UPPSALA
UNIVERSITET

The Distribution of Mood

An Exploration of Distributional Compositions in Sentiment Classification

Jimmy Callin

Uppsala University
Department of Linguistics and Philology
Språkteknologiprogrammet
(Language Technology Programme)
Bachelor's Thesis in Language Technology, 15 credits

May 25, 2014

Supervisors:
Joakim Nivre, Uppsala University
Jussi Karlgren, Gavagai AB

Abstract

Distributional semantics is a research area investigating unsupervised data-driven models for quantifying semantic relatedness. This thesis investigates the possibilities of using distributional semantic models for sentiment classification of utterances, by composing distributional vectors of words in utterances.

For evaluation a set of manually classified movie reviews is used for training and testing. While the purpose of this study has been to test compositions in distributional semantic model, the work has mainly been focused on finding a useful model configuration for the DSM. The thesis concludes that more associative window sizes performed better than less associative ones. Weighting the DSM by PPMI gave the most stable performance improvements as well. Context selection is essential for achieving higher scores. While DSM does not reach beyond baseline results in its evaluation, there are still unexplored areas in which potential improvements may lie.

Contents

Acknowledgements

First and foremost I would like to extend a thank you to the developers of the DISSECT framework. This was one of the main deliverables of the COMPOSES project, which provides a great framework for distributional vector compositioning (**Dinu2013Dissect**). The machine learning classifiers are a part of the scikit-learn Python framework (**Pedregosa2011ScikitLearn**). Without these libraries there would have been no way for me to finish the thesis in time.

Thank you to my two supervisors, Jussi and Joakim, for providing excellent help and feedback. I did not bother you nearly as much as I should have. My friends and colleagues made a great job in providing feedback and finding typos, for which I am very grateful.

1 Introduction

Most of the sentences you will read in this thesis have never been uttered before. The productivity of language makes this not only possible, but highly probable. Despite this, there will hopefully be little trouble understanding the meaning, motive and conclusion of this thesis. These meanings are built up by words you have had a lifetime to learn and ponder. A few of these you might have had to look up, but for the majority all you needed was the *context* in which they were used.

Unlike humans, however, computers have difficulties making the same kind of inference from text. Today, words do not carry the same level of meaning to a machine as it does to humans. To a machine, these words are just another string of data to be processed or ignored. Either we can program new commands to the machine directly, or we can give it manually compiled training data for it to generalize based on recognizable patterns. But once a word appears outside of the domain of available commands, never seen in the training session, things start to get tricky. Suddenly the computer is confronted with a new piece of data; a new symbol. Processing this new symbol would be trivial for you, but for the machine the best strategy would typically be to ignore it, hoping it is redundant in the bigger picture. No matter how many times the machine would see this new unfamiliar symbol, without the interference of human hands it would not know what to do.

Text has structure; a *distribution* of building blocks. The purpose of distributional semantics is to enable the computer to go beyond the relatively fragile domain of previously taught commands, and to begin learning new words based on their surrounding neighbors. No longer would the productivity of language be a problem, but rather a source from which to learn new knowledge.

While still a field in its infancy, promising results are already emerging for various applications. The *mood* of text is but one of them, and the one we are to investigate further below.

1.1 Purpose

The purpose of this thesis is to investigate if it is possible to use compositions of distributional vectors for classifying the mood of utterances. The main contribution of this thesis will be to establish a useful parameter tuning to use in a compositional distributional bag-of-words model. This is later evaluated in sentiment classification using a machine learning algorithm.

I will assume that a bag of words model will be sufficient for representing the necessary linguistic information for sentiment classification. I will further

assume that the composition vectors are linearly separable in the feature space, and that all utterances can be classified as either positive or negative. Finally, I will assume that word meaning can be learned from linguistic environment.

The final model is to be evaluated against a *tfidf* classifier, which is explained in detail in section 4.2.4.

1.2 Outline

Chapter 2 will be dealing with the necessary background for the experiments, broadly going through the whats and the whys of sentiment classification and distributional semantics, and the benefits of combining these two fields. Chapter 3 introduces the parameters included in the model which is to be tuned, while section 4.1 explains the experimental framework. After going through the method in chapter 4, including the topics of dataset and baseline, chapter 5 presents the results. They will be split into subparts based on the tested parameters. Thereafter, in chapter 6, I will closely analyze the results from the output. Finally, chapter 7 will conclude the thesis.

2 Background

2.1 Sentiment analysis

Sentiment analysis has become a hot topic in recent years, and is of increasingly growing importance in industrial sectors related to business intelligence and data mining. We have started to see a wide variety of consumer-end applications, mainly driven by the desire to find computerized methods for automatic market analysis within a large variety of topics. In a world of *Big Data*, the possibilities of actionable early warning systems are of great interest for companies with a large social media exposure.

This is not only attractive from a market analysis perspective, but also within investment banking, security intelligence, public opinion polls, and other areas where predictive analytics are of interest (**Karlgren2012Usefulness**). Some of the research has been motivated by trying to find an attitudinal signal in social media data which predicts the rise and fall of stock market shares, with some successful results (**Bollen2011Twitter**). Others have found close correspondence between parties' and politicians' popularity and their sentiment ranking on Twitter (**Tumasjan2010Predicting**).

Although the current applications may be driven primarily by relatively straightforward data analysis, there are important aspects of sentiment in other text analytics areas as well. That mood[1] affects the flow of text is not a controversial statement, which makes the relevance of sentiment analysis for text generation applications such as machine translation and speech synthesis interesting (**Wei2010Cross**; **Alm2005Emotions**).

From a technological perspective, the sentiment classifier predicts an attitude in text, where the text is either on a document level or a sentence level (**Kim2013Beyond**). While certain psychologists argue for at least six basic human emotions, the field of study so far has mainly been focused on the binary identification of positive and negative sentiments in text. While the view of mood as a binary classification problem is clearly a gross oversimplification, even this simple relationship is not without its problems. The attitude of utterances is often in the eye of the beholder. For instance, would you classify *the food was good, but the service was awful* as a positive or negative utterance? The mood of an utterance is a delicate research problem, and the assumptions made here cannot be argued to represent an actual model of the human mood.

Early, naive methods of sentiment classification belong to rule-based classification models mainly using pre-compiled word lists. While a computationally

[1] While there might be subtle differences between the mood, attitude, and sentiment of an utterance, there are no established xx within the field. In this thesis these words are used interchangeably.

efficient and transparent method, it is of limited explanatory power with regard to the complexity of sentiment. The sentiments of words are rarely, if ever, universal, and context is of great importance: whether a word such as *rise* determines positive or negative sentiment, depends entirely on the domain of the document, e.g. stock market analysis or global warming discourse.

Data driven methods have, for the last decade, been the preferred way to go, and mainly use popular machine learning classifiers such as Support Vector Machines (SVM), Naive Bayes, Logistic Regression, or k-Nearest Neighbors (**Feldman2013Techniques**). By manually classifying a large number of texts as one of the desired sentiments, or potentially neutral when attitude is missing, you have compiled a data set which to divide into a larger set for training, and a testing set for evaluation. When the training instances are manually annotated, as in most sentiment classification methods, it is called *supervised learning*. There are also experiments in avoiding pre-classified training data, and instead extrapolate from a select few words prototypical for each sentiment class. These typically belong to *unsupervised learning methods*, which emphasize the lack of manually annotated data.

By simple means, given a high enough quality of training data, it is possible to train classifiers to achieve relatively high scores in evaluation. The problem lies in compiling attitudinally classified datasets, which are mainly available for only a select few languages.

2.2 The usefulness of distributed representations

Comprehensive training data for most classification tasks are rare, especially outside of the English language. Even for English the training data for sentiment classification is almost guaranteed to be too patchy to contain sufficient linguistic information to cover all possible attitudinal situations. This is especially true when applying sentiment classification in media composed of user-generated content, where the productive nature of these domains creates a linguistic environment filled with noisy, unreliable, unorganized text. This forms a demand for models that are able to take these aspects into consideration. Classification models that are meant to perform well on user-generated content should be able to learn about new situations *online*, i.e. learning while doing, and not depend entirely on attitudinally classified training data. This is where distributional semantic models become relevant.

2.3 Distributional semantic models (DSM)

Distributional semantics is a research area whose purpose is to automatically model semantic information using word collocation data from large text collections, which enables quantification of *word similarity* (**Schuetze1993Word**). The motive behind applying DSMs[2] on sentiment classification comes back to the high productivity of language in social media contexts, where new words, hashtags, and misspellings occur frequently enough to be difficult to ignore.

[2]While these models also are known as *word spaces*, *vector spaces*, and *semantic spaces*, I will use *distributional semantic models* exclusively in this thesis.

By generalizing words to a higher form of semantic representation, the goal is to remove the problems linguistic productivity usually introduces, since the generalization gives context to words not available in the training set.

Even if linguistic productivity would not be a problem, DSMs are still of great interest. There are manually compiled lexical resources such as WordNet readily available, providing manually annotated semantic representations. Creating these resources are often expensive and tedious and only available for a select few languages. It also lies in their nature to be based on linguistic assumptions including the personal bias of semantic relatedness which is next to impossible to avoid.

The DSM is a *vector space* of *distributional vectors* built from word co-occurrence counts in large corpora. These distributional vectors are quantified semantic representations of each word available in the corpus. By using a *similarity measure* for calculating how much the words' context correlate, it is possible to compare these distributional vectors to all other available words' semantic representation (**Schuetze1993Word**).

J. R. Firth uttered *you shall know a word by the company it keeps*; an inspiration for *the distributional hypothesis* (**Sahlgren2008Distributional**). The distributional hypothesis is the central cornerstone of distributional semantics, stating:

> [W]ords that occur in the same contexts tend to have similar meaning (**Pantel2005Inducing**).

What could be considered as context is deliberately vague, since different types of context seem to generate different types of semantic representations. Most experiments in modeling DSMs use word co-occurrence with varying sizes of maximum word distance. There have been further experimentations on the inclusion co-occurrence distance, part-of-speech tags, word order, and dependency structures to affect the context. While tagging with linguistic data in a few cases have shown to improve evaluation performance, they typically lead to sparse data problems (**Sahlgren2008Distributional**).

The most basic of DSMs is the word co-occurrence matrix. This is the basis of all other models, where each row represents a specific word and each column initially represents the frequency of each word's appearance within context of the focus word (see table 2.1 for example). The drawback of this model is its computational and spatial inefficiency. While there are other more efficient models available that represent various DSMs, such as *LSA* (**Landauer1997Solution**), *HAL* (**Lund1995Semantic**), and *Random Indexing* (**Kanerva2000Random**), the effect of the compression algorithms would be an uncertain variable in the classification task. As such, I will continue to use the basic co-occurrence matrix.

	drink	abuse
coffee	780	220
tea	660	80

Table 2.1: A simple co-occurrence matrix of *coffee* and *tea*, in relation to *drink* and *abuse*. Each row defines a word seen in the corpus, and each column defines the number of times *drink* or *abuse* has been mentioned within context of the word in focus.

The initial building process of a basic DSM is quite straight-forward. When reading the corpus, co-occurrence counts are collected by focusing on each word in the text and gathering occurrence statistics about the neighboring words within a set maximum distance (hereafter called a *context window*). After processing a large amount of data, each row in the matrix will represent the *distributional vector* of a word. By comparing the *similarity* of distributional vectors, we obtain the similarity of word senses. See table 2.2 for an example of the closest neighbors to two common words using the cosine distance.

Spectacular	Horrible
stunning: 0.109	horrid: 0.0995
scenic: 0.103	awful: 0.0947
impressive: 0.103	foyster: 0.0787
amazing: .0871	terrible: 0.0784
vistas: .0829	horrific: 0.0770
elaborate: .0815	lehnert: 0.0764
dramatic: .0810	calumny: 0.0761
shivanasamudra: .0801	horrifying: 0.0729
breathtaking: .0797	edifying: 0.067
magnificent: 0.078	abhorrent: 0.0667

Table 2.2: Closest cosine neighbors in a DSM built from 600MB Wikipedia data with a context window of 2+2, i.e. a maximum of 2 words to the left and 2 words to the right of the focus word.

2.4 Compositionality in distributional models

The idea that the meaning of an utterance is composed by its words takes its initial steps in *the principle of compositionality*, stating:

> The meaning of a complex expression is a function of the meaning of its parts and of the syntactic rules by which they are combined (**Partee1990Mathematical**).

Traditionally, compositionality in computational linguistics has been developed using logical frameworks (**Kartsaklis2014Compositional**). Distributional compositional models step beyond this, trying to solve the problem using data driven methods based on distributional semantic modeling.

3 Model

The model in use for sentiment classification will be built by applying the DSM as explained in section 2.3 to a distributional semantic vector composer, resulting in an utterance vector per training instance, where a training instance is an utterance paired with its corresponding sentiment class (i.e. positive or negative). The quality of the DSM is heavily dependent upon a large number of parameters, such as the size of the context window in use. Other available parameters are the weighting of the DSM, weighting of specific contexts and words, context selection, etc. Below I will go through each of them in detail.

3.1 Window size

In his dissertation, **Sahlgren2006WordSpace** argues for the importance of how differing window sizes reveal different linguistic relations. Sahlgren concludes that 2+2 window sizes expose neighbors with a *paradigmatic* relationship, while using larger window sizes reveal more *associative* neighbors. What this essentially means is that smaller window sizes will uncover neighbors similar to synonyms, antonyms, hyponyms, i.e., words that are *interchangable* in their prototypical contexts. Larger window sizes, on the other hand, reveal words that are *associated* to the word in focus, and do not necessarily share any specific linguistic semantic classes.

Using the co-occurrence matrix example in table 2.1, *coffee* and *tea* share a paradigmatic relationship since they in many contexts are of interchangable nature. *Drink* and *abuse* have an associative relationship with *coffee* and *tea*, but the connection is stronger to *coffee* rather than *tea*.

3.2 DSM weighting

An unweighted co-occurrence matrix is of limited use as a DSM. The frequency of common function words do not correspond to their semantic relevance, and negatively affect the result. Raw frequency count is also dependent on the size of the corpus, which makes it difficult to compare results between different corpora. Weighting word pairs by their mutual semantic relevance decrease the importance of common function words that have little semantic relevance and normalize the distributional vectors.

There are several algorithms for performing such weighting. In the evaluation, we are to test *Positive Point-wise Mutual Information* (PPMI), *Exponential Point-wise Mutual Information* (EPMI), and *Positive Local Mutual Information* (PLMI) (**Baroni2010Distributional**), as specified in eq. (1). While EPMI

and PPMi are considered traditional measures in information retrieval, PLMI is a relatively new measurement whose purpose is to improve PPMI in the area of low-frequent words, where PPMI has a tendency to exaggerate.

In **Bullinaria2007Extracting** they review common weighting operations on DSM, demonstrating that the PPMI algorithm performs best in their evaluations.

$$\begin{aligned}
\text{pmi}(w_1, w_2) &= \log \frac{P(w_1, w_1)}{P(w_1)P(w_2)} \\
\text{epmi}(w_1, w_2) &= \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \\
\text{ppmi}(w_1, w_2) &= \text{pmi}(w_1, w_2) \text{ if pmi}(w_1, w_2) \geq 0 \text{ else } 0 \\
\text{plmi}(w_1, w_2) &= \text{ppmi}(w_1, w_2) \times \text{count}(w_1, w_2)
\end{aligned} \tag{1}$$

Where w_1 and w_2 correspond to a word pair, P being the probability of either a single word or the co-occurrence of two words.

3.3 Word and context weighting

Not all words are of equal importance in classification tasks. By applying different weighting methods to the vectors, we can boost or reduce the importance of either specific context words or entire distributional vectors in utterances before composition. When weighting the distributional vector $\vec{v}$ of a word w, the weighting scheme s acts as a scalar value to be multiplied with the distributional vector:

$$\text{word\_weight}(w, s) = s_w \vec{v_w} \tag{2}$$

When weighting the contexts of words, the weighting scheme s acts as a weight vector, performing dot product multiplication with the distributional vector $\vec{v_w}$.

$$\text{context\_weight}(\text{w}, \text{s}) = \vec{s} \cdot \vec{v_w} \tag{3}$$

In case of words missing from the weighting scheme, both word and context weighting default to 1.

The two weighting schemes to be tested are *inverse document frequency* (*idf*), and the *Gini*-index. *Idf* is a popular weighting mechanism in information retrieval tasks for increasing the importance of words that are less common, and reduce the importance of words that occur more frequently. Defined as:

$$\text{idf}(w, D) = \log \frac{\text{df}_w}{N_D} \tag{4}$$

Where df_w is the document frequency of word w, and N is the total number of documents in the set of documents D.

While *idf* is well suited for reducing the importance of commonly used words, it does not take into account the discriminatory power of such words

given the classification problem at hand. **Zhu2013Using** introduce the Gini-index as a solution to this problem, and also argue that the Gini-index performs better at various classification tasks.

The Gini-index in this classification problem is defined as:

$$\text{Gini}(w) = \sum_{i=1}^{N_S} P_i(w)^2 \tag{5}$$

Where P_i is the probability of word w ending up in class i belonging to set S. This means that words which are evenly distributed among the categories in S will get lower values, compared to words that mainly occur in one specific class. Higher Gini-index correlates with higher discriminatory power.

While DSM weighting explained in the previous chapter uses the co-occurrence data gathered from an unlabeled corpus, the word and context weighting uses labeled training data. The main hypothesis in play here is that by using training data, any potential data variance between the unlabeled and labeled corpus is reduced. Labeled training data is also necessary for calculating the Gini-index of each word.

3.4 Context selection

Unimportant context words may have the tendency to confuse the classifier. By applying feature selection, or rather context selection in this case, it is possible to remove potentially noisy context words.

The method to be used for context selection is to either sum all of the DSM weighted word co-occurrence frequencies, or calculating the length of the vector. Thereafter the top k context vectors are selected, where the optimal value of k is to be empirically decided in evaluation. As such, it will choose the contexts that are of most semantic relevance according to the weighting scheme w, which will be one of the weighting algorithms introduced in section 3.2.

$$\begin{aligned} \text{sum}(\vec{v}) &= \sum_{i=1}^{n} w(v_i) \\ \text{length}(\vec{v}) &= \sqrt{\sum_{i=1}^{n} w(v_i)^2} \end{aligned} \tag{6}$$

Where $\vec{v}_i$ is a specific field in the context vector $\vec{v}$. A weighting scheme is used as a way to retrieve different top k words. If using no weighting scheme, the sum and length of the context vector correspond closely to the top word frequency. If using e.g. PPMI as a weighting scheme, the function words and other common words are reduced in their importance, making the top k containing more semantically rich words. See table 3.1 for comparison of different weighting methods using vector length.

Raw frequency	PPMI	PLMI
the	de	the
of	born	of
in	include	was
and	known	age
to	son	median
a	la	income
was	n	he
is	called	in
for	name	as
as	near	were
on	like	his
by	named	females
with	former	is
that	located	males
he	e	km
from	r	to
at	john	for
it	daughter	united
his	famous	states
were	m	be

Table 3.1: The top 20 context words of applying different weighting scheme to the context selection. Calculation is based on the context vector length.

3.5 Distributional vector composition

While the work in this thesis does not try to develop a probable model of the meaning of an utterance, it still draws inspiration from distributional compositional semantics. The purpose of the composed distributional vectors in this work is rather about finding an empirically tested representation well suited for machine learning classifiers.

The composition opseration used in the model is the weighted additive function:

$$\vec{p} = \alpha\vec{u} + \beta\vec{v} \tag{7}$$

Setting α and β to 0.5 essentially creates an averaged centroid model, which has shown to perform well in various information retrieval tasks (**Mitchell2010Composition**). That said, there are many properties of word composition that are missing from vector addition, where the arguably most important factor is non-commutativity, i.e. an additive function does not differ between *coffee house* and *house coffee*.

4 Method

4.1 Framework

Separate modules were developed for the construction of DSMs, training and testing distributional vector compositions toward machine learning classifiers, as well as a module for results evaluation. Altogether, this makes the evaluation framework used for the experiments in this thesis.

4.1.1 Distributional semantic model builder

This module builds the DSM according to a set of parameters:

1. Context window size – see chapter 3.1.
2. DSM weighting – see chapter 3.2.
3. Context weighting – see chapter 3.3.
4. Word weighting – see chapter 3.3.
5. Context selection – see chapter 3.4.
6. Secondary DSM – Providing a secondary DSM would merge two DSMs into one, giving the ability to add domain specific content. This is simply done by summing co-occurrence frequencies.
7. Stopwords filtering – Ignoring stop words is a common occurrence in information retrieval tasks. Stopwords are removed *before* applying any weighting mechanism.

4.1.2 Utterance parser

This module parses utterances into a binary *composition tree*, where the leaves are the distributional vectors of the utterance, and each node represents the compositionality function to be used on the children vectors. See figure 4.1.

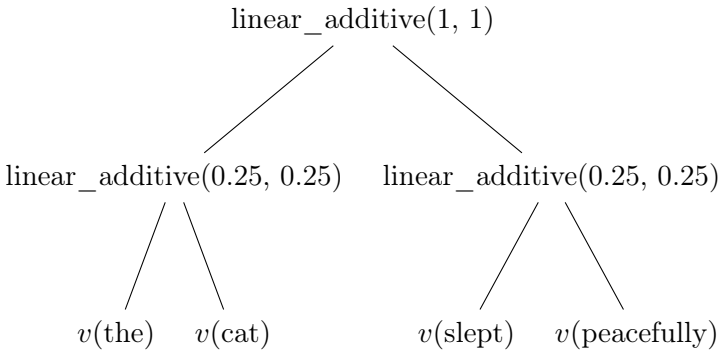

Figure 4.1: A parse tree of the centroid model. The distributional vector composer iterates in an infix order over the tree.

4.1.3 Distributional vector composer

The distributional vector composer takes a composition tree as an argument, traversing the tree in an *infix* order, and step-by-step applies the given compositionality functions to its children vectors. This returns a *distributional composition vector*.

4.1.4 Sentiment classifier

The sentiment classifier contains the training and testing data, sends off the sentences to the utterance parser for parsing, and thereafter to the distributional vector composer for creating the distributional composition vectors. These composition vectors are then used as the instances in the classifier's training session, together with the corresponding sentiments for the original utterances. The utterances are also converted to their distributional composition vectors in the testing session for predicting the sentiment of the utterance.

4.2 Experimental setup

4.2.1 Data

Training and testing data is a set of movie review sentences introduced by **Pang2005Seeing** It is a made up of 10662 polarized reviews evenly split between positive and negative instances, and is particularly chosen for its sentence level sentiment classification. It is a commonly used data set for this type of sentiment classification task. For example data, see example 1.

(1) a. this is a film well worth seeing , talking and singing heads and all . analyze that " is one of those crass , contrived sequels that not only fails on its own , but makes you second-guess your affection for the original.

b. this bond film goes off the beaten path , not necessarily for the better .

c. a movie that will touch the hearts of both children and adults , as well as bring audiences to the edge of their seats .

For construction of DSMs, I used a corpus of 330 MB worth of text from English Wikipedia. The main motive here is to have a source of neutral, high-quality text with a broad domain of topics. The corpus was normalized by removing any non-alphanumeric characters, keeping smileys and emoticons, and transformed into lowercase letters.

Using the dataset in this paper, **Socher2013Recursive** have gotten an F1 score of 85.4%, which is considered the current state-of-the-art.

4.2.2 Metrics

Ten percent of the movie review data set is used for testing while the rest is used during training session.

The standard metrics for measuring classification accuracy are precision and recall, as are used in these results as well. Presented along the mentioned metrics is the F1 score, giving a combined score based on the precision and recall values. These are calculated as shown in equation 8.

$$\begin{aligned} \text{Precision} &= \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \\ \text{Recall} &= \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \\ \text{F1} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{8}$$

4.2.3 Choosing a classifier

There are mainly two aspects in the experiment to consider when picking a classifier. First and foremost, because of the multitude of data and the uncompressed nature of the DSM, it is difficult to keep all of the training instances in memory at once. The classifier should be able to learn in an iterative fashion of only having to keep a subset of all instances in memory, i.e. an *online* classifier. The other aspect is the fact that the model will have a large number of features, as such it would be beneficial if the classifier would be optimized at handling many possibly noisy features.

Using the *Stochastic Gradient Descent* (SGD) optimization algorithm it is possible to train linear classifiers using subsets of the training data, living up to the online learning criterion. The *Support Vector Machine* (SVM) is a commonly used linear classifier in computational linguistic areas, and previous research has concluded that it performs well compared to other classifiers in sentiment classification tasks **Yao2011Sentiment** Based on this, the SVM is our choice of classifier.

4.2.4 Baseline

As a baseline, I used a SVM model using a *tfidf* weighted term-document matrix for training the SVM classifier. This is a common vector space model

Polarity	Precision	Recall	F1
Neg	0.73	0.74	0.74
Pos	0.74	0.73	0.73
Average	0.73	0.73	0.73

Table 4.1: Baseline results. Classified with the SVM classifier explained in section 4.2.3.

in information retrieval as a tool for finding arbitrary topics in document collections. See table 4.1 for results.

4.2.5 Experiment workflow

Since there are too many available parameters for testing each and every combination, each model parameter, as listed in chapter 3, will be tested on its own using the setting which performs best when evaluating the next step. The order of the parameter testing will affect the end result, and creating an optimal workflow is difficult to do in advance. Hence the order will rather be a mix of my intuition of a well designed order, as well as being motivated by what makes most sense from a pedagogical perspective.

The parameters are to be tested in the following order:

1. Window size
2. DSM weighting
3. Removal of stop words
4. Adding domain specificity
5. Context weighting
6. Word weighting
7. Context selection

5 Results

This chapter lists the results as introduced in the previous chapter. Each section will list the parameter variations with their corresponding F1, precision, and recall values. The best evaluation scores for given metrics are emphasized in each column, where F1 score generally decides which parameter setting to pick for coming experiments.

5.1 Window size

Table 5.1 lists the results of running the classifiers against various window sizes. See section 3.1 for details.

Window size	F1	Precision	Recall
2 + 2	0.603329	0.637012	0.617783
5 + 5	0.607171	0.67205	0.631194
10 + 10	**0.708129**	**0.714285**	**0.709513**
20 + 20	0.700301	0.707315	0.701975

Table 5.1: Average scores for each window size, using a PPMI weighted DSM.

5.2 DSM weighting

Because of the previous results in section 5.1, the window size to be used is 10+10 when running the space weighting experiments. See table 5.2 for scores, where you will see that PPMI performs best given a 10+10 window size. section 3.2 contains details about DSM weighting.

Weighting	F1	Precision	Recall
None	0.341719	0.60667	0.502786
PPMI	**0.708129**	**0.714285**	**0.709513**
EPMI	0.666205	0.669963	0.667316
PLMI	0.503742	0.532896	0.52668

Table 5.2: F1, Precision, and Recall results using different weighting mechanisms, as defined in chapter 3.2. Window size used is 10+10.

5.3 Removal of stop words

The stop words are removed *before* performing the DSM weighting. See table 5.3 for results. Removal of stop words does not affect the results in a beneficial manner neither for removal of context nor utterance words. Why removal of stop words is tested is explained briefly in section 4.1.1.

	F1	Precision	Recall
None	**0.708129**	**0.714285**	**0.709513**
Contexts	0.668122	0.669705	0.668727
Utterance words	0.682691	0.685202	0.683336
Contexts+words	0.609653	0.678644	0.633566

Table 5.3: Results of removing stop words from the DSM. Window size: 10+10, weighting: PPMI.

5.4 Adding domain specificity

Adding movie reviews training data to the model, which is to be considered domain specific content, does not increase performance. Why inclusion of domain specific content is tested is explained in section 4.1.1.

Corpus	F1	Precision	Recall
Wiki	**0.708129**	**0.714285**	**0.709513**
Wiki + training	0.695549	0.696601	0.695918

Table 5.4: Results of adding training data to the word space. Window size: 10+10, weighting: PPMI.

5.5 Context weighting

Weighting specific contexts does not increase performance in the case of a DSM with a window size of 10+10. See section 3.3 for details.

	F1	Precision	Recall
None	**0.708129**	**0.714285**	**0**.709513
idf	0.671952	0.724149	0.686413
Gini-index	0.694719	0.739686	0.706048

Table 5.5: Results of adding weighting to columns. Window size: 10+10, weighting: PPMI.

5.6 Word weighting

Weighting the distributional vectors of each utterance word with scalar values calculated by neither *idf* nor Gini-index seem to improve performance. See section 3.3 for details.

	F1	Precision	Recall
None	**0.708129**	**0.714285**	**0.709513**
idf	0.697238	0.714254	0.701975
Gini-index	0.603461	0.692121	0.634156

Table 5.6: Results of adding weighting schemes to words as scalar values in the utterance. Window size: 10+10, weighting: PPMI.

5.7 Context selection

Applying context selection improves the results, where selection by context vector length is better than summation of context. See table 5.7 for results, and section 3.4 for details.

Weighting	N contexts	F1	Precision	Recall
length	500	0.651737	0.681782	0.661814
	1000	0.659086	0.709866	0.674211
	2000	0.646023	0.707617	0.664918
	3000	**0.752076**	**0.752164**	**0.752072**
	4000	0.737595	0.744773	0.739323
	5000	0.516312	0.710203	0.587709
	10000	0.596466	0.721539	0.636202
	15000	0.596466	0.721539	0.636202
	20000	0.596466	0.721539	0.636202
	30000	0.596466	0.721539	0.636202
	40000	0.596466	0.721539	0.636202
	50000	0.596466	0.721539	0.636202
	60000	0.596466	0.721539	0.636202
	80000	0.596466	0.721539	0.636202
	100000	0.596466	0.721539	0.636202
sum	500	0.689111	0.698652	0.69158
	1000	0.636127	0.703597	0.657459
	2000	0.681463	0.71108	0.689949
	3000	0.728748	0.737466	0.730912
	4000	0.707386	0.722132	0.711321
	5000	**0.739104**	**0.748253**	**0.741245**
	10000	0.596466	0.721539	0.636202
	15000	0.596466	0.721539	0.636202
	20000	0.596466	0.721539	0.636202
	30000	0.596466	0.721539	0.636202
	40000	0.596466	0.721539	0.636202
	50000	0.596466	0.721539	0.636202
	60000	0.596466	0.721539	0.636202
	80000	0.596466	0.721539	0.636202
	100000	0.596466	0.721539	0.636202

Table 5.7: Results of performing context selection to the DSM. Window size: 10+10, weighting: PPMI.

5.8 Final configuration

The result of the final configuration is based on the top 3000 context length from last section. In table 5.8 the configuration is compared against the original baseline. The 2 percentage points performance increase is not a statistically significant improvement.

	DSM			Baseline		
	F1	Precision	Recall	F1	Precision	Recall
Negative	0.76	0.75	0.76	0.74	0.73	0.74
Positive	0.75	0.75	0.74	0.73	0.74	0.73
Average	**0.75**	**0.75**	**0.75**	0.73	0.73	0.73

Table 5.8: Results of the final model. Window size: 10+10, weighting: PPMI, feature selection: top 3000 contexts using context vector length.

6 Discussion

Window sizes which create DSMs of more associative natures, i.e. larger window sizes, seem to perform better. Intuitively this should make sense, since paradigmatic relations rarely catch what could be considered attitudinal information; in a paradigmatic distribution *bad* is a close neighbor to *good*, and the paradigmatic neighbors to *cotton candy* do not bring any of the associations we usually connect to it.

The best weighting method for associative DSMs is PPMI. This corresponds with previous studies on DSM weighting methods. Interestingly enough, applying EPMI on DSMs with smaller window sizes enhances the result significantly, so while it does seem to be possible to achieve fairly decent results using other combinations of weighting and window size, these appear to be less stable when experimenting with other parameters. I have yet to find a parameter setting using smaller window sizes with alternative weighting schemes that out-performs the final model.

Based on this, it appears that there is a strong dependency between a given parameter of the DSM and the remaining variables; finding the optimal setting for a certain parameter will not necessarily translate to an optimal setting if you change another value in the model configuration.

The fact that removing stop words from either the context of the distributional vector or the utterance does not improve performance is an interesting phenomenon, considering how common such a procedure otherwise is in information retrieval tasks. The effect of function words and how they can be used to manipulate the context of the DSM is surely something that would be an interesting topic for further studies.

Weighting of either words or columns does little to affect the result. My interpretation of this is that the correlation of the context words is too delicate for simple feature weighting. If the weighting needs to be improved, it should be done in correspondence with the initial DSM weighting, rather than as an independent weighting scheme as done here.

Domain specific content was hypothesized to bring better results than not, by training on the provided training data. My interpretation of this is that there was either too little training data for learning new linguistic domains, as I consider movie reviews to be, or that the mix of two different domains into one DSM is rather more confusing than helpful for the model.

Despite SVM being known to handle large amount of features in a robust manner, context selection is still a fundamental part of higher performance, creating significantly better results. Initial experiments show that applying the context selection after the weighting is key in selecting good context candidates. PPMI weighting appears to remove any function words that would otherwise

be included. Based on the performance of stop words filtering, I wonder if there could be interesting results from trying differing weighting schemes between the DSM weighting and when choosing context words. PLMI could be a good candidate, based on its top 20 context words in table 3.1.

The fact that the baseline *tfidf* model and a DSM with an associative window size perform similarly might be because in some sense they model similar structures. The term-document matrix which makes up the *tfidf* model is traditionally used in information retrieval tasks for answering what a document is *about*, assuming the *bag-of-words* hypothesis (**Turney2010From**). The associative DSM could be seen as an approximation of such a model.

The lack of compositional operation testing here depends on the fact that it was difficult to find alternative compositional operations that could be performed on entire utterances. Most previously published operations depend in some way on methods related to elementwise multiplication. When performed on entire utterances, this would cause the length of the sparse distributional composition vector to reach zero. Any methods trying to mix multiplicative and additive models gave no promising results. I believe this is the key area for further improvements in classification using distributional semantic models.

As previously discussed, the lack of compression on the DSM increases the spatial and computational cost exponentially. From a developer's perspective, this meant a great deal of optimization for assuring the model would fit in memory. Still, the training and testing take considerably longer time for composing the vectors compared to the baseline model. Any models meant for practical use without doubt need to use more scalable versions of DSM.

With that in mind, how the results of these experiments would translate to a DSM like LSA or Random Indexing is uncertain; not all calculations would be readily available for each model (LSA, for instance, already has built-in dimensionality reduction through singular value decomposition), and what the compression does to affect any compositional operations must be further investigated.

7 Conclusion

This thesis has investigated the possibilities of using compositions of distributional semantic vectors for sentiment classification of utterances. The work has mainly been focused on finding an optimal parameter tuning for the DSM, where the results concluded that more associative window sizes performed better than less associative ones. Weighting the DSM by PPMI gave the most stable performance improvements as well. Context selection is essential for achieving higher scores.

While the model does not reach beyond baseline levels in its evaluation, there are still unexplored areas in which potential improvements may lie, mainly in the investigation of composition operators beyond a simple additive model.