KTH Computer Science
and Communication

Sentiment analysis of Swedish social media

Using random indexing to improve cross-domain sentiment classification

TOMAS LYSEDAL

Master's Thesis at NADA
Supervisor: Kai-Mikael Jää-Aro
Examiner: Olle Bälter

TRITA xxx yyyy-nn

Abstract

Social media has grown extremely fast in recent years and in the vast number of posts being made everyday people express their opinions about all kinds of topics. These opinions are very valuable and there is a need for a way to automatically identify and extract them. This is what sentiment analysis is about but there are a number of issues related to this task. In particular the large number and diversity of the texts to analyze causes problems for ordinary methods of natural language processing. In this thesis a method utilizing a technique called Random Indexing is proposed which tries to overcome some of the issues. The conclusion is that the use of Random Indexing does aid in solving the problem but also that more work is needed in order to have a fully satisfying solution.

Sammanfattning

Sentimentanalys av svenska sociala medier

Användningen av sociala medier har vuxit snabbt de senaste åren och i den stora mängd inlägg som skrivs varje dag gömmer sig många människors åsikter. Dessa åsikter innehåller värdefull information och det behövs ett sätt att automatiskt identifiera och ta tillvara på den. Sentimentanalys behandlar precis detta men det finns ett antal svårigheter med att lösa denna uppgift. Svårigheterna rör framförallt att det finns en så stor mängd texter att analysera och hur väldigt olika de kan vara. I det här exjobbet föreslås en metod som använder sig av en teknik kallad Random Indexing för att överkomma vissa av dessa svårigheter. Slutsatsen är att användningen av Random Indexing hjälper till att lösa problemen men att det fortfarande krävs mer arbete för att få fram en fullt fungerande lösning.

Contents

Chapter 1

Introduction

The use of social media has grown extremely fast in recent years and it is becoming the preferred channel of communication for a lot of people. On websites like Facebook and Twitter a huge number of posts is being made every day and with all Internet forums and private blogs even more data are produced. Hidden in all these data are the views and opinions of a large group of people making social media an invaluable source of information. Companies might want to know what potential buyers think about their products and political parties the voters' views on certain issues. This leads to the need to monitor the stream of data for relevant texts combined with an analysis of the expressed opinions.

By using the API:s provided by the sites themselves or external companies, you can filter out texts that mention a certain product or keyword. Even when only considering these texts however, there are usually still too much data for a human to go through. To be able to make use of all the information some kind of automatic analysis is needed. This is what sentiment analysis, or opinion mining, is about; analyzing human sentiment, opinions and emotions expressed in text.

1.1 Background

Sentiment analysis is a relatively new area of research and it was with the growth of the Internet that it began to get more attention. One of the reasons behind that is the fact that, until then there was a very limited amount of opinionated text available for researchers to study [11]. The focus of the field of Natural Language Processing (NLP) was instead directed on tasks like information retrieval and topic based classification and clustering in contrast to opinion-based. While some aspects of sentiment analysis are closely related to those tasks and the same techniques can be applied new methods are also needed to better capture opinions and emotions.

Just as the use of the Internet and social media grew extremely fast so did the amount of research in the area of sentiment analysis. The benefits of automatic opinion analysis became apparent which resulted in a vast number of reports and applications from both the academic as well as the commercial world[1]. In an attempt to unite the different branches Bing Liu tries to define the different problems involved and summarize what has already been done [11].

The general definition of the problem in sentiment analysis consists in short of the following parts. An *object* is any entity about which opinions can be stated, like a product or an event, further each *object* has a set of *features* like the screen of a cell phone or the length of a concert. An *opinion holder*, usually the author of the text, can then express an opinion with an *opinion orientation* like a positive or negative view on a certain *feature*, e.g. that the screen of a phone is good. With these definitions the general problem is, given a document of text, find all quintuples of *object*, *feature*, *opinion orientation*, *opinion holder* and *time of opinion*. That is, to extract what each opinion holder says about the different features of every object mentioned and also when in time this happened. This is clearly not a trivial task since even identifying just, for instance, the different objects can be difficult on its own.

Due to the complexity of the task most research has been focused on either some subproblem or simplification of this general problem such as review summarization or the analysis of comparative sentences. The area that has been researched most however is *sentiment and subjectivity classification* where the problem is reduced to ordinary text classification. Sentiment classification refers to the problem of classifying a given opinionated text as positive or negative and subjectivity classification deals with the task of distinguishing between opinionated and non-opinionated texts. Two reasons why this has attracted so many researchers are that ordinary machine learning techniques can be applied and that training data are available in the form of on-line reviews. A popular dataset used in many reports consist of reviews from the Internet Movie Database (IMDb)[2] and was created by labeling each review according to its rating [17]. The results presented using this dataset are extremely good, over 80% accuracy on the sentiment classification task, using relatively simple approaches. These results were however obtained when only considering reviews from one domain, movie reviews, and the same techniques applied on multi-domain datasets performed considerably worse [1].

Another, completely different, approach to sentiment analysis is with the use of vector space models which also comes from more traditional text processing tasks. In information retrieval for instance, it is common to build a so called term-document matrix where the row vectors represent words and the columns documents. The similarity between two documents, or between a search query and document, can

[1] In [11] it is stated that at least 20 to 30 companies dealt with sentiment analysis at the time (2010) in USA alone and well over 100 reports about the topic are mentioned as references.

[2] http://reviews.imdb.com/Reviews/

then be calculated as the distance between its corresponding column vectors [26]. By instead regarding the row vectors the similarity between words can be calculated and models used for this is sometimes referred to as word space models [9]. These are the models used in sentiment analysis by, for example, manually creating two points in the high-dimensional space, one representing positive attitude and one negative, and calculating which point a given text or word is closest to [25]. The problem with word space models created as above is that they tend to get very large, the term-document matrix will in fact have as many columns as documents and rows as the number of unique words. Furthermore the matrix will be extremely sparse since most words do not co-occur with that many different other words. To counter these problems some kind of dimension reduction technique is usually applied, such as Singular Value Decomposition (SVD) [6], but these are computationally costly and become infeasible to use on the large amounts of data in social media. Random Indexing is a method that has gained popularity in recent years because it produces similar results as SVD but is both more efficient and can be constructed incrementally [18]. Word space models and Random Indexing will be discussed in more detail in chapter 2.

1.2 Problem specification

This thesis was performed at Lissly, a company that provides their customers with a tool that searches all the big social media sites for specific keywords and presents relevant statistics of the result. The current statistics only show how many posts that are written about a keyword and not what the authors of those posts actually say. The purpose of this thesis is to investigate the use of sentiment analysis to capture the writers' opinions. More precisely given a text, written in Swedish, to try to classify it as positive, negative or neutral (no opinion). Due to the lack of labeled data from actual social media, reviews from different domains will be used to train and test the classifier. The idea being that the issues with cross-domain reviews should be quite similar to the problems with social media posts. A smaller set of hand labeled texts from the real data will then be used to evaluate the final classifier.

To achieve higher domain independence than existing methods the representation of a word space model built with Random Indexing will be used. A classifier using standard machine learning techniques will be trained on features extracted through the model to answer basically the following questions:

- Can Random Indexing be used to aid in the creation of a computationally efficient, domain-independent method of classifying texts based on the expressed opinions?
- Do the issues with cross-domain review classification transfer to classification of posts made in social media?

Chapter 2

Theory and previous work

This chapter will introduce the theory behind the methods used to perform sentiment and subjectivity classification and give a description of the methods themselves. In general there are two main directions of the field; data-driven and knowledge-based approaches. Data-driven means that only the actual data are taken into account whereas knowledge-based relies on some existing knowledge of the language. For instance one might argue that adjectives carry more opinions than nouns and create classifiers based on that. This will be followed by the theory of word space models and Random Indexing and how they are used in sentiment analysis. The main point of these models are that according to the *distributional hypothesis* [18, 19, 20, 26] the meanings of words can be extracted by the contexts in which they occur. By sampling a large set of unlabeled documents the meanings of words can be translated into distances in a high dimensional space which is mathematically well defined and easier for a computer to deal with.

In order to assimilate all the information some understanding of machine learning terminology and techniques is required, so first follows a quick briefing on that. If you are already familiar with basic machine learning you can skip the next section.

2.1 Basic machine learning

In machine learning you often speak of *supervised* and *unsupervised* learning where in supervised learning you have labeled training data, where the correct label is given for each training example, and in unsupervised you do not. In this report mainly supervised learning will be discussed and in most such methods you begin by defining *features* derived from the training data. The features can be integer or real numbers which are combined into a *feature vector* which is used as input to the classifier. During training the algorithms are presented with a set of feature vectors

combined with their corresponding labels to learn a *decision rule*. Once trained, given an unlabeled data point, the classifier uses the decision rule to try to assign the new point to the correct class. For a concrete example see section 2.2. [12]

One supervised learning method commonly used in text classification tasks is the *Naïve Bayes' Classifier* which uses statistics and probabilities to make the decision rule. Given a feature vector $(F_i, \ldots, F_n)$ and a set of classes C we want to calculate $P(C_j|F_1, ..F_n)$ for all $C_j \in C$. That is to calculate the conditional probabilities that the data point belongs to each class given the feature vector. Using Bayes' theorem this becomes $\frac{P(F_1,...F_n|C_j)P(C_j)}{P(F_1,...,F_n)}$ where $P(F_1, ..., F_n)$ is the same for all classes and can be disregarded leaving only $P(F_1, ...F_n|C_j)P(C_j)$. $P(C_j)$ is called the prior probability and can easily be calculated by simply counting the number of points belonging to the different classes in the training data. Calculating $P(F_1, ..., F_n|C_j)$, called the class-conditional probability, is harder and becomes infeasible for larger values of n. This is solved by assuming, naïvely, that the feature values $F_1, ..., F_n$ are conditionally independent of each other giving; $P(F_1, ..., F_n|C_j|) = \prod_{i=1}^{n} P(F_i|C_j)$, which is easier to calculate.[12]

The final decision rule is then to choose the class C_j which maximizes the following expression:

$$\prod_{i=1}^{n} P(F_i|C_j)P(C_j)$$

Another method often used is the *Support Vector Machine* (SVM) which also works on feature vectors to solve classification tasks. The idea behind SVMs comes from the realization that there are possibly an infinite number of decision rules that discriminate between two classes, and tries to formalize why one is better than another. The solution is the rule that produces the largest *margin*, that is the rule that maximizes the minimum distance between any data point and the discriminating line given by the decision rule. The points that lie on the margin are called *support vectors* and, after the training is done, these are the ones used to make classifications by simply calculating on which side of the line between them the new data point lies. To achieve this the problem is formed as a quadratic optimization which can be solved efficiently by existing solvers. Apart from finding the "best" decision rule SVMs have more properties which make them interesting. To allow for mislabeled data points during training, possibly due to noise in the training data, *slack variables* are introduced. These come with a cost parameter which lets you make a trade off between larger margin and training error. SVMs also incorporate something called the *kernel trick* which in short enables you to efficiently transform the data to higher dimensions which might be necessary to be able to do the classification. The interested reader is referred to a textbook on the subject like [12] for a more in-depth description.

The goal of using supervised learning methods is of course not only to be able to classify the training data correctly but to perform well on all possible inputs. This

is referred to as *generalization* and is the reason why a larger margin in the SVM is preferred since it should reasonably predict the class of new data points better. The opposite of generalization is called *overfitting*, where the learning algorithm produces a too complex decision rule to fit to the training data than to perform well on the data it is intended for. In the case of a SVM this could be either to use a too high cost parameter or to, unnecessarily, transform the data to higher dimensions. To be able to tell when overfitting occurs you need to measure something else than the error of the training set. One method of doing this is called *cross-validation* where you partition your training data into two separate sets, one used for training and one for testing. By measuring the error on the test set for different choices of parameters to the classifier you can find the optimal settings. In many cases however, training data are hard to acquire and there is a risk of undertraining the classifier if a too small set is used for training. Then *K-fold cross-validation* is usually used where the training set is split into K subsets and the classifier is trained on the $K-1$ subsets left leaving each one out in turn for testing.[12]

To evaluate the performance of a classifier a number of different metrics are usually used and the most common ones are *accuracy*, *precision* and *recall*. They can all be expressed in terms of true positives t_p, true negatives t_n, false positives f_p and false negatives f_n:

$$accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$
$$precision = \frac{t_p}{t_p + f_p}$$
$$recall = \frac{t_p}{t_p + f_n}$$

Positive and negative here does not refer to positive or negative sentiment but just to two general classes in an arbitrary classification problem. Accuracy simply measures the fraction of correct classifications and is useful when the classes are balanced, that is when there are an equal amount of examples in each class. If the classes are unbalanced however, accuracy provides less information since if one class consists of for instance 90% of the examples, simply assigning all examples to that class would give an accuracy of 90%. In that case it is often useful to instead measure how well the classifier performs on examples in and actually classified as belonging to the smaller class, which is what precision and recall measures.

2.2 Bag of words

The first step of any classification task is to decide on how to represent the relevant objects and in text classification the incomparably most common approach is to use a bag of words representation. The name *bag of words* refers to that in such a representation a text is simply regarded as a set or, in natural language, bag of independent words. This is sometimes criticized since it ignores any semantic or conceptual information about the words coming from, for instance, the order in which the words occur. Even with these shortcomings however, bag of words representations have been used to yield good results in both ordinary text, as well as sentiment and subjectivity, classification tasks.

2.2.1 Lexicon based methods

A simple approach to perform sentiment and subjectivity classification relies on the assumption that certain words, mostly adjectives, are exclusively used to express strong positive or negative sentiment. It is quite easy for a human to create two lists which only contain such words, positive in one and negative in the other. By simply counting the number of positive and negative words the classification can be made by assigning a given text to the class with the highest count. This was tested in [17] by letting two human test subjects independently select around ten words of each class and evaluating the resulting classifiers on the IMDb-dataset mentioned in the background section. The results achieved were 58% and 64% accuracy for the two respectively with 75% and 39% ties (same number of positive and negative words). The relatively poor results and high tie count could be due to the low number of words used but as another test they also tried the same number of words but selected by inspecting the training data. With these words they achieved 69% accuracy with only 16% ties which lead to the conclusion that even if humans select words that look plausible a data-driven approach could probably do better.

One improvement of the just described method is to use the manually created word lists as seed words and extend the list by adding other semantically related words. There are several ways of finding these related words but one of the most used is through WordNet[14]. WordNet is basically a dictionary of the English language constructed in a way that makes it easy to access programmatically. One method utilizing this is described in [8] where the lexicon is used to perform sentence level subjectivity and sentiment classification as a part of a review summarizer. The method only considers adjectives and as a first step a complete list of these are extracted from the review collection. A set of seed adjectives which have been manually assigned a sentiment orientation is then used together with WordNet to predict the orientation of the others. The prediction is done by extracting all synonyms and antonyms of the unassigned adjective and checking if any of those are in the seed set. If a synonym is found, the prediction becomes the same as the

seed word and if it is an antonym the opposite is used. This is executed for all unassigned adjectives and when a prediction is made the adjective is added to the seed set thus continuously growing the lexicon. The entire process is repeated until all adjectives are assigned an orientation or the growth stops.

Taking the idea even further it is possible to not only assign an orientation to the words in the lexicon but also some kind of weight or score. This is motivated by the fact that different words can express different of levels emotions. The word *great* for instance expresses a stronger positive sentiment than the word *good*. A simple decision rule to classify a text is then to sum all scores and apply a threshold instead of counting occurrences. An example of this is presented in [2] which also deals with review summarization and the proposed method of lexicon construction is heavily inspired by the one described in the previous paragraph. One difference is that the produced lexicon contains all words in WordNet and not only a selected set of adjectives but the main addition is in the procedure to extend the lexicon. At the start of the algorithm a score vector is constructed such that all positive seed words have a value of $+1$, all negative -1 and the rest a value of 0. Then, instead of just setting the score of found relations to $+1$ or -1, each value is updated by adding the score of all synonyms and subtracting the score of antonyms[1]. This way the orientations of the seed words are propagated through the synonym and antonym relationships causing non-seed words with at least one relation to a seed word to get its value updated. By repeating the process the updates spread to the entire set of words and finally producing the finished lexicon.

2.2.2 Data-driven methods

In contrast to the lexicon-based methods discussed in the last section where individual words are ranked or assigned an orientation by prior knowledge of the language, the data-driven methods only rely on information in labeled training data. The goal is however basically the same, to assign each word with a weight to be able to create a decision rule that discriminates between the classes. To find these weights and the decision rule supervised training techniques such as Support Vector Machines (SVM) or Naïve Bayes' Classifiers (NB) are used. Each document d in the training and test sets can be expressed by the feature vector $f = (n_1(d), n_2(d), ..., n_m(d))$ where $n_i(d)$ is the number of times word i occurs in document d. This technique is tested on the IMDb-dataset in [17] achieving accuracies of 78.7% with a NB classifier and 72.8% using a SVM, both clearly outperforming the simple approach with seed words.

Using the frequency $n_i(d)$ is however not the only, or the best, way to create the feature vector from a document. By simply using binary features, a 1 if the word is in the document and a 0 otherwise, the results are improved to 81.0% using NB

[1] A scale factor $\lambda < 1$ is multiplied with the scores before the addition or subtraction in order to control the rate of propagation.

and 82.9% with SVM on the same dataset [17]. In topic based classification and information retrieval a metric called Term Frequency–Inverse Document Frequency (TF-IDF) is commonly used. TF-IDF gives a term (word) a higher weight if it occurs frequently in the given document but is infrequent in the whole document collection. This is to increase the importance of words that statistically are well suited to discriminate between the documents but this does not seem to work for sentiment classification [13]. Instead a slightly modified version called Delta TF-IDF is proposed where the terms are weighted by the difference in frequency of the term in the positive and negative training set. With $C_{t,d}$ being the number of times term t occurs in document d, P_t and N_t the number of positive and negative documents with term t and $|P|$ and $|N|$ the number of positive and negative documents, the feature value $V_{t,d}$ for term t in document d becomes:

$$V_{t,d} = C_{t,d} \cdot \log_2 \left(\frac{|P|}{P_t} \right) - C_{t,d} \cdot \log_2 \left(\frac{|N|}{N_t} \right) = C_{t,d} \cdot \log_2 \left(\frac{|P|}{P_t} \frac{N_t}{|N|} \right)$$

Using this feature value and a SVM an accuracy of 88.1% is reported[13][2].

Another important part of the data-driven methods, besides using an appropriate feature value, is feature selection. Using all unique words in the training data as separate features can lead to an extremely high-dimensional feature space which is both computationally costly and can lead to overfitting. Since every extra feature means another free parameter for the supervised learning method to estimate a more complex decision rule can be made. To deal with this feature selection is used where the words are ordered according to some statistical metric and only the top scoring ones are kept for learning and classification. A number of different metrics are used in text categorization and for a extensive description and comparison the reader is referred to [7]. One of the methods described there is called Information Gain (IG) which is extensively used in a wide range of machine learning tasks. IG measures the decrease in entropy of the dataset when the value of a feature is given contra not given. Entropy describes the amount of impurity in a dataset S and is, for a two class problem, defined as [12]:

$$E(S) = -P_{s_1} \log_2 P_{s_1} - P_{s_2} \log_2 P_{s_2},$$

where P_{s_1} and P_{s_2} are the probabilities of at random drawing an example of class 1 or class 2 respectively. With this definition of entropy the Information Gain $IG(S, f)$ of feature f in the dataset S becomes [12]:

$$IG(S, f) = E(S) - \frac{|S_{f=1}|}{|S|} E(S_{f=1}) - \frac{|S_{f=0}|}{|S|} E(S_{f=0}),$$

where in our case $S_{f=1}$ is the subset of documents containing word f and $S_{f=0}$ the subset that do not.

[2]It is not completely clear in [13] if any other techniques were applied to achieve this result.

2.2.3 Cross domain issues

While the supervised machine learning techniques usually outperform the lexicon-based ones the increased performance comes at a cost. Classifiers trained on data from one domain do generally not perform well when applied to data from another[1]. This is because the learning algorithm only looks at the actual data and can assign strong positive or negative sentiment to words that in general carry no sentiment at all. It is even possible that a word that strongly indicates positive sentiment in one domain actually means something negative in another. For instance, it is a good thing for a movie to be *unexpected* but the same word is probably less than the desirable when talking about a car's steering. When only regarding one of the domains it is clearly beneficial to assign a sentiment to the word whereas in general it is not. Lexicon based methods, using only words with a known sentiment orientation, is therefor more domain independent while supervised learning, which utilizes more information, performs better within domains.

The goal is obviously to achieve both domain independence and high performance and one solution is to combine the two techniques. This is done in [2] where they take two different scores obtained through a lexicon and, if present, a rating and use those as features to a Maximum Entropy classifier[3]. Another solution is to use the parameters and possibilities at hand in the supervised learning algorithms and try to overcome the cross domain issue. This is extensively studied in [1] where they take reviews from four different domains and perform a number of cross domain classification tasks. The first obvious attempt is to simply train a classifier on a mixture of data from different domains which was done using a Support Vector Machine (SVM) and a feature selection metric called LLR[4]. Another possibility is to use a small amount of labeled data from the target domain combined with the full sets of the others. This is done by training multiple classifiers on the different domains and parameters and combine them in an ensemble weighting each classifier's prediction using the target domain data. Using this approach they managed to significantly improve the results[5].

The best performance reported in [1] however was achieved using a method which only relied on a small amount of labeled and a larger amount of unlabeled data from the target domain. The algorithm used was a generative Naïve Bayes' Classifier where the parameters were iteratively updated using Expectation Maximization. The full details will not be discussed here but are available in [1] and [15]. The interesting conclusion is however that the algorithm which could make use of the information in unlabeled target domain data outperformed the others[6].

[3]The Maximum Entropy classifier is another supervised machine learning technique which relies on statistics and probabilities.

[4]LLR, log likelihood ratio, is equivalent to Information Gain.

[5]Using 100 examples from the target domains the accuracy was increased by around 6 percentage points in three of the domains and 1.5 in the other.

[6]Increasing the accuracy of the ensemble approach by 1 to 6 percentage points in the different

2.2.4 Utilizing the context

As mentioned earlier the bag of words approach is sometimes criticized because it disregards all information about the context contained in written text. One of the most obvious examples of this in sentiment analysis is with the use of negation. All the methods discussed this far would treat the word *great* exactly in the same way independent of it was preceded by *not* or *very*. This is clearly a mistake since in reality the phrases have opposite sentiment orientations. A simple way of dealing with negations in the bag of words model is to either prepend each word between any negation and punctuation terms with a special tag [17] or reverse the sign of those words values in a lexicon [2]. In the first case however, the classification performance actually dropped when those tags were added. There are of course more advanced methods of dealing with negation, in fact there are whole reports like [5] aimed at just that subject.

Another way of inferring contextual information into the model is to not only look at each separate word (unigrams) but instead at every word pair or triplet (bigrams and trigrams) or even tuples of every possible length (ngrams). This of course increases the number of features drastically which makes feature selection even more important. These are tested (some or all of them) as features to supervised machine learning algorithms in [1, 13, 17] with varying results. In [1] they achieve the highest reported accuracy on the IMDb-dataset, 90.45%, using the top 20000 ranked ngrams according to LLR but also notice that when tested across domains unigrams still perform better than the others.

There are also methods to capture even more complex parts of written language such as grammatical information. One example of that is the use of the information from a position of speech (POS) tagger[7] to associate each word in a text with its corresponding word class. A straightforward way to incorporate this information in a bag of words model is to prepend every word in a text with its word class. This was tested in [17] but did not yield any improvements. A more advanced way of using grammatical information is called appraisal groups where one tries to extract groups of appraisal containing detailed information about how it was expressed. Using such groups based around only adjectives combined with bag of words features the authors of [27] manage to achieve an accuracy of 90.2% on the IMDb-dataset[8].

target domains

[7]There are several POS taggers freely available but details about such algorithms was out of scope for this project.

[8]The method was not tested on any cross domain tasks.

2.3 Word space models

As mentioned earlier the main idea behind word space models is to represent words by high dimensional vectors where semantically related words will have vectors closer to each other than words that are not. This idea comes from the *distributional hypothesis* which in essence states that words that occur in similar contexts will have similar meaning[18, 19, 20, 26]. This hypothesis is supported by both numerous experimental results, see section 2.3.4, as well as more philosophical arguments [10]. The power of these models is that they make semantics computable in a mathematically well defined way without relying on any previous knowledge about the language [18]. By only looking at the actual data they also only capture what is really there and not anything else.

2.3.1 Traditional methods

The normal way of constructing a word space model is to go through the data and create a co-occurrence matrix, for instance the term-document matrix mentioned in the background used for information retrieval. The context used, as discussed in section 2.3.3, does not have to be a document however so in general it is a term-context matrix. The matrix M is created such that each row M_w corresponds to a word and each column M_c to a context [18]. So if word w occurs four times in context c the value of the cell M_{wc} would be four. When the entire set of data has been processed a words row vector, or context vector, will contain exactly which contexts the word has occurred in. As a final step the values of the context vectors are usually normalized and weighted to handle high frequency words and possibly contexts of different lengths. By the distributional hypothesis it is then easy to calculate the semantic similarity between two words by simply measuring the similarity between their context vectors. There are numerous well known ways of measuring the similarity between two vectors, A and B, and one of the most commonly used is called Cosine similarity, $sim(A, B)$, which calculates the cosine of the angles between them. Using the Euclidean dot product formula this becomes:

$$sim(A, B) = \frac{A \cdot B}{\|A\|\|B\|},$$

where $A \cdot B$ is the dot product of A and B and $\|V\|$ is the magnitude of the vector. The similarity between contexts can of course also be calculated in the same way by regarding the columns of M. Such a column vector is actually exactly what is used as feature vector in the bag of words methods described in section 2.2.

While word space models built in this manner have interesting properties they also come with some problems. As mentioned in the background the context vectors will have as many dimensions as the number of contexts which for real life problems

could easily be in the millions[9]. This high dimensionality makes the model computationally difficult to handle. Furthermore according to Ziph's law [18] most of the words in any natural language will only occur in a very limited set of contexts which makes the term-context matrix extremely sparse. In fact, in a typical word space model matrix more than 99% of the cells will have a value of zero [18].

To handle the problems with high dimensionality and sparseness most methods using word space models apply some kind of statistical dimension reduction technique. There are several different ways this can be performed but one well known example is Singular Value Decomposition (SVD), which is used in Latent Semantic Analysis (LSA) [6]. Most of these techniques however require that the entire term-context matrix is created before it can be transformed to something more manageable. This still leaves the problem with initial high dimensionality and for large datasets it can be infeasible to actually perform the SVD [18]. Another problem is that once the dimension reduction is applied it is difficult to add new data to the model which often means that you actually have to redo the entire process. These problems, among others, make such word space models impractical for applications dealing with large amounts of data such as analysis of social media.

2.3.2 Random indexing

Random Indexing is another method of creating word space models which tries to overcome the issues with the normal approach and still produce similar results. This is achieved by instead of initially creating the term-context matrix accumulate context vectors of a predefined dimensionality as the data are processed. By doing so the large initial matrix step is avoided with an implicit dimension reduction and allows the model to be updated incrementally when new data are available. The name *Random Indexing* comes from the fact that the dimension reduction is performed by projecting the data into a randomly selected subspace. The motivation for why this works will be discussed later in this section. [18]

The process of creating a word space model with Random Indexing is relatively simple, it consists of two steps. First each context is assigned a unique randomly generated vector, called index vector. The index vectors are sparse and consists of a small number of ones and minus ones randomly distributed with the rest of the values set to zero. The dimensions of these vectors are usually around a few thousand and they can of course be generated on the fly when new contexts are discovered. In the second step the data are processed and every time a word occurs in a context the index vector of that context is added to the context vector of the word. This way the context vectors are still effectively the sum of the contexts in which the words occur. [18]

[9]The largest dataset used in this project consists of about two million documents and 140 million words.

As an example, consider the two following documents:

This is document one about apples.
The second document is about oranges.

If we assign the documents index vectors, i_1 and i_2, as:

$$i_1 = (\ \ 0, +1, -1, \ \ 0)$$
$$i_2 = (-1, \ \ 0, \ \ 0, +1)$$

the context vector for each word w, c_w, would be:

$$\begin{aligned}
c_{This} &= (\ \ 0, +1, -1, \ \ 0) \\
c_{is} &= (-1, +1, -1, +1) \\
c_{document} &= (-1, +1, -1, +1) \\
c_{one} &= (\ \ 0, +1, -1, \ \ 0) \\
c_{about} &= (-1, +1, -1, +1) \\
c_{apples} &= (\ \ 0, +1, -1, \ \ 0) \\
c_{The} &= (-1, \ \ 0, \ \ 0, +1) \\
c_{second} &= (-1, \ \ 0, \ \ 0, +1) \\
c_{oranges} &= (-1, \ \ 0, \ \ 0, +1)
\end{aligned}$$

That is, the words that only occur in document one (*This*, *one* and *apples*) get a context vector equal to i_1 and the ones that only occur in document two (*The*, *second* and *oranges*) a context vector equal to i_2. For the words that occur in both documents (*is*, *document* and *about*) the context vectors would first become i_1 as the first document is being processed and then added by i_2 resulting in $i_1 + i_2$ as the second is processed. If a word were to occur two times in a document that document's index vector would be added twice to the context vector.

To motivate why Random Indexing does indeed work, first consider using index vectors with a dimensionality as the number of contexts and consisting only of a single one at different positions. Then the index vectors will be truly orthogonal and the process will produce the ordinary term-context matrix normally used. It has however been demonstrated that there exist many more nearly orthogonal than truly orthogonal directions in high dimensional spaces so the randomly generated index vectors of random indexing will approximate orthogonality. Because of this near orthogonality the process can be viewed as projecting the data into a random subspace. When this subspace is of "sufficiently high dimensionality" it has been shown that the projection approximately preserves the distances between the points in the original vector space. Reducing the dimension while preserving distances is of course the purpose of any dimension reduction technique including SVD. Note that sufficiently high dimensionality above refers to order of thousands while the original vector space is in the order of hundreds of thousands or even millions so the reduction in dimension is still significant. [18]

While Random Indexing has been shown to perform well on many of the tasks ordinary word space models are used for, see section 2.3.4, it has some weaknesses. One of these is the lack of ability to find implicit connections between words that do not occur together in any context. This is shown in [4] where they also present an extension to Random Indexing which can also discover these connections by indirect inference. They call the method Reflective Random Indexing and the idea is to update not only the context vectors but also the index vectors. The first step is identical to normal Random Indexing using documents as contexts but when the context vectors are generated the randomly generated index vectors are replaced by the sum of the context vectors of the words in the corresponding document. Using these new index vectors the process is repeated to produce the final context vectors. The entire process becomes[10]:

1. Assign each document with a randomly generated index vector.
2. Accumulate the words' context vectors by summing the index vectors of the documents in which the words occur.
3. Replace the documents' index vectors with the sum of the context vectors of the words in the document.
4. Repeat 2.

This is actually one of the two variants presented of Reflective Random Indexing, namely document-based. The other is called term-based and works in a similar way but instead begins by assigning each word with a randomly generated index vector and then start the process at step 3. A big drawback of document-based Reflective Random Indexing is however that the created model is not capable of being incrementally updated but needs to be redone if new data are to be added.

2.3.3 Different contexts

All examples so far of word space models have used documents as context, as mentioned however this is not the only possibility. There are several other options [26] but the most common one, together with documents, is word windows. With word windows the context of a word is regarded to be, instead of the entire document, only the *window* of directly preceding and succeeding words. The size of the window may of course vary between applications but a common setting is to take two words before and two words after the focus word (a 2+2 window) [9, 10, 19, 20, 22, 24]. The original term-context matrix in a model using word windows actually becomes a term-term matrix. When the data are processed each word (the focus word) is looked at in turn and the cells of that word's corresponding row vector get incremented at the cells corresponding to the words in the window. With the same example documents as in the previous section and a 1+1 window the term-term matrix would be:

[10] Where step 1 and 2 alone is normal random indexing with documents as context.

	This	is	document	one	about	apples	The	second	oranges
This	0	1	0	0	0	0	0	0	0
is	1	0	2	0	1	0	0	0	0
document	0	2	0	1	0	0	0	1	0
one	0	0	1	0	1	0	0	0	0
about	0	1	0	1	0	1	0	0	1
apples	0	0	0	0	1	0	0	0	0
The	0	0	0	0	0	0	0	1	0
second	0	0	1	0	0	0	1	0	0
oranges	0	0	0	0	1	0	0	0	0

The matrix is still sparse and will usually have an even higher dimensionality than when documents are used as context so the need for a dimension reduction is still present. Luckily the same techniques can of course be applied. With Random Indexing it is just a matter of assigning each word with a randomly generated index vector and accumulate the context vectors in the normal way.

A fact that is sometime overlooked is the impact the choice of context has on the model and which semantic relations it captures. This is discussed extensively in [19] and [20] where the terms *syntagmatic* and *paradigmatic* are used to distinguish between the relations models using document and word window contexts capture. Syntagmatic relations exists between words that co-occur like for instance the words *plot* and *cast* which indicates that they are used to describe the same thing, a movie in this case. Paradigmatic relations conversely exist not between words that co-occur together but between words that co-occur with the same other words. A good example of words with this relation is adjectives that modify the same noun, like *hungry* and *thirsty*. The difference between the two relationships can be visualized by the following grid where the rows represent words with syntagmatic relations and the columns paradigmatic[11]:

	Paradigmatic relations			
Syntagmatic relations	she	adores	green	paint
	he	likes	blue	dye
	they	love	red	colour

Using documents as context captures more syntagmatic relations and word windows paradigmatic and at a first glance paradigmatic may seem more desirable. As we will see later however this is not necessarily the case when performing sentiment analysis.

Word space models are sometime criticized, just as the methods discussed earlier, for not taking the word order in to account. When using word windows as context it is however possible to add that information to the model as well. The basic idea is to represent the words in the window differently depending on where in the window they occur. This means that in a 2+2 window for instance each unique word would need to have four different representations, one for each position, which would produce an even larger term-term matrix. When using Random Indexing

[11]The example sentences are taken from [20].

this is not a problem since that matrix is never actually produced. A method of including this information with Random Indexing is presented in [22] which uses random permutations of the index vectors to encode the position.

2.3.4 Applications

As mentioned earlier one use of vector space models is to perform information retrieval by measuring the similarity between a search query and all documents in a collection. Since then word space models have evolved and been used successfully in a number of different applications. One of the most interesting uses which really indicates that such models do indeed learn the meaning of words is when applied on the synonym finding part of TOEFL[12]. In the test you are given a word and four alternatives and are supposed to select the synonym among them. Solving this using a word space model is done by calculating the distance between the given word and each of the alternatives context vectors and simply choose the alternative that is closest as the answer. Using this approach LSA[13] achieved 64.4% correct answers and Random Indexing with window contexts[14] 63.5%–72.0% compared to the average result of real non-English-speaking humans of 64.5% [10].

Two other related applications for word space models which are closer to the subject of this project are presented in [24] and [23]. They are called *Terminology mining* and *Buzzword monitoring* and are both implemented using Random Indexing and deal with data in social media. Terminology mining refers to the task of understanding and keeping up with the everchanging vocabulary used in social media. To do this a word space model is created using Random Indexing with word windows as context and word order encoding and it is evaluated by looking at the nearest neighbors of selected words. For instance, the nearest neighbors of the word *recommend* includes *reccomend*, *looove* and *lurve*. That is, probably, both deliberate and unintentional misspellings which would be of great importance when performing sentiment analysis but extremely hard to detect using, for instance, a predefined lexicon. There are also however some problems with this approach, among the closest neighbors of the word *bad*, for example, are *cool* and *fantastic*. This, on the other hand, would pose a real problem if trying to distinguish between positive and negative words and comes from the fact that antonyms are a kind of paradigmatic relation. For buzzword monitoring documents are instead being used as context to see what is generally said about a certain keyword. Since such a model would represent words as similar if they co-occur a lot you can see if, for instance, a product name is talked about in a generally positive or negative manner by calculating and comparing the distance between the name and the words *good*

[12]Test Of English as a Foreign Language, used to test foreign applicants' language skills when applying to universities.

[13]LSA is explained in section 2.3.1.

[14]Using word stemming and different window sizes.

and *bad*. This concept is not limited to only positive and negative but can be used for any comparison task where predefined *poles* can be defined.

Word space models have also been used to perform and aid in different text classification tasks. One such example is presented in [21] where they use a model created with document context Random Indexing as a feature extractor to a Support Vector Machine. The feature vector used is simply a weighted sum of the context vectors of all words in a document. They call the method "Bag of Concepts" since each dimension of the context vectors can be regarded as an abstract concept. This approach was tested on a set of news wire documents assigned to 90 categories and produced similar results as ordinary bag of words methods overall but outperformed them in a number of the categories. In [25] news headlines are classified as being loaded by positive or negative emotions in the same manner as with buzzword monitoring. The sum of the context vectors of eight seed words were used for each *pole* which was then compared to the sum of the context vectors in the headline. For this application a normal term-document matrix was used, that is, no dimension reduction was applied.

Chapter 3

Methodology

As stated in the introduction and problem specification the method used in this project will be based on a word space model built with Random Indexing. A brief motivation for that choice was also given but now enough background has been presented to motivate it more thoroughly. The goal of the method and entire project is to improve cross-domain classification of reviews and transfer that improvement to classification of general posts made in social media. There are several factors which make a word space model an attractive base for such a method. First, it has already been shown, see section 2.2.3, that methods which can utilize the information in target domain unlabeled data are likely to outperform methods that do not. Unlabeled data are, compared to labeled, available in great quantities so acquiring it is not a problem. Word space models have also been shown, in numerous applications, to actually capture the meaning of the words in contrast to the more mathematically supported method of utilizing unlabeled data used in previous work. Further, the intended domain, social media, is especially hard to analyze due to the type of language used. Misspellings, slangs and fashion words are common and the vocabulary changes over time. This makes knowledge-based method such as using a predefined lexicon or even grammatical constructs less suitable for the task. Word space models on the other hand have been shown to deal well with these issues and one might argue that a word space model actually can be seen as a dynamic lexicon of the language it is built on. Since the target language of this project is Swedish which does not have as many resources available, as for instance WordNet for English, this last property is especially important. In fact, a truly domain independent method should of course be able to handle different languages with as small effort as possible anyway. The choice of using Random Indexing to build the word space model can also be motivated by the fact that the method is intended for use on social media posts. Due to the extreme amounts of data produced and the changing vocabulary the computational efficiency and incremental construction of Random Indexing are very desirable properties. Last, the choice was made to actually perform classification on each post instead of, for instance,

just using the techniques of buzzword monitoring, see section 2.3.4. The problem with the latter approach is that it is very hard to actually verify or measure its results. Just stating that a product name is twice as close to the positive pole than the negative is harder to explain or verify than actually producing the positive and negative posts about it.

3.1 Datasets

Since the language primarily targeted in this project is Swedish, the datasets used in previous research, like the IMDb-dataset, cannot be used. However, one of the motivations for performing document level sentiment and subjectivity classification in the first place was that labeled data are available in the form of online reviews. So the first task was to actually acquire all needed datasets which include labeled data, positive, negative and objective, from five different domains. The chosen domains are movies, books, cellphones, games and online shops. The samples in the movie and book domain were taken from the Swedish sites Filmtipset[1] and Boktipset[2]. Reviews at those sites are rated on a five star scale and reviews with one or two stars were treated as negative and four or five stars as positive. Reviews with a rating of three were discarded and the objective samples were instead taken from plot summaries on the same sites. For the rest of the domains the positive and negative samples were taken from the review site Prisjakt[3]. There a ten star rating is used and by a frequency count of the different ratings it became apparent the users there were biased towards higher ratings. By inspection[4] the choice was made to treat all reviews with zero to five stars as negative and only reviews with eight to ten as positive. Finding objective samples in these domain proved difficult however so instead random articles from the Swedish Wikipedia[5] was used.

To have something to evaluate the final method, labeled data from real social media were needed. Two distinct sets were created by letting six humans, all working with analysis of social media, hand label posts actually collected by the tool at Lissly. For the first set each person labeled different posts to make the set as large as possible while for the other everybody labeled the same predefined smaller set of posts. This second set was used for comparing the method proposed here to how well actual humans agree on the classifications and the "correct" label of a sample was set by majority vote. The number of samples in each domain and class is presented in table 3.1.

As also seen in table 3.1 the classes are kept balanced for the training sets to make evaluating intermediate methods easier. Besides the already mentioned datasets

[1] http://nyheter24.se/filmtipset/
[2] http://www.boktipset.se/
[3] http://www.prisjakt.nu/ (An English version is available at http://pricespy.co.uk/.)
[4] Most of the reviews with six or seven stars actually contained mostly negative expressions.
[5] http://dumps.wikimedia.org/svwiki/

Set\Class	Pos	Neg	Obj
book	4k	4k	8k
shop	4k	4k	8k
movie	10k	10k	20k
cell	1k	1k	2k
game	800	800	1.6k
real1	157	190	361
real2	23	9	68

Table 3.1. Number of samples in each set and class.

a larger set of unlabeled posts from social media was also created to be used to build the final word space model and test the scalability of the method. This set consists of about two million posts, 140 million words in total and about 2 million unique words. For the initial tests on the five domains the word space model was instead built on all available data from the domains (without using the labels) which consists of 79200 documents, 7 million words and about 200000 unique words. All data were collected during April and May of 2012.

3.2 Baseline

To have something to compare the final method to and make sure the results are comparable with the results in earlier research a baseline was established. For the baseline a Support Vector Machine (SVM) was trained on unigram features using Information Gain as feature selection metric with presence or Delta TF-IDF as weights. The process of training and testing this classifier is easy to automate so a large number of tests with different combinations, feature selection limit ranging from all to 1000 with both weighting schemes, were performed and the best setup and result recorded. Larger ngrams such as bigrams or trigrams were not used since it would have added to the complexity and the time required for the tests and they have already been shown to not yield any improvements for cross-domain classification. The SVM was chosen over, for instance, the Naïve Bayes classifier because of its extra features which will be needed for later tests and using the same method in all tests makes comparisons easier.

Two separate tests were executed as part of the baseline, the first to establish that there actually exist cross domain issues with this normal approach. To do this the SVM was trained as described above on each domain in turn and tested in both its own domain and the remaining others. The second test made the actual baseline used for comparison by using the naive approach of training on a mixture of domains. Leaving one domain out in turn for testing the SVM was trained on the others and the results recorded. In these cases not all the available data were used but an equal amount from all the remaining domains so that each domain

would influence the decision rule equally. All tests were performed on both the subjectivity (subjective – objective) and sentiment (positive – negative) classification tasks.

3.3 Random Indexing

While the decision to use a word space model built with Random Indexing is taken there are still a number of choices and questions left. First, as mentioned in section 2.3.3, the choice of context greatly affects which relations the model will actually capture. This will in turn of course affect the performance of the resulting classifier. Secondly, parameters like the dimensionality and the number of nonzero entries of the context vectors must be decided along with any preprocessing of the data such as for instance stop word removal. Finally, once a model is built that contains the desired relations the question of how to actually use the information in the model to create a classifier remains. A couple of the methods of doing this have been proposed in earlier research, see section 2.3.4, and yet another one is presented here.

3.3.1 Building the model

So far two different types of contexts for the word space model have been discussed, documents and word windows plus the two variants of Reflective Random Indexing, document-based and term-based. In previous research the syntagmatic relations of using documents as context have been proposed for the tasks most similar to the one investigated in this project [21, 25]. Word windows and paradigmatic relations have however been proved to be more suitable in tasks where the actual meaning of words are dealt with like terminology mining and the TOEFL test [10, 23, 24]. The ability to capture the actual meaning seems intuitively like a desired property and even with the problem of also capturing antonyms the choice cannot be discarded. Luckily it is a small task to change context type in a Random Indexing implementation so all the different choices could be implemented and evaluated experimentally. As proposed in previous research a quick way of seeing some properties of a model is to look at the nearest neighbors of selected words. Since we deal with sentiment classification and are trying to distinguish between positive and negative attitude the Swedish words for “good” and “bad” (“bra” and “dålig”) seem like good choices. Word space models with the different context types were created using the smaller dataset described in section 3.1 and the nearest neighbors calculated using cosine similarity. The top ten neighbors when using documents as context are presented in table 3.2 and with word windows in table 3.3.

Bra (Good)	Similarity	Dålig (Bad)	Similarity
är (is)	0.3669	den (it)	0.1979
så (so)	0.3318	är (is)	0.1940
inte (not)	0.3223	på (on)	0.1910
men (but)	0.3204	jag (I)	0.1876
det (it)	0.3145	det (it)	0.1871
med (with)	0.3143	inte (not)	0.1852
att (that)	0.3137	så (so)	0.1850
på (on)	0.3003	telefon (telephone)	0.1803
mycket (very)	0.2957	för (for)	0.1784

Table 3.2. Top 10 related words using cosine similarity of "bra" and "dålig" in a word space model built using documents as contexts with Random Indexing. English translations in parentheses.

Bra (Good)	Similarity	Dålig (Bad)	Similarity
dålig (bad)	0.9215	kass (stinks)	0.9255
intressant (interesting)	0.8969	bra (good)	0.9215
underhållande (entertaining)	0.8872	usel (terrible)	0.8940
rolig (funny)	0.8870	grym (awesome)	0.8687
dåligt (bad)	0.8801	dåligt (bad)	0.8611
imponerande (impressive)	0.8501	rolig (funny)	0.8506
ganska (fairly)	0.8469	läskig (scary)	0.8425
kass (stinks)	0.8456	jättebra (very good)	0.8362
usel (terrible)	0.8434	ganska (fairly)	0.8347
uselt (terribly)	0.8419	skön (sweet)	0.8250

Table 3.3. Top 10 related words using cosine similarity of "bra" and "dålig" in a word space model built using word windows as contexts with Random Indexing. English translations in parentheses.

The results are as expected, with word window contexts we get words that look semantically related but positive and negative words are closely related and with document contexts we get mostly stop words. The stop words will of course be related to everything since by definition they occur in almost every document. So by only looking at the nearest neighbors neither of these contexts seem to be ideal for our purpose. Turning then to the variants of reflective random indexing, using the document-based approach everything got closely related to everything and since that also removes the ability to update the model incrementally that option was not pursued any further. However, using the term-based approach actually yielded some interesting results. Doing this is effectively the same as using window contexts but with an infinite window size and can be seen as a sort of mix between document and word window contexts. The nearest neighbors in this model are presented in table 3.4.

Bra (Good)	**Similarity**	**Dålig (Bad)**	**Similarity**
batteritid (battery life)	0.9215	knapparna (the buttons)	0.9255
skärm (screen)	0.8969	så (so)	0.9215
ljud (sound)	0.8872	skärmen (the screen)	0.8940
kamera (camera)	0.8870	funktionen (the function)	0.8687
överlag (mainly)	0.8801	touch (touch)	0.8611
hyffsat (fairly)	0.8501	inte (not)	0.8506
riktigt (really)	0.8469	kameran (the camera)	0.8424
knappsats (keypad)	0.8456	batteritiden (the battery life)	0.8362
flyter (flows)	0.8434	batteriet (the battery)	0.83478
desgin (design)	0.8419	mobil (cell phone)	0.8250

Table 3.4. Top 10 related words using cosine similarity of "bra" and "dålig" in a word space model built using term based Reflective Random Indexing. English translations in parentheses.

The nearest neighbors in the model built with term based reflective random indexing are indeed very interesting. The model seems to have been greatly influenced by the cell phone reviews even though there are many more reviews of books and movies in the dataset. Most of the nearest neighbors are features of a cell phone and even though the same features are present as neighbors to both "good" and "bad" the form of the word is different. Apparently when talking about a feature in indefinite form we are usually more positive and in definite more negative, in Swedish reviews at least. There are numerous examples of this in table 3.4, "battery life" and "the battery life", "screen" and "the screen", "camera" and "the camera" and so on. This would clearly not be the case if the model was built with English reviews since the forms are identical except the preceding "the" which would not affect the model.

The top ten nearest neighbors do obviously not show all the properties of a word space model but are good as a pointer. To actually verify that the conclusions made from that experiment hold when performing classification, experiments using all techniques described in the next section were performed using all the different context types as well. Those initial experiments did in fact also show that the word space model built with term based reflective random indexing seems to be best suited for the purpose of this project. During these initial experiments the other parameters, like dimensionality and number of nonzero entries of the context vectors, were also evaluated and the results were similar to what has been presented in previous research. A dimensionality of 1500 proved to be sufficient using eight nonzero entries. Stop word removal was also evaluated since it is common practice to remove such words before actually building a word space model. A normal way of selecting stop words is to simply use a frequency count and remove, for instance, the top 1% most occurring terms. The problem with this approach is that among them are usually a lot of sentiment bearing words like "good" and "bad". To counter this

an ad-hoc solution was used, and proved to improve the model, which instead used the top 1% most common words that were not in the top 1000 closest neighbors of the words “good” or “bad”.

3.3.2 Using the model

To clarify, the final word space model used was created using term based reflective random indexing, with 1500-dimensional context vectors and eight non-zero entries and the ad-hoc stop word list described in the last section. With this decided the best way of actually using the information in the model must be evaluated. The first tested method of doing this was with the use of a predefined positive and negative pole consisting of seed words and calculating the distance between a document and the poles, described in section 2.3.4. This can be seen as similar to the lexicon based approaches described in section 2.2.1 and should be good at domain independence since no actual training data are used. This might also have been the case but the results were still inferior to the baseline results. A number of different variations of this approach were evaluated, such as different numbers of seed words, seed words generated manually or from training data and even different ways of calculating the distance between the documents and the poles. The best separation of the classes was achieved by taking the average of the distance between each word and its closest neighbor in both poles. The results and the problem with this approach is visualized in figure 3.1.

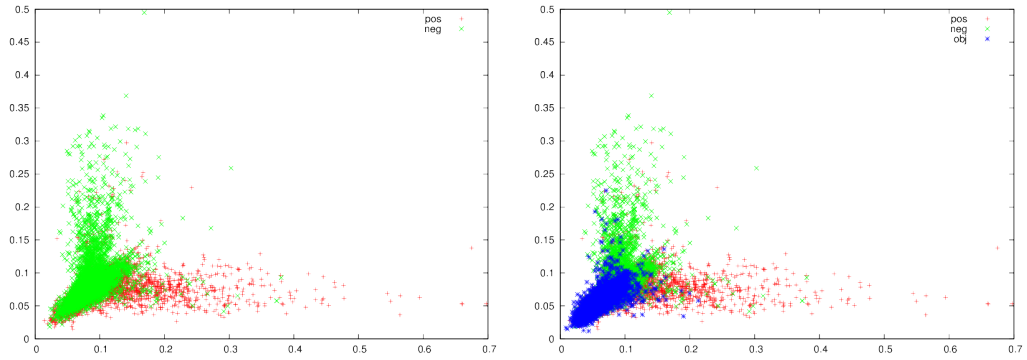

Figure 3.1. Class separation of positive and negative samples (left) with objective samples included (right) using eight seed words and the average of closest distances. X-axis represents the similarity with the positive pole and Y-axis similarity with the negative.

As seen in figure 3.1 some positive and negative samples are separated correctly but the majority of the samples are right on top of each other together with the objective samples around the line $x = y$. While this could be used for a high precision but low recall task by using appropriate thresholds the results are not good enough. Looking for improvement we turn to another method described in

section 2.3.4, namely the one using "Bag of Concepts". The idea there is to use the word space model as a feature extractor and represent each document by the sum of its context vectors. By doing so we have 1500-dimensional feature vectors which are used to train a Support Vector Machine (SVM) to be able to make use of the available training data. This constitutes a major dimension reduction, from 200000 to 1500, and as proposed in the paper presenting the method a polynomial kernel was used in the SVM, see section 2.1. This approach did indeed yield good results and beat the baseline in almost all tests.

Even though the previously described method did produce good results it seems like a crude way of utilizing the information in the word space model. The actual values of the context vectors, which are summed in that method, do not seem to carry as much information as the distances between the context vectors. Specially interesting is the distances between the context vectors of arbitrary words to context vectors of words with known polarity. To test this idea a new method is proposed which can be seen as a mixture of the previously described methods. Again an SVM with a polynomial kernel is used to utilize the information in the training data but feature vectors of even lower dimensionality n are used. Each dimension in these vectors represents a seed word sw_i, either positive or negative, but known by some measure to be a good indicator of sentiment. For a given document D the value of the cell f_i in the feature vector f is set to be the average distance of all the words in D to the seed word sw_i. This can be realized by matrix operations, if we denote the dimensionality of the context vectors d and the number of words in D m, we can create the m by d matrix A and the d by n matrix B. The rows of A are set to the normalized context vectors of all words in D and the columns of B are set to the normalized context vectors of the seed words. Then the m by n matrix C defined as $C = A \times B$, will contain the cosine similarity between all words in D and the seed words. By simply averaging over the columns of C, the feature vector f is obtained.

Using this method with 200 seed words selected as the top 200 nearest neighbors of the words "good" and "bad" in the word space model even better results were achieved. It did not beat the "Bag of Concepts" method by much but performed slightly better[6] on all tests. Given that this method performs marginally better on the training data while using even fewer and more well defined dimensions compared to the "Bag of Concepts", this new method should be preferred. Furthermore, since the dimensions actually relate to words, in contrast to an abstract concept, any of the weighing schemes discussed in section 2.2.2 could be applied. This was not exhaustively tested but initial tests showed no improvements of the results so the idea was not pursued any further.

[6] About one percentage point increase in accuracy on average.

3.4 Finalizing the classifier

Now we have a method of performing both sentiment and subjectivity classification but the goal of the project is to have a single classifier capable of assigning an arbitrary document to one of the three classes *positive*, *negative* and *objective*. There are, at least, two ways in which this can be achieved using the proposed method for the individual tasks. One of the alternatives is to regard the problem as a general three class problem and train two separate classifiers to distinguish between, for instance, samples in the positive class from samples in both the negative and objective class. This approach however produced extremely poor results, barely beating choosing at random. The results could be explained by that this approach completely disregards the relation between the classes. A better way is to simply make the classifier work in two steps, given a document, first classify it as subjective or objective and if the result is subjective feed the document to the second classifier which decides if it is positive or negative. This way the implicit relation between the classes is used.

The primarily intended application for the method developed in this project is to perform the classifications as a part of a social media analysis tool. The results would be presented both as statistics (how many positive and negative texts have been made about a certain keyword) as well as actually displaying the positive and negative posts to the end user. This makes the precision of the classifier more important than the recall (as long as the recall is the same for both classes). It is better to incorrectly classify posts as being objective, which would not be visible, than to put a incorrectly classified post in the list of posts displayed to the user. Luckily Support Vector Machines have a property which could be used to increase the precision at the cost of recall. For every new sample classified we can measure how far away from the decision line the sample is and simply apply a threshold to this value to make the classification. This is motivated by the fact that the SVM is more "certain" about the classification when the data point is far from the decision line. To make it a bit more complicated with the approach of combining the classifiers, discussed in the previous paragraph, we actually have two thresholds to set, one for each classifier. This makes it harder to find appropriate thresholds and was done by trial and error for the final classifier in this project.

3.5 Evaluation

When dealing with this type of problem where labeled data from the real domain are hard to acquire and only available in small quantities care must be taken to not overfit the method to the training data. This is the reason why we in this project start with the issue of cross domain review classification, where getting labeled data is not a problem, and try to transfer it to the real domain instead of just focusing on the real problem from the start. Keeping this in mind, all choices about which

method and parameters to use were decided based on results obtained using only the training data. Not until everything was set was the method tested on the real samples. This is true for everything discussed in this chapter except the last section, the final classifier combination and thresholding were evaluated on the real data to produce as good final results as possible. Furthermore all tests which used training and test data from the same domain were carried out using 10-fold cross validation, see section 2.1. By using this methodology the method and results presented in this project should be as general as possible and not only apply to the small amounts of hand labeled data used for verification.

3.6 Implementation

A complete description of the implementation used in this project will not be provided but since one of the goals is that the method should be computationally efficient a couple of things should be noted. All code produced for this project is written in perl, simply because the rest of the intended application was written in perl as well. Perl is not the fastest language available but all heavy computing tasks were carried out by external code. The Support Vector Machine implementation used was LIBSVM [3], which is very fast and feature rich and provides wrappers for many languages including perl. While there are implementations of Random Indexing, for instance The Semantic Vectors Package[7], none of the found ones did precisely what was needed for this project so one own version was implemented. Luckily the actual implementation of Random Indexing is fairly straightforward and all computing, both creating and using the model, is performed by vector and matrix operations. For these operation the Perl Data Language[8] (PDL) was used which is implemented efficiently in C. The only real performance issue encountered in this project comes from the size of the word space model. As mentioned the largest dataset used contains about two million unique words which leads to a size of $2000000 \times 1500 \times 8 = 24000000000$ bytes[9] or about 25Gb with some overhead added. This model did not fit in the main memory of the used computer so some kind of disk cache was needed. For this the Linux utility mmap[10] through perl wrappers was used which lets the model be treated as if in the main memory and all caching functionality is handled by the operating system. This made the implementation extremely simple while good performance was achieved. The rest of the functionality like file reading and word tokenization was implemented in pure perl.

[7] http://code.google.com/p/semanticvectors/

[8] http://pdl.perl.org/

[9] Using the proposed dimensionality of 1500 for the context vectors and storing each value as 8 byte double

[10] http://man7.org/linux/man-pages/man3/mmap64.3.html

Chapter 4

Results

In this chapter the results of the different tests executed during this project are presented. Note that only the results of the final methods using the best found parameters are displayed since there is not enough space to present all intermediate results. The results are divided into three sections, baseline results, random indexing and real data. Baseline results refers to the results of the usual way of performing text classification tasks on the five different domains and under random indexing the results of the same tests but with the method proposed here are presented. Finally in the section named real data comes the results of both the usual and our new method when tested on real data from actual social media.

4.1 Baseline results

As mentioned in the last chapter a number of tests were performed to establish a baseline for comparison and to verify that the assumed cross domain problems actually exist. For this second purpose a number of different classifiers using different parameters and techniques were trained and tested within and across domain. Since the datasets used are balanced the results are measured in accuracy (applies to all results presented here and in the next section) and are presented in tables 4.1 and 4.2.

A few things are worth noting about these results, first that the assumption of cross domain issues seems to hold since there is obviously a great decrease in accuracy when testing across domains. The positive vs negative classification results are also comparable to the results presented in previous research using the same techniques on English datasets. This indicates that the created datasets of reviews are sound and that the methods used for the baseline are implemented properly. On the other hand the results of the subjective vs objective task are extremely high and do

Train \Test	book	shop	movie	cell	game
book	**83.08**	73.92	76.73	72.10	77.44
shop	63.69	**87.55**	65.14	77.40	74.88
movie	79.47	73.79	**81.88**	74.80	80.69
cell	65.54	77.45	65.47	**84.15**	74.75
game	69.17	74.88	66.98	74.15	**81.19**

Table 4.1. Best accuracy of positive vs negative classification in and across domains.

Train \Test	book	shop	movie	cell	game
book	**95.49**	65.76	79.48	68.62	69.16
shop	77.53	**98.78**	84.61	98.25	95.69
movie	87.68	87.09	**97.84**	90.33	91.56
cell	68.59	87.39	79.10	**98.88**	86.81
game	77.81	90.98	87.32	95.85	**98.09**

Table 4.2. Best accuracy of sentiment vs objective classification in and across domains.

Target	Accuracy
book	74.39
shop	79.84
movie	73.97
cell	79.15
game	81.69

Target	Accuracy
book	84.25
shop	93.11
movie	91.10
cell	97.60
game	96.75

Table 4.3. Accuracy when leaving one target domain out for testing in positive vs negative classification (left) and subjective vs objective (right) using the normal bag of words plus SVM approach.

not leave room for much improvement. That these sets would be quite easy were expected since completely different types of texts were used for the different classes, but perhaps not this easy.

Next the tests used for the actual baseline to which all later methods would be compared were executed. Here the technique of training on a mixture of domains is employed and the results when leaving each domain out in turn for testing are presented in table 4.3. The results are mostly as expected, still quite high accuracies for the subjective vs objective task but for every test (except positive vs negative in the games domain) the result is lower than when performed in domain. The reason for the increase in accuracy of the game domain test could be due to the relatively small number of samples in that dataset[1].

[1]It is the smallest domain with only 800 positive and negative samples

Target	Accuracy
book	76.01
shop	84.05
movie	75.36
cell	80.55
game	78.50

Target	Accuracy
book	87.16
shop	98.41
movie	97.08
cell	99.02
game	98.94

Table 4.4. Accuracy when leaving one target domain out for testing in positive vs negative classification (left) and subjective vs objective (right) using Random Indexing plus SVM.

4.2 Random indexing

The results produced when using the method described in the last chapter using Random Indexing and a Support Vector Machine are presented in table 4.4. As we can see this new approach improves the accuracy of all the tests except one, even with the already high accuracies of the subjective vs objective task. The only test in which the accuracy decreased was again in the game domain and the explanation could of course still be that it is the smallest domain. The word space model used in these experiments is built using all available training data and since there are fewer samples from the game domain these will not influence the model as much as samples from the other domains. These results still show that adding information from both out of domain and target domain unlabeled data by the use of a word space model built with Random Indexing does indeed help with the issues of cross domain classifications. The issues still remain however as can be seen when comparing these results to the indomain results in tables 4.1 and 4.2.

4.3 Real data

The first of the two labeled datasets from real social media is almost balanced so we keep evaluating it with accuracy. The results of applying both the normal bag of words plus SVM method and the new one using Random Indexing is presented in table 4.5. Note that the results there are achieved when training on the review data and only testing on the real samples but using a word space model built from the large unlabeled dataset collected from real social media. Further the entire process of testing all combinations of parameters for the normal approach was redone for this example and only the best result is shown here. For the new method the same parameters as chosen during testing on the training datasets were used. The results show that the Random Indexing approach still beats the normal one but also that the real data are generally harder to classify. For the positive vs negative task the results are at least comparable with the results of the training data but for subjective vs objective both methods perform extremely poorly. It was expected

Method	Accuracy
BOW	71.66667
RI	76.33333

Method	Accuracy
BOW	60.66667
RI	62

Table 4.5. Accuracy on the first real dataset for positive vs negative classification (left) and subjective vs objective (right) using both the normal bag of words approach and the new using Random Indexing.

Method	Accuracy
BOW	63.66667
RI	78.3

Method	Accuracy
BOW	61.66667
RI	66

Table 4.6. Accuracy on the first real dataset for positive vs negative classification (left) and subjective vs objective (right) training both methods on the actual real dataset.

	Pos precision	Pos recall	Neg precision	Neg recall
Human 1	0.826	0.826	0.615	0.889
Human 2	0.656	0.913	0.800	0.889
Human 3	0.667	0.957	0.600	1.000
Human 4	0.857	0.522	0.750	0.333
Human 5	0.750	0.652	0.538	0.778
Human 6	0.867	0.565	0.467	0.778
RI	0.900	0.391	0.600	0.667

Table 4.7. The precision and recall of both the positive and negative class by the different test persons and the final classifier proposed in this project.

that the real data would be harder, especially for the subjectivity classification, but these results just barely beat choosing at random.

Since we have produced labeled data from the real domain to test with, another experiment where both methods were trained and tested on the real data was performed. The results of this test are presented in table 4.6 and basically show the method's ability to deal with smaller amounts of training data. The Random Indexing method does this quite well and actually beats the results of training with more but out of domain data while the normal approach performs worse on one task and about the same on the other. The results of subjectivity classification are even with this improvement still very poor.

To test how this poor performance on the subjectivity classification task would affect the real intended usage of the method the second real dataset was used. This set, containing data from real social media was created, as described in section 3.1, by letting different test persons classify the same samples and using majority vote to decide the "correct" label. Doing this lead to an unbalanced set so instead of just accuracy, the precision and recall of samples classified as positive and negative were

used to be able to compare the different results. For this the method of producing the final classifier described in section 3.4 was used on the classifiers created using Random Indexing and trained on the data in the first real dataset. The trade off between precision and recall was also performed to try to achieve as good results as possible. These results are presented in table 4.7 and show that this is a hard task not only for automatic classifiers but also for actual humans.

Chapter 5

Conclusions

The goals of this project were twofold, to see if Random Indexing could be used to help with cross domain classification issues and if the same issues and solutions transfer to the task of classifying general posts made in social media. For the first part the answer is definitely "yes", the results clearly show that the proposed method using Random Indexing outperforms the normal bag of words approach. In all but a single test case this new method achieved higher accuracy and in many of the cases even by a large margin (see tables 4.3 and 4.4). Furthermore the proposed method relies on more than just experimental results but is also motivated by previous research in the area as well as by discussions made in this report. For the second part the answer is still "yes", only not as definitely as in the first part. The new method did outperform the ordinary one but the result was still not as good as desired.

5.1 Real application

As mentioned, the results on actual posts made in social media was improved by the new method but for the task of classifying them as subjective or objective the result was not good enough. Even though sufficient performance was reached for the positive vs negative task the overall result for the final classifier definitely left room for improvements. This is due to the two step approach of combining the classifiers which means that with a poor first step classifier the second, better one, will both miss a lot of subjective texts and be faced with many objective ones. This is clearly not what it was trained for so the relatively poor overall results are to be expected. All this said, using the thresholds of the classification and trading recall for precision the results could be used in a real application as the one intended in this project. Because of the extreme amounts of data produced in social media it might be acceptable to only catch a small amount of the subjective texts if it means that few false positives occur.

5.2 Revisiting the problem

The reason for the poor performance on real data is not necessarily the fault of the proposed method. It might be that the simplification of regarding sentiment analysis as a document level classification problem is to simplify it too much. One single document can of course contain positive, negative and completely objective expressions all at once and even if for instance only positive expressions are present the document as a whole might still not be regarded as positive in the sense looked for. For instance, a common sight among posts in social media is blogs where the author describes what he or she has done during a day. These are typically about completely mundane things and do not really contain any real opinions but the general tone of the posts is still extremely positive. Such a post would of course be classified as positive because it only contains positive words but would also be completely uninteresting if trying to capture the public view on some subject accidentally contained in the post. This possibility is indicated further by the results of the different test persons on the smaller dataset of real social media posts. While real humans beat the automatic classifier the difference between it and the person with the worst results is not that big. Furthermore, if real humans do not agree on the class of a post how should an automatic method perform better? The response of a human must be seen as a "correct" answer since the only other way of really knowing what is correct is to ask the actual author of each post.

One approach to the problem that is similar to the one taken here is to regard each sentence as a separate sample instead of the entire document. This is actually the preferred approach of most previous research dealing with only subjectivity classification. By doing so some of the problems are dealt with and you get more resolution in the data. The reason why this was not used in this project was partly because then reviews could not have been used as training data but also that the initial tests indicated that the subjectivity classification task was extremely easy.

5.3 Future work

Despite the poor performance on the subjectivity classification task of real data the method of using Random Indexing does show some potential. The results of the sentiment classification task were actually quite good so continuing this work seems like a good idea. Obviously any future work based on the ideas presented in this project would have to deal with the subjectivity classification part. This could be as simple as trying exactly the same method on sentences instead but there are also entire papers aimed at only subjectivity classifications with more advanced and interesting methods. One example is presented in [16] where subjective sentences are extracted from a document by posing the problem as a graph problem solved

with minimum cuts. Using that as a preprocessing step would probably increase the performance significantly.

Besides the obvious issue there are a number of smaller parts of the method which probably could be improved to increase the performance as well. One example of this is the process of tokenizing a given document into its individual words which was naively implemented for this project. Furthermore some choices and parameters, like the properties of the context vectors or stop word removal, could probably be more thoroughly evaluated in order improve the results.

Lastly, if this method were to be employed in a real application running over a longer time, the questions of how much data the word space model should hold and how it should be updated needs to be answered. To be able to keep up with new fashion words, for instance, data from too long ago can not be left to influence the model too much. Because of the incremental nature of Random Indexing a lot of interesting options are available and given enough data over a stretch of time interesting properties could probably be found.

Bibliography

[1] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, volume 1, pages 2–1, 2005.

[2] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G.A. Reis, and J. Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*, 2008.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] Trevor Cohen, Roger Schvaneveldt, and Dominic Widdows. Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. *Journal of Biomedical Informatics*, 43(2):240–256, 2010.

[5] Isaac G Councill, Ryan McDonald, and Leonid Velikovich. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59. Association for Computational Linguistics, 2010.

[6] S.T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2005.

[7] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.

[8] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[9] J. Karlgren, A. Holst, and M. Sahlgren. Filaments of meaning in word space. *Advances in Information Retrieval*, pages 531–538, 2008.

[10] Jussi Karlgren and Magnus Sahlgren. From words to understanding. In *Foundations of Real-World Intelligence*, pages 294–308, 2001.

[11] B. Liu. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing,*, pages 627–666, 2010.

[12] S. Marsland. *Machine learning: an algorithmic perspective*. Chapman & Hall/CRC, 2009.

[13] Justin Martineau and Tim Finin. Delta TFIDF: An improved feature space for sentiment analysis. In *International Conference on Webogs and Social Media*. The AAAI Press, 2009.

[14] G.A. Miller et al. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[15] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.

[16] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[17] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[18] Magnus Sahlgren. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*, 2005.

[19] Magnus Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm, 2006.

[20] Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54, 2008.

[21] Magnus Sahlgren and Rickard Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *COLING '04 Proceedings of the 20th international conference on Computational Linguistics*, pages 487–493, 2004.

[22] Magnus Sahlgren, Anders Holst, and Pentti Kanerva. Permutations as a means to encode order in word space. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci'08)*, 2008.

[23] Magnus Sahlgren and Jussi Karlgren. Buzz monitoring in word space. In *Proceedings of the 1st European Conference on Intelligence and Security Informatics*, pages 73–84, 2008.

[24] Magnus Sahlgren and Jussi Karlgren. Terminology mining in social media. In *The 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, 2009.

[25] Magnus Sahlgren, Jussi Karlgren, and Gunnar Eriksson. SICS: Valence annotation based on seeds in word space. In *Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, 2007.

[26] P.D. Turney, P. Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

[27] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.