustering of dimensions, their dimensionality and sparsity reduction is lower. Each distinct transformation scheme results in an equal sparsity and dimensionality for all three conditions, since a modification was applied to the *multiply* variant for the dependency-based transformation (see 3.2).
The differences in sparsity reduction between the DErivBase-transformed models and the two baseline models is the same as observed for the word-based experiments (see 4.1.4.1): The top-n transformation reduces the sparsity more strongly than all DErivBase transformations do, but SVD reduces it even more, by extinguishing all zero values.

4.2.4.2 Word Similarity

Just as in experiment 1, we evaluate our dependency-based models' ability to represent word similarity and word relatedness on the Gur350 data set.
The Pearson correlation coefficient and coverage for all syntactic-based models are listed in table 4.5.

The *plain* model's coverage of 87.7% is higher than word-based model's, whereas the correlation coefficient is only 0.386, which is far lower. *Top50k* improves the correlation, but slightly impairs the coverage.
We observe that the *add*-transformation scheme performs always best compared to the other transformation schemes. Combined with the *words* condition, its correlation coefficient is 0.001 higher than the *plain* model's whilst having a far higher coverage. Interestingly, its overall performance is ranked between the baseline models *SVD* and *top-n*: It tops the *SVD* baseline by 0.006 concerning the correlation, but stays lower concerning the coverage. Nevertheless it has a higher coverage than *top-n*, but a lower correlation. The *top-n* baseline model has clearly the highest correlation of all models.

Model		**Pearson's ρ**	**Coverage**
plain		.386	.877
top50k		.403	.863
words	*add*	.387	.929
	multiply	.144	.929
	avg	.244	.929
SVD		.381	.931
top-n		**.410**	.860
wordlinks	*add*	.403	.869
	multiply	.199	.869
	avg	.344	.869
SVD		.400	**.940**
top-n		.404	.863
wordsimlinks	*add*	.402	.871
	multiply	.184	.871
	avg	.339	.871
SVD		.395	.931
top-n		.404	.862

Table 4.5: Correlation and coverage of dependency-based models for the word similarity task

Combining the *add*-scheme with the *wordlinks* transformation condition, similar tendencies emerge: *add-wordlinks* has the same correlation as the original *top50k* model, but still performs slightly better in terms of coverage. At the same time, it still has a slightly lower correlation than the *top-n* baseline model and lower coverage than the *SVD* model, but higher correlation than *SVD* and higher coverage than *top-n* reversely.
The *wordsimlinks* results are ranged in between the results for*words* and *wordlinks*, since the transformation itself can be seen as a compromise between these two transformation conditions. The *wordsimlinks* conditions for the transformation takes the links into account, but is not as strict as *wordlinks*, because it does not differentiate between the links "obj" and "iobj" (see 3.2).
To sum up these results, we found that the *add-wordlinks* transformation yielded the best correlation of the transformation models, and *add-words* the best coverage. The *SVD* baselines model the best overall coverage and the *top-n* baselines the best overall accuracy.
As in the word-based experiment, none of the transformed vector spaces produces significantly different similarity scores for the word similarity task at p = 0.05 compared to the *plain* model.

4.2.4.3 Synonym Choice

For the synonym choice task we proceed in the same way as in experiment 1. The results are listed in table 4.6.

Alpha evaluation:
The *plain* model scores 226, with 409 tasks covered. With those results it achieves a higher coverage but a lower accuracy than the word-based *plain* model. *Top50k* yields a slightly higher accuracy of 58.3% and lower coverage of 41.2%. None of the other models achieves an equally high accuracy.
Across all DErivBase transformation schemes, *add* works best again: *Add-words* increases the coverage up till 49.2%, yet losing some accuracy, since the growth of the numbers of covered tasks is much larger than the improvement of the score. *Add-wordlinks* by contrast has a higher accuracy, but a lower coverage. This observation is consistent with the results from the semantic similarity task in section 4.2.4.2. Also, *SVD* again manages to increase the coverage best. Although the *top-n* baseline model scores lower and covers less items than most other models its accuracy is one of the highest.

Model		**alpha**				**beta**			
		Score	**Covered**	**Acc.**	**Cov.**	**Score**	**Covered**	**Acc.**	**Cov.**
plain		226	409	.553	.416	438	831	.527	.845
top50k		236	405	**.583**	.412	433	820	.528	.833
words	*add*	255	484	.527	.492	424	873	.486	.887
	multiply	211	484	.436	.492	347	873	.397	.887
	avg	213	484	.440	.492	357	873	.409	.887
SVD		297	544	.546	.553	450	**906**	.497	**.921**
top-n		224	402	.557	.409	413	815	.507	.828
wordlinks	*add*	241	426	.566	.433	422	830	.508	.843
	multiply	199	426	.467	.433	353	830	.425	.843
	avg	217	426	.509	.433	390	830	.470	.843
SVD		**302**	**549**	.550	**.558**	448	**906**	.494	**.921**
top-n		235	404	.582	.411	432	819	.527	.832
wordsimlinks	*add*	239	429	.557	.436	419	832	.504	.846
	multiply	193	429	.450	.436	338	832	.406	.846
	avg	214	429	.499	.436	386	832	.464	.846
SVD		299	**549**	.545	**.558**	**454**	904	.502	.919
top-n		235	494	.476	.502	433	819	**.529**	.832

Table 4.6: Accuracy (Acc.) and coverage (Cov.) of dependency-based models on the word choice task

When examining the results for the other DErivBase transformation conditions, we recognize that choosing the *wordlinks* or *wordsimlinks* transformations instead of *words* improves the accuracy and deteriorates the coverage: Using *wordlinks* results in a higher accuracy than the *SVD* models', but lower than *top-n*s', and the coverage is lower than the *SVD* models' coverage, but higher than the *top-ns*'. *Wordsimlinks*' accuracy is increased such that it performs better than the two baselines in terms of accuracy but not in terms of coverage.
Beta evaluation:
The comparison of alpha and beta evaluation methods corresponds to the comparison we made in experiment 1: The accuracy is generally lower, whilst the coverage is generally higher. The *plain* model now covers 831 of the 984 pairs and scores 438, *top50k* covers 831 and scores 433. Once again, the *add*-transformed models perform best across all DErivBase transformation schemes. The *add-words*-transformed vector space's accuracy is approximately 8% lower than the *top50k* model's, but the coverage roughly 5% higher. In contrast to the alpha evaluation, also the score is lower. Again, we observe that *SVD*'s strength is the coverage, whereas it is the accuracy for the *top-n* baseline. If we use other transformation conditions instead of *words*, we slightly increase the accuracy but decrease the coverage.
To sum up our word choice evaluation, the *top-n* model's accuracy and the *SVD* model's coverage are hard to outscore. Nonetheless, our *add-words* model almost reaches the highest coverage and *add-wordlinks*'s accuracy is not far behind.
Again, none of the transformed models produces significantly different similarity scores for the synonym choice tasks at $p = 0.05$ compared to the *plain* model.

The tendency we observe across both evaluation tasks is that the original model's coverage is easy to increase, whereas the accuracy or correlation is already quite high. The model that manages to increase the coverage best in most cases is the *SVD* model, whereas it is *top-n* that increases the accuracy or correlation best. The DErivBase-transformed models usually come off somewhere in between. The strength of the *word*-transformation is found to be the coverage, the *wordlinks*' strength is the accuracy. It is the *add* transformation scheme that works best with the LMI weights.

5 Discussion

5.1 Word-based vs. Dependency-based Models

Word-based and dependency-based semantic vector space models have different characteristics: word-based spaces are rather small, usually less sparse and in our case already have a comparably high coverage. Dependency-based spaces, in contrast, usually are much larger and more sparse, but achieve high accuracy because of their fine-grained definition of contexts. We discuss tendencies emerged during the evaluation jointly, and in this manner find advantages and disadvantages of the dimension reduction with DErivBase. First, we examine the differences in the performance on our evaluation tasks between the two plain models, in order to subsequently assess the performance of the transformed models. In both evaluation tasks, we examine the models' ability to predict lexical semantic similarity or relatedness, both between two words ("How similar are words *x* and *y*?") and word pairs ("Are *x* and *y* more related than *y* and *z*?"). Coverage is measured similarly in both tasks, and it expresses the extent to which the model is able to make predictions. Accuracy measures the quality of this prediction in the synonym selection task. Likewise, the correlation for the word similarity task expresses the quality of the model's prediction, which allows it to be also interpreted as a measure of accuracy. So for the further discussion, we include the correlation measure for the word similarity task when we speak of "accuracy".
We observe that the coverage of the synonym choice task is relatively low across all models. Only 594 (of 984) target words (i.e., the words that the synonyms have to be selected for) are found in the word-based plain model's targets, of which 587 tasks are covered (i.e., predictions >0 are made for at least one candidate in beta evaluation version, see Table 4.3). This low coverage can be explained by the fact that the synonym choice tasks contains many special, rarely-used words or loan words, for example "julklapp", describing a Swedish Christmas-present tradition. This is why not all of these particular words can be found in the word-based vector spaces targets. The dependency-based plain model does not struggle at this point and covers 939 of the task's target words, since it contains a larger number of targets (3.5M) than the word-based model. But it makes predictions

for only 831 tasks (see Table 4.6). 108 tasks get lost because all of their target-candidate pairs are assigned the similarity score of zero. Here it is clearly the sparsity, i.e., too many zero values in the target vectors, that impairs the coverage. The potential of increasing the coverage by dimension reduction is hence much larger for the dependency-based vector space.

Analysing the alpha evaluation method, it is striking that the dependency-based model achieves an accuracy of a similar magnitude as the word-based model. Although containing more information in a larger co-occurrence matrix, the dependency-based model is thus not able to make a more accurate ("safe") prediction of synonyms. For the beta method, the coverage of the dependency-based model is much higher, but the accuracy is even lower than the one achieved by the word-based model. This indicates that the dependency-based model makes many "risky" predictions which are not qualitatively good.

For the word similarity task, the coverage of both *plain* models is almost equal. Again, the word-based plain model's coverage is due to the relatively low number of targets, whereas the dependency-based plain model struggles with a high number of zero similarity scores. The correlation of the dependency-based plain model with the human annotated scores is considerably lower than the word-based model's. This demonstrates that the increased number of dimensions and a finer-grained specification of contexts do not help to predict semantic similarity here, caused by the high number of zero values in the highly sparse co-occurrence matrix.

In summary, we found empirical evidence that the performance on the semantic similarity tasks of the dependency-based plain model is much more affected by sparsity than the word-based model. The dependency-based model does therefore not profit from its linguistically richer and deeper way of representing word meaning. This finding is important to take into account for the following analysis of the transformed models.

When comparing the word-based plain models with the transformed models in terms of their performance on the Gur350 data set, we see that an improvement in coverage is simply not feasible, because it is merely dependent of the number of targets covered which does not change throughout our transformations. However, all transformed models manage to increase the accuracy here. For the synonym choice task, small improvements of the coverage by the DErivBase transformations are noticeable. The DErivBase-transformed models *add-words* and *avg-words* also cause a slight rise in accuracy. To sum up, reducing the word-based model's dimensionality does generally not influence the coverage, but leads to a slightly impaired or improved accuracy, depending on task and transformation

method. This is due to the fact that the plain model's coverage is already high and that the influences of sparsity are only minor.
Next we take a look at the dimension reduction's influence on the dependency-based models. When comparing the results for the word similarity task, it is striking that only the *top-n* model could improve the accuracy. However, the coverage is improved by all models except *top-n*. For the synonym choice task, none of the models improves the accuracy, but *top-n* comes closest. Yet it is not able to increase the coverage, which all other models are. Furthermore it is worth mentioning that *SVD* always achieves the best coverage.
In contrast to the effects on the word-based model, reducing the dimensionality of the dependency-based space generally leads to a considerably increased coverage, but an only marginally improved accuracy. This observation can be explained by the fact that the dependency-based plain model's high accuracy, based on the fine-grained distinctions between dimensions, cannot be optimized any further by dimension reduction techniques, for they do not refine the dimensions, but instead coarsen them. However, they do reduce the sparsity's comparably high influence on the dependency-based model by reducing the percentage of zero values in the matrix. This is indicated by the increased coverage.

5.2 DErivBase Transformation in Comparison

Now that we contrasted the difference in the dimension reduction's influence on both types of vector spaces, we focus on contrasting the differences between the dimension reduction techniques.
As stated above, *top-n* performs exceptionally well in improving the accuracy, in particular when predicting semantic relatedness. By contrast, it is *SVD*'s strength to improve the coverage, in particular when detecting synonyms. Recalling its functionality (see section 2.2) we are able to explain this observation: The truncated SVD matrix approximates the original matrix and the inherent dependencies, but in a lower-dimensional space. Because of the approximation, the transformed matrix does not contain zero values any more, but values very close to zero. This effect is illustrated by the histograms in Figure A.3 in the appendix that show the distribution of similarity scores for the word pairs of the word similarity task. One can clearly see how the *SVD* model approximates the *top50k*'s distribution closest. Although the zero values in the matrix and thereby zero similarity

scores were removed, those similarities mainly remain <0.1. This is how *SVD* solves the issue of sparsity, but the disadvantage of a fully-filled matrix is the vast storage space it takes. Our approach however, produces much more efficiently storable matrices which are still sparse but contain the information from the plain model in a more compressed manner.

When comparing the similarity scores for the semantic relatedness task (see Figure A.3), the dimension reduction with DErivBase (here we examine the *add-words* setting), makes the overall biggest change to the distribution, as it reduces the number of similarity scores <0.1 drastically and simultaneously increases the number of similarity scores >0.1. The scatter plots for the same tasks (see Figure A.5) additionally underline this change: Even if the correlation is not directly visible, we still distinguish a slight rise in the similarity scores and a reduction of zeros. On the one hand, a diminished number of zero-similarity scores is really good for a gain in coverage, but on the other, a general rise in all similarity scores impairs the accuracy, at least in the dependency-based vector space. In the word-based vector space, these effects are reduced by the PPMI weighting after the transformation with raw frequencies (see Figure A.4) which normalizes the raised values.

In contrast to the two baseline transformations, *top-n* and *SVD*, the DErivBase transformation does not show one-sided strengths. *Top-n* and *SVD* tend to have a favourable effect either on the accuracy or the coverage, respectively. Our DErivBase transformation performs somewhere in between: By tendency its coverage is better than *top-n*'s, and its accuracy is better than *SVD*'s.

We observe that the *avg* scheme works best for word-based models and the *add* scheme for dependency-based. A possible explanation is that the cell values in the dependency-based model's matrix are generally so small, that the highest growth of the values caused by *add* helps best. This is not needed for the word-based model, where the values are not that small (and neither the similarity scores, see histograms) and an averaging, more subtle change is sufficient. The *multiply* scheme is outperformed in all tasks, because it makes the similarity scores too extreme, due to the characteristics of multiplication.

The different transformation conditions in fact show the characteristics that we expected (see section 3.2) with *words* having the highest coverage and lowest accuracy among the DErivBase models, *wordlinks* having the least coverage and highest accuracy, and *wordsimlinks* ranged between them.

Concerning the word-based models, *avg-words* is in fact a good alternative to *plain* with half the number of dimensions and an equal or better performance on semantic similarity tasks. For the dependency-based model the DErivBase-transformation is not strongly recommended because it impairs the accuracy[1]. Using a DErivBase-transformed model instead of *plain* in an application, would mean that one had to exchange higher coverage for lower accuracy, which is not really attractive. Also, we notice that the transformation with DErivBase achieves generally better results in the semantic relatedness task than in the synonym choice. Interestingly this effect was also observed by [Padó *et al.*, 2013] in their experiments with derivational smoothing. The synonym choice tends to require a finer-grained differentiation of similarity scores, which the DErivBase-transformed models are less able to represent. Thus our transformation is rather recommended to use in an application which resembles more the semantic relatedness prediction than the synonym choice.

5.3 Future Work

From the results in Table 4.5 we learn that it is essential to use the information about dependency links for an at least maintained accuracy, so transformation conditions like *wordsimlinks* and *wordlinks* are preferable, even though their coverage is low compared to the baselines. This indicates that we need to find another way to refine our transformation condition, such that we achieve both at least a maintained accuracy and a further increased coverage.

One approach is to further elaborate the way how we use DErivBase to form the new dimensions. When analyzing the synonym choice results, we find it striking that particularly the newly-covered tasks (i.e., the tasks that were not processed by *top50k*, but by the DErivBase-transformed models) are answered incorrectly by *add-words*. We observe that the larger a derivational family, the more often it is involved in these tasks and decreases the accuracy. This correlation is not as strong for the involvement in correctly answered tasks (see Figure A.7). To get an impression of the characteristics of a large cluster, we take a closer look at the DErivBase family no. 4 (see Figure A.6 in the Appendix).

It is obvious that clusters of this size contain more noise than smaller ones. The example cluster contains words with both the morphological roots "recht" and "richt". In some cases, they might be derivationally related, but "unrichtig" ("incorrect") is clearly not

1 Here we have to take into account that DErivBase could not work under the same conditions as the word-based models concerning the vector space weights, as we observed that PPMI weighting of the frequencies after the transformation works best. Now simply calculating (building sums of probabilities) does mathematically not make a proper sense.

strongly related to “Unterricht” (“lessons”), neither to “entrechtete” (“disenfranchised”). Here, the cluster fails at capturing the disambiguity of the root “richt” and the distinction to the root “recht”. Using clusters of this size and quality for the DErivBase transformation clearly leads to overgeneralization and therefore to a loss in accuracy. For a more prudent use of DErivBase we propose to make use of the additional information that DErivBase authors [Zeller *et al.*, 2013] provide. In another format of the resource (“DErivBase-v1.4-rulePaths.txt”) they list the derivational rule paths that connect each two lemmas of a derivational family. A confidence score, expressing the degree of derivational relatedness of a pair of lemmas, is easily calculated by the multiplicative inverse of the rule path’s length. On the basis of this confidence score we could then define another transformation condition that only admits two dimensions to be integrated into the same new dimension if the confidence score of their derivation is larger than a certain threshold[2]. Alternatively, one could exclude all derivation paths produced by certain derivational rules which were proven to be unreliable or produce inaccurate clusters. For instance, the exclusion of rule “dVV12.1” could have improved the above example cluster as it connected the two roots “recht” and “richt”: “berechtigen_Ven dVV12.1> berichtigen_Ven”.
If the coverage of a link-sensitive model with one of these more prudent DErivBase transformations suffers from a resulting low coverage, one could then proceed to define equivalence classes of links, like we did for *wordsimlinks*, to achieve a higher degree of abstraction over link types and balance accuracy and coverage. Also, one could additionally apply smoothing techniques like derivational smoothing ([Padó *et al.*, 2013]) to low-accuracy models.

2 One would then have to deal with words occurring in more than one new dimension, which does not allow us to use the concept of equivalence classes.

6 Conclusion

With this thesis, we introduced a new approach to dimension reduction in semantic vector spaces that uses derivational information. We formally defined the DErivBase-based transformation methods, applied them to two different types of semantic vector space models, word-based and dependency-based, and evaluated them on two semantic tasks. After all, we found that our DErivBase transformation succeeds in keeping up with common dimension reduction techniques like truncated SVD. In our word-based vector space, it achieves the highest accuracy, and in the dependency-based it particularly improves the coverage. It does not completely eliminate the negative effects of sparsity, but represents information in a more compact form by abstracting from word forms to word families, thereby reducing the vector space's size by over a half. Its matrices are more efficient to store than both SVD-reduced and top-n reduced matrices. At the same time, it outperforms the former's accuracy and the latter's coverage.

It definitively stands out with its low computational complexity and linguistically sound justification. Practically, it does not require a ready-built co-occurrence matrix, but can directly be integrated during construction. It is easily parametrized and highly adaptable to individual needs, since its performance is controlled by the choice of transformation condition and scheme.

In the previous chapter we suggested a refined use of DErivBase families in future work to avoid an impaired performance due to noise in very large families that cause overgeneralization. We have demonstrated that the approach of dimensionality reduction in semantic vector spaces using derivational information is definitively worth further consideration and examination.

A Appendix

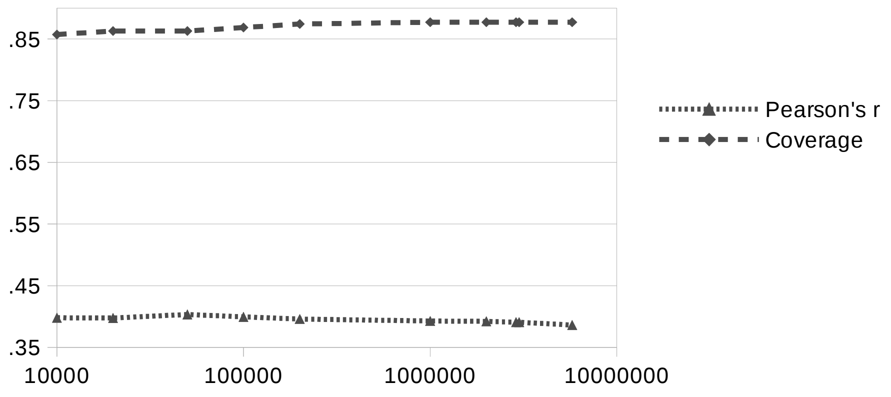

Figure A.1: Gur350 results for *top-n* with varying *n* in dependency-based vector space: Pearson's *r* and Coverage in dependency of *n*

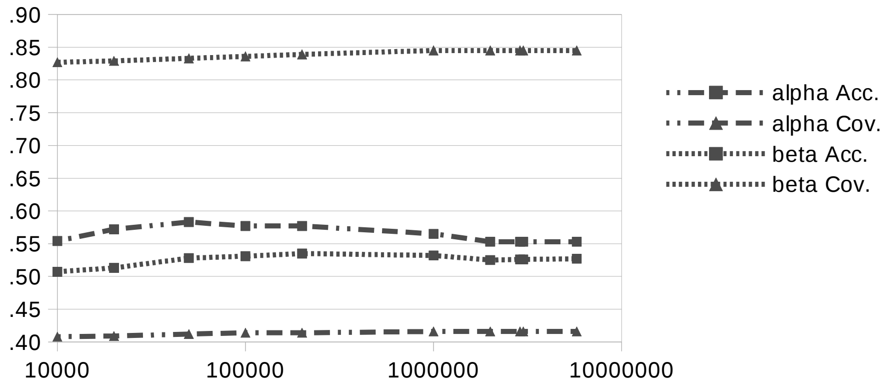

Figure A.2: Synonym choice results for *top-n* with varying *n* in dependency-based vector space: Accuracy and Coverage in dependency of *n*

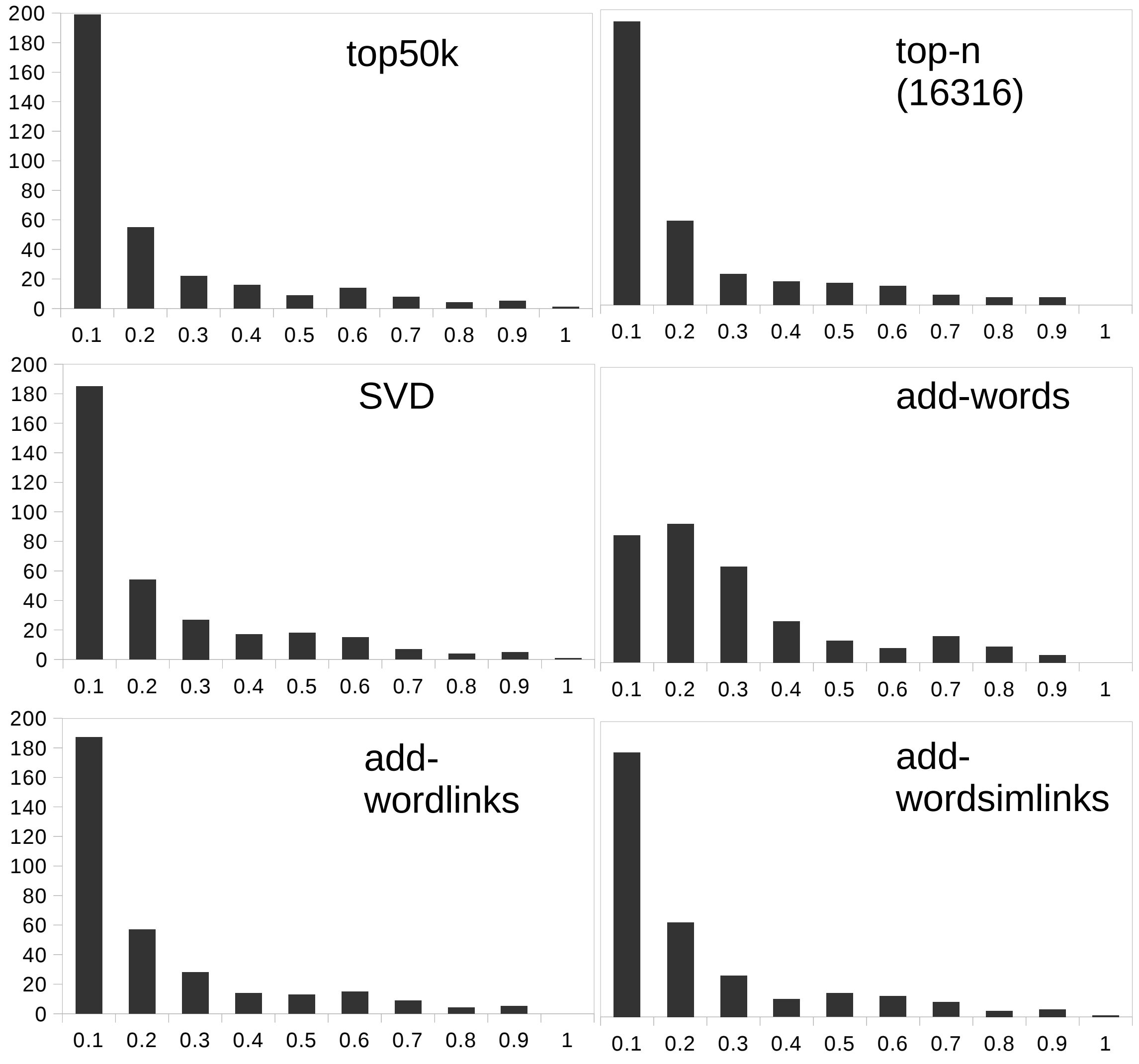

Figure A.3: Similarity score histogram for dependency-based models on Gur350 data

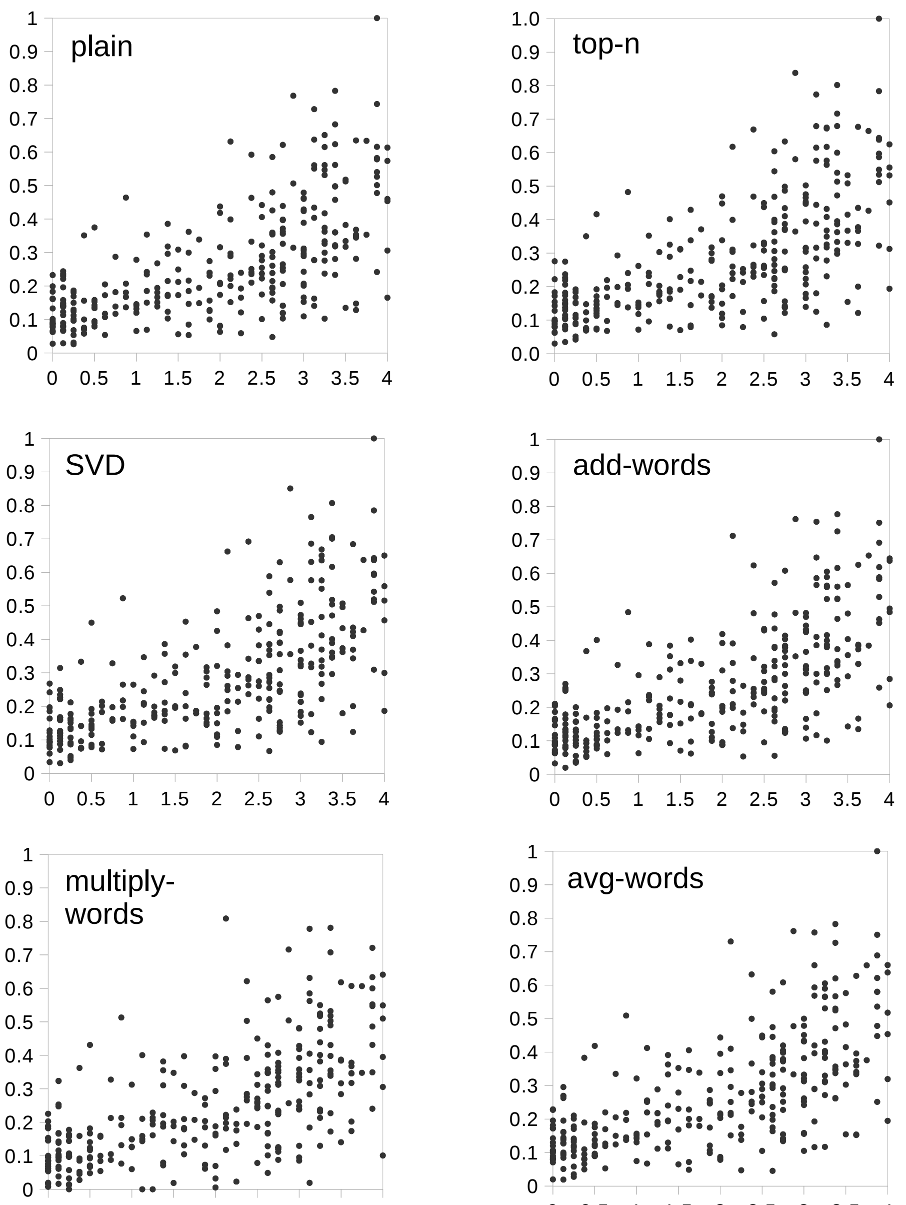

Figure A.4: Scatter plots for word-based models' (PPMI weighted) similarity scores on Gur350 data set. Annotated scores on the x-axis, predictions on the y-axis.

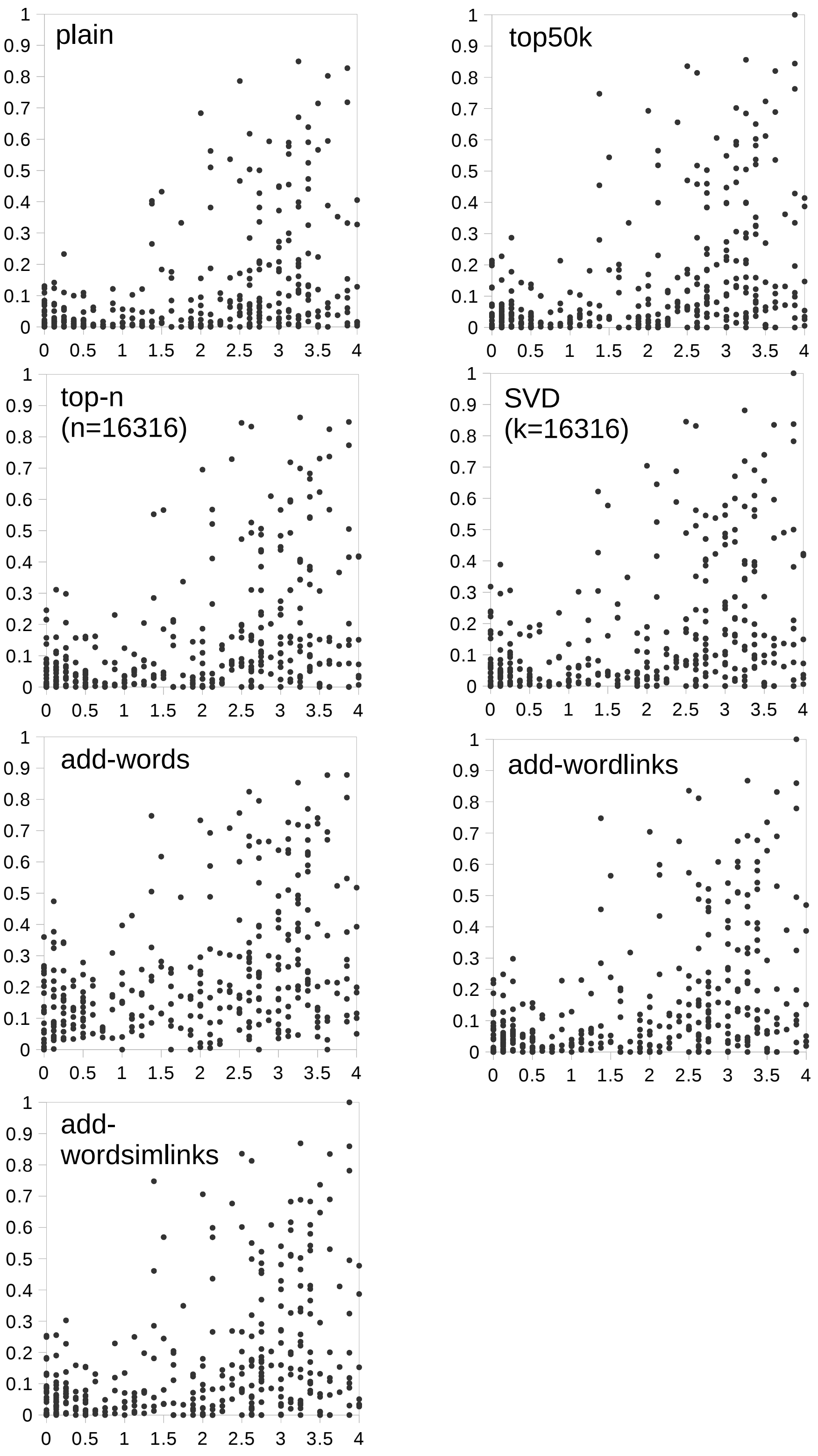

Figure A.5: Scatter plots for dependency-based models' (LMI weighted) similarity scores on Gur350 data set. Annotated scores on the x-axis, predictions on the y-axis.

```
Wiedererrichtung_Nf wiedererrichtet_A Wiederaufrichtung_Nf unverrichtet_A
Verrichtung_Nf verrichtet_A Unterrichtende_Nm unterrichtet_A Rechtsmäßigkeit_Nf
Rechtmäßige_ Unrechtmäßige_Nn Rechtmäßigkeit_Nf Unrechtmäßigkeit_Nf
Rechtlosigkeit_Nf Rechtliche_N Rechtlichkeit_Nf widerrechtlich_A Entrechtete_N
entrechtet_A Berechtigte_N Unberechtigte_N unberechtigt_A Berechtigung_Nf
berechtigt_A berechtigend_A unrechtsmäßig_A unrechtmäßig_A Aufrechte_Nf
Rechte_Nf Nebenrecht_Nn Grundrecht_Nn grundrechtlich_A Recht_Nn
Unrecht_Nn Rechtler_Nm Vorrecht_Nn Sonderrecht_Nn rechtlich_A
rechtsmäßig_A rechtmäßig_A rechtlos_A entrechten_V berechtigen_V Anrecht_Nn
antirecht_A unrecht_A recht_A berichtigen_V Berichtigen_N berichtigend_A
berichtigt_A entrichtend_A entrichtet_A nachentrichten_V entrichten_V errichtend_A
errichtet_A wiedererrichten_V Errichten_Nn errichten_V Verrichten_Nn
verrichten_V Errichtung_Nf Einrichtung_Nf  Nebeneinrichtung_Nf
Grundeinrichtung_Nf Sondereinrichtung_Nf Inneneinrichtung_Nf Inneneinrichter_Nm
Einrichten_Nn Einrichter_Nm einrichten_V vorrichten_V Richtigkeit_Nf
Unrichtigkeit_Nf Aufrichtigkeit_Nf Unaufrichtigkeit_Nf unaufrichtig_A
Wiederaufrichten_Nn wiederaufrichten_V Anrichte_Nf unterrichtend_A
Aufrichtung_Nf aufrichtend_A Ausrichtung_Nf Grundausrichtung_Nf ausrichtend_A
Abrichtung_Nf  Grundrichtung_Nf Hauptrichtung_Nf Richtung_Nf Vorrichtung_Nf
Zurichtung_Nf Unterrichtung_Nf richtend_A richtungslos_A Gegenrichtung_Nf
zurichten_V Unterricht_Nm unterrichtlich_A unterrichten_V Sonderunterricht_Nm
nachrichten_V Aufrichten_Nn aufrichtig_A aufrichten_V Ausrichten_Nn
Ausrichter_Nm ausrichten_V Abrichten_Nn abrichten_V Richten_Nn Richter_Nm
Zurichten_Nn Unterrichten_Nn richtig_A richten_V Anrichten_Nn anrichten_V
Gerichtliche_Nn außergerichtlich_A obergerichtlich_A Gericht_Nn Vorgericht_Nn
Hauptgericht_Nn Obergericht_Nn Sondergericht_Nn Zwischengericht_Nn
gerichtlich_A Entrichtung_Nf Richtungslosigkeit_Nf Richtige_Nn Unrichtige_N
Aufrichtige_N unrichtig_A Oberrichter_Nm Richterin_Nf Sonderrichter_Nm
richterlich_A Ausrichterin_Nf
```

Figure A.6: DErivBase family no. 4

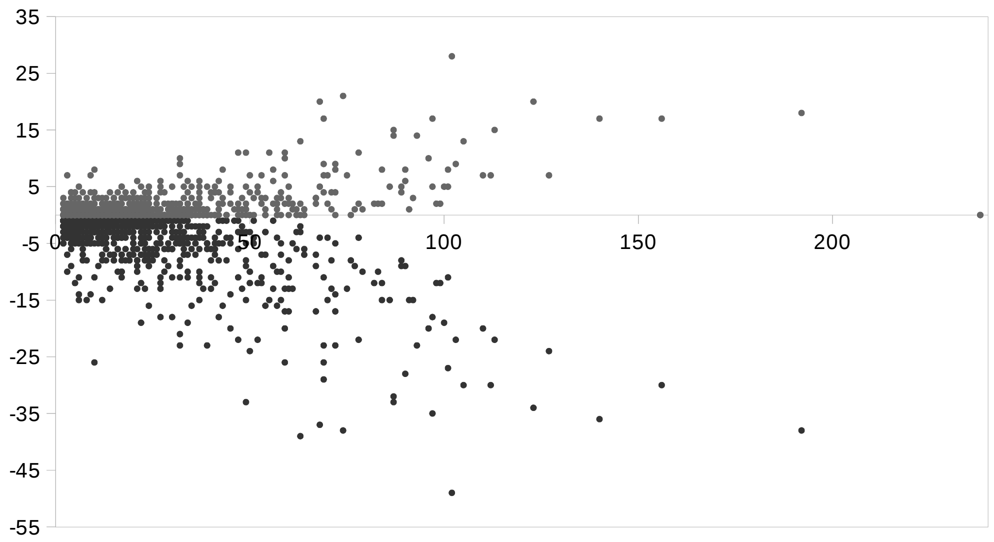

Figure A.7: The involvement of clusters in correct (light grey) and incorrect (dark grey, transformed into negative figures) in the synonym choice task for the dependency-based model, in dependency of their size. Cluster size on the x-axis, frequency of involvement (i.e. the cluster was amongst common dimensions for the predicted word pair) on the y-axis.

Bibliography

[Baroni & Lenci, 2010] Baroni, Marco, & Lenci, Alessandro. 2010. Distributional memory: A General Framework for Corpus-based Semantics. *Computational Linguistics*, **36**(4), 673–721.

[Bohnet, 2010] Bohnet, Bernd. 2010. Very High Accuracy and Fast Dependency Parsing is not a Contradiction. *Pages 89–97 of: Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, Beijing, China.

[Bullinaria & Levy, 2007] Bullinaria, John A., & Levy, Joseph P. 2007. Extracting Semantic Representations from Word Co-occurrence Statistics: A Computational Study. *Behavior research methods*, **39**(3), 510–526.

[Church & Hanks, 1990] Church, Kenneth Ward, & Hanks, Patrick. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, **16**(1), 22–29.

[Cline & Dhillon, 2006] Cline, Alan Kaylor, & Dhillon, Inderjit S. 2006. Computation of the Singular Value Decomposition. *Pages 45—1 of: Handbook of Linear Algebra*. CRC Press.

[Deerwester *et al.*, 1990] Deerwester, Scott, Dumais, Susan T, Furnas, George W, Landauer, Thomas K, & Harshman, Richard. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, **41**(6), 391–407.

[Evert, 2005] Evert, Stefan. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Stuttgart University, Stuttgart.

[Faaß & Eckart, 2013] Faaß, Gertrud, & Eckart, Kerstin. 2013. SdeWaC – A Corpus of Parsable Sentences from the Web. *Pages 61–68 of:* Gurevych, Iryna, Biemann, Chris, & Zesch, Torsten (eds), *Language processing and knowledge in the Web*. Lecture Notes in Computer Science, vol. 8105. Springer Berlin Heidelberg.

[Fano, 1961] Fano, R.M. 1961. *Transmission of Information: A Statistical Theory of Communications*. MIT Press Classics. M.I.T. Press.

[Fisher, 1925] Fisher, Ronald Aylmer. 1925. *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh.

[Gurevych, 2005] Gurevych, Iryna. 2005. Using the Structure of a Conceptual Network in Computing Semantic Relatedness. *Pages 767–778 of: Natural Language Processing (IJCNLP 2005)*. Jeju Island, Republic of Korea: Springer.

[Harris, 1954] Harris, Zellig S. 1954. Distributional structure. *Word*, **10**(23), 146–162.

[Hofmann, 1999] Hofmann, Thomas. 1999. Probabilistic Latent Semantic Indexing. *Page 50–57 of: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

[Jacquemin, 2010] Jacquemin, Bernard. 2010. A Derivational Rephrasing Experiment for Question Answering. *In: Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

[Landauer & Dumais, 1997] Landauer, Thomas K., & Dumais, Susan T. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological review*, **104**(2), 211.

[Lowe, 2001] Lowe, Will. 2001. Towards a Theory of Semantic Space. *Pages 576–581 of:* Moore, J D, & Stenning, K (eds), *Proceedings of the Annual Meeting of the Cognitive Science Society*. Lawrence Erlbaum Associates.

[McDonald *et al.*, 2006] McDonald, Ryan, Lerman, Kevin, & Pereira, Fernando. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative Parser. *Pages 216–220 of: Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, New York, NY.

[McDonald, 2000] McDonald, Scott. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh, Edinburgh.

[Mohammad *et al.*, 2007] Mohammad, Saif, Gurevych, Iryna, Hirst, Graeme, & Zesch, Torsten. 2007. Cross-Lingual Distributional Profiles of Concepts for Measuring Semantic Distance. *Page 571–580 of: EMNLP-CoNLL*.

[Padó & Lapata, 2007] Padó, Sebastian, & Lapata, Mirella. 2007. Dependency-based Construction of Semantic Space Models. *Computational Linguistics*, **33**(2), 161–199.

[Padó & Utt, 2012] Padó, Sebastian, & Utt, Jason. 2012. A Distributional Memory for German. *Page 462–470 of: Proceedings of KONVENS 2012 workshop on lexical-semantic resources and applications*.

[Padó *et al.*, 2013] Padó, Sebastian, Šnajder, Jan, & Zeller, Britta. 2013. Derivational Smoothing for Syntactic Distributional Semantics. *In: Proceedings of ACL 2013*.

[Salton *et al.*, 1975] Salton, Gerard, Wong, Anita, & Yang, Chung-Shu. 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, **18**(11), 613–620.

[Schmid, 1994] Schmid, Helmut. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. *Pages 44–49 of: Proceedings of the international conference on new methods in language processing*, vol. 12. Manchester, UK.

[Schütze, 1998] Schütze, Hinrich. 1998. Automatic Word Sense Discrimination. *Computational linguistics*, **24**(1), 97–123.

[Szpektor & Dagan, 2008] Szpektor, Idan, & Dagan, Ido. 2008. Learning Entailment Rules for Unary Templates. *Pages 849–856 of: Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics.

[Turney & Pantel, 2010] Turney, Peter D., & Pantel, Patrick. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of artificial intelligence research*, **37**(1), 141–188.

[Wallace & Wallace, 2005] Wallace, DeWitt, & Wallace, Lila Acheson. 2005. *Reader's Digest, das Beste für Deutschland*. Verlag das Beste, Stuttgart.

[Zeller *et al.*, 2013] Zeller, Britta, Šnajder, Jan, & Padó, Sebastian. 2013. DErivBase: Inducing and Evaluating a Derivational Morphology Resource for German. *In: Proceedings of ACL 2013*.

[Zipf, 1949] Zipf, George Kingsley. 1949. Human Behavior and the Principle of Least Effort.