Text Clustering with Random Indexing

PER-ANDERS STAAV

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2009
Text Clustering with Random Indexing

PER ANDERS STAÅV

Master’s Thesis in Computer Science (30 ECTS credits) at the School of Electrical Engineering
Royal Institute of Technology year 2009
Supervisor at CSC was Magnus Rosell
Examiner was Stefan Arnborg

TRITA-CSC-E 2009:011
ISRN-KTH/CSC/E--09/011--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.csc.kth.se
Abstract

This project explored how the language technology method Random Indexing can be used for clustering of texts from Swedish newspapers. The resulting Random Indexing based representation yields similar results as an ordinary representation when the number of clusters matches the real categories. With an increased number of clusters the Random Index based representation yields better results than the regular representation.

Random Indexing is a scalable and computationally effective method that employs random projections to be able to compare all encountered contexts of the words. The downside of the method is that a certain level of random noise is added to the information content of each word. Unfortunately the small random disturbances become a real concern when combining a number of semantically related words.

A number of methods were examined to make a Random Index based representation perform better. Weighting the context of a word with the Inverse Document Frequency and normalizing the resulting vector were found to be the most effective ways to leverage the information content for clustering purposes. The project also included experiments on removal of repeated words, filtering of words based on word frequency and the use of dampening in Random Indexing's weighting schemes.

Finally the project examined how to detect programming errors that prevent the index to function properly. Calculating the variance of how the word representations are distributed in the Random Index was shown to be a possible way to find at least one type of severe error.

Textklustering med Random Indexing

Sammanfattning


Ett antal metoder undersökt för att få den Random Index baserade representationen att presteras bättre. Viktning med Inversa Dokument Frekvensen av de ord som finns i närheten av ett ord samt normalisering av den resulterande vektorn visade sig vara de mest effektiva sätten att bevara informationen som behövs för textklustering. Projektet undersökte även filtrering av upprepade ord, filtrering baserad på ordfrekvens samt användning av dämpning i Random Indexings viktningsschema.

Avslutningsvis undersökt projektet hur programmeringsfel som hindrar Random Indexing representationen att fungera kan upptäckas. Att beräkna variansen över hur ordens representationer finns distribuerade visade sig vara ett möjligt sätt att upptäcka åtminstone ett sorts allvarligt fel.
Acknowledgements

This master's thesis would not have been possible without aid from a number of persons. Most important has been my supervisor Magnus Rosell at KTH who came up with the original idea for the project. He has given plenty of good advice about how to organize the work and what experiments that are interesting. I chose to look for a master's project at NADA instead of some company because I was interested in doing a project close to academic research and I am very happy about the choice. I had no idea this project would be so inspiring when Magnus first presented the idea for me.

Secondly I would like to thank Dr. Martin Hassel that has shown me an incredible patience in dealing with my questions and bug reports for the JavaSDM program package. The response time has often been incredible, to have a patch ready just a few hours after a bug report has been sent is comparable to a professional service. One of the reasons this has been possible is probably Martin's very good and clear coding style when programming, but I am still very glad that he has devoted time to help even though busy with the final stages of his doctoral thesis.

The rest of the people at the Human Language Technology Group have also been very valuable with feedback and encouragement. I will seriously miss the creative 'couch meetings' that helped me stay focused on the project.

I would also like to thank Dr. Magnus Sahlgren at Swedish Institute of Computer Science (SICS) that directed me to the text of Samuel Kaski when I had trouble understanding how the cosine measure perform for an Random Index.

A number of friends have given me feedback about the text. I value your assistance very much. Worthy of special mentioning is Eva Granbeck that has spent lots of time giving me constructive suggestions about how to improve the language of the report.

Finally I want to thank my beloved Loka who has given me much encouragement and support when ever progress with the project was limited.
## Contents

1 Introduction.................................................................................................................................. 1
  1.1 Computer Aid in Classification of Texts.................................................................................. 1
    1.1.1 What Decides if Two Texts are Similar?............................................................................. 1
  1.2 Models and Computational Cost............................................................................................... 1
    1.2.1 Word-spaces (Bag-of-words)............................................................................................. 2
    1.2.2 Projections to Reduce Dimensionality............................................................................... 3
  1.3 Introduction to Random Indexing............................................................................................. 4
    1.3.1 What Random Indexing Aims to Solve............................................................................... 4
    1.3.2 Basic Description................................................................................................................. 4
    1.3.3 Parameters.......................................................................................................................... 5
    1.3.4 Random Indexing Compared to LSA.................................................................................... 5
    1.3.5 Benefits of Random Indexing .............................................................................................. 6
  1.4 Language preprocessing.......................................................................................................... 6
    1.4.1 Stemming and lemmatizing.................................................................................................. 6
    1.4.2 Distinguishing Words and TF-IDF/LogIDF Weighting....................................................... 6
  1.5 Clustering.................................................................................................................................. 7
    1.5.1 Clustering Algorithms......................................................................................................... 7
    1.5.1.1 Clustering Algorithm: K-Means..................................................................................... 7
    1.5.1.2 Clustering Algorithm: Agglomerative............................................................................ 8
    1.5.2 Clustering Evaluation.......................................................................................................... 8

2 Methodology.................................................................................................................................. 10
  2.1 Goal of the Project.................................................................................................................... 10
  2.2 Existing Programs and Servers.................................................................................................. 10
  2.3 Evaluation Set........................................................................................................................... 10
    2.3.1 KTH News Corpus.............................................................................................................. 10
  2.4 Programming............................................................................................................................ 11
    2.4.1 Initial Obstacles................................................................................................................ 11
    2.4.2 The Real Obstacle.............................................................................................................. 12
  2.5 Work Flow.................................................................................................................................. 12
    2.5.1 Sampling Resulting Quality and “Hill Climbing”............................................................... 12
    2.5.2 Inconsistent Results and the Solution................................................................................. 12
    2.5.3 Roadmap of the Report..................................................................................................... 13

3 Random Indexing......................................................................................................................... 14
  3.1 Theory...................................................................................................................................... 14
    3.1.1 Representation.................................................................................................................. 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.2 Different Kinds of Similarities</td>
<td>14</td>
</tr>
<tr>
<td>3.1.3 Function Words</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Context Window Size</td>
<td>15</td>
</tr>
<tr>
<td>3.2.1 Different Types of Weighting Schemes</td>
<td>15</td>
</tr>
<tr>
<td>3.2.2 Earlier Use of Weighting Schemes</td>
<td>16</td>
</tr>
<tr>
<td>3.3 The Cosine Similarity</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Dimensionality</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1 Dimensionality Linked to Document Representation Errors</td>
<td>17</td>
</tr>
<tr>
<td>3.4.2 Negative Similarity Caused by Dimensionality Size</td>
<td>17</td>
</tr>
<tr>
<td>3.5 Preprocessing Before Random Indexing</td>
<td>18</td>
</tr>
<tr>
<td>4 Document Representations</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Normalization</td>
<td>19</td>
</tr>
<tr>
<td>4.2 Preprocessing of Numbers</td>
<td>20</td>
</tr>
<tr>
<td>4.3 Filtering of Low Frequency Words</td>
<td>20</td>
</tr>
<tr>
<td>4.4 Filtering of High Frequency Words</td>
<td>20</td>
</tr>
<tr>
<td>4.5 Scaling with LogIDF</td>
<td>21</td>
</tr>
<tr>
<td>4.6 The Effect of Repeated Words</td>
<td>21</td>
</tr>
<tr>
<td>4.7 Similarity Signatures or Random Index Data</td>
<td>21</td>
</tr>
<tr>
<td>4.7.1 Random Index Data</td>
<td>21</td>
</tr>
<tr>
<td>4.7.2 Similarity Signatures</td>
<td>22</td>
</tr>
<tr>
<td>5 Experiments, Results and Discussion</td>
<td>23</td>
</tr>
<tr>
<td>5.1 Evaluation Set</td>
<td>23</td>
</tr>
<tr>
<td>5.1.1 The Effect of Document Length</td>
<td>23</td>
</tr>
<tr>
<td>5.1.2 Slow Performance</td>
<td>23</td>
</tr>
<tr>
<td>5.1.3 Categories and Text Sets</td>
<td>24</td>
</tr>
<tr>
<td>5.2 Fundamental Parameters</td>
<td>24</td>
</tr>
<tr>
<td>5.2.1 Context Window Size</td>
<td>24</td>
</tr>
<tr>
<td>5.2.2 Examined Weighting Schemes</td>
<td>24</td>
</tr>
<tr>
<td>5.3 Main Experiment</td>
<td>25</td>
</tr>
<tr>
<td>5.3.1 A Random Index Representation vs a Standard Representation</td>
<td>25</td>
</tr>
<tr>
<td>5.4 Complementary Experiments</td>
<td>27</td>
</tr>
<tr>
<td>5.4.1 How to Combine Words</td>
<td>27</td>
</tr>
<tr>
<td>5.4.2 Does Dampening Improve Clustering Quality?</td>
<td>31</td>
</tr>
<tr>
<td>5.4.3 Can Incremental LogIDF Replace LogIDF?</td>
<td>35</td>
</tr>
<tr>
<td>5.4.4 Is 1000 in Dimension Acceptable?</td>
<td>36</td>
</tr>
<tr>
<td>5.4.5 Agglomerative Clustering</td>
<td>37</td>
</tr>
<tr>
<td>5.5 Analysis of JavaSDMs Use of Dimensionality</td>
<td>40</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter presents the problem domain and the theoretical cornerstones that are the background for this project. The major terminology is introduced, but is covered in more detail in the following chapters.

Automatic processing of human written text is something the casual user of a computer has come to expect. Search engines for finding pages on the internet started as tools to find pages that might be about the topic the searcher is interested in, but have evolved and the famous search engine Google is today probably the main source when people want information about a subject. What is there really left to be done when there is such brilliant tools for extracting information?

The answer is, of course, that not every task concerns extracting known information from a large set of texts. In many real world tasks the lack of information about the data is the very reason that the human user want computer assistance. An example would be survey answers from a customers feedback program. A large number of answers is desirable to make sure the answers are representative, but it is very expensive to manually try to find out if there is a general pattern hidden in free text answers. This is especially true if the answers are many. A computer that could aid in this task could potentially give great benefits.

1.1 Computer Aid in Classification of Texts

A very legitimate question at this point is what makes the problem of classification of unknown texts so difficult that it has not been adequately solved already? One aspect of importance is available computer resources. The processing power demands on the servers that power a major internet search engine is massive, but fortunately the number of potential users are even greater. Computer aided examination of survey answers on the other hand will usually have a single user, and it is not until recent years that standard computers have become powerful enough to handle the amount of data needed in any realistic situation. Still this problem is arguably secondary to the problem of determining to what degree two texts are similar.

1.1.1 What Decides if Two Texts are Similar?

The question if two texts are similar may at first seem simple. Why just not compare the texts? This leaves the question open about what in the texts that the computer program should be comparing. The meaning of the texts should be compared, but can a computer really be programmed to recognize what a text really means?

When a human reads a text there are different kinds of understanding. The face value of the words might not match the interpretation when the style of the text is considered or when the human reader compares to what other sources write about the same topic. Fortunately there is a clear difference between understanding the meaning of a text and to be able to recognize that two texts concern the same main topic.

In any case it is quite clear the human designer need to determine what aspects of two texts that indicate the topic is the same. If the texts contain the same words it strongly supports the conclusion that they have the same topic. However, with many different authors of texts with different purposes and styles, the comparison of raw word lists have severe limitations.

1.2 Models and Computational Cost

Effective methods to deal with the language data are imperative when processing natural languages. The need for correctness of the model used to describe the data must be balanced
Introduction

with the need to limit ourselves to models that have a computational cost that is acceptable. Languages are developed and changed by actual users and the number of unique words is ever increasing. The Information Retrieval domain deals with the problem how to match a query for information with a large set of texts to determine what texts answer the query. A number of results from that domain are typically employed when processing natural languages and the subject will therefore be given a presentation.

1.2.1 Word-spaces (Bag-of-words)

The overview of the model that is presented here is based on the description in Jurafsky (2000). The idea used by major search engines to allow Information Retrieval is surprisingly simple. First every unique word that exists in the document collection is given a unique number to identify the word. Secondly every document in the document collection is assigned a vector with a length equal to the number of unique words found in all documents. Since the number of unique words equals the length of the vector the number can be used as a index in the vector.

For a particular text start with zero on all positions in the vector. Iterate over the text and for every word found determine what number is assigned to it and increase the number on that position in the document's vector by one. The document vector will afterwards have positive entries proportional to the frequency of the words if the word exists and zero otherwise. Two texts can be considered similar if their vectors have positive entries on the same position. The degree of similarity is proportional to how many words are used in both texts.

The collection of vectors of all documents constitute what is called a word-space. This is a word-by-document matrix that answers the question what words are found in each document. The word-space ignores word order, and is therefore often called a bag-of-words. The dimensionality of the vectors in the word-space is the same as the number of unique words found in all documents. In literature this approach to Information Retrieval is often referred to as the Vector Space Model.

1.2.1.1 The Properties of the Word-space

A problem with handling bag-of-words vectors is that language itself causes the data to be naturally sparse. Most words only exist in a very limited percentage of the texts, leading to the word vectors having a very high dimensionality and being filled mostly with zeros. As an example on how sparse the data is, Sahlgren reports that in the experiments he perform a matrix with co-occurrences between all words (a word-by-word matrix) has more than 99% of the entries as zero (Sahlgren 2005).

The high dimensionality of the vectors raises large computability problems in terms of storage needed and consumes much time when the word-space is processed, a more compact representation is preferred.

The idea that two texts need to share words to be similar in meaning is also more or less incorrect. In real language data there are synonyms, and the vector indexes for those words are obviously not independent. In some cases the synonym pair never appear together in any document, even though they mean the same thing. For instance the words *truck* and *lorry* refer to the same thing in the real world but will usually not both be found in one document.

1.2.1.2 Dimensionality Reduction of the Word-space

One popular way to reduce the dimensionality of the word-space is using a method called Latent Semantic Analysis/Indexing (LSA/LSI) (Landuer, Dumais 1998). The idea is to analyse the recorded word frequencies with a matrix factorization technique called Singular Value Decomposition (SVD) and truncate those parts that contribute little to the final value.

One interesting feature of this approach is that synonyms will be bundled together since they do occur in similar contexts and the information loss is small if they are treated as the same. The “latent semantic” part of the name refers to the method makes such hidden connections between
words visible. In some sense the dimension reduction moves the representation from containing words towards containing concepts.

Hassel (2007) points out that LSA’s effect to compact words into concepts is dependant on the target dimensionality. The SVD finds the optimal way to reduce the word space to the target dimensionality, but there is no real theory of what the dimensionality should be. A very low target dimensionality will mean that the words are merged to comparably few concepts while a high target dimensionality will mean that related words are kept separate.

Ignoring the question about choice of dimensionality the major downside of LSA is that the method is very computationally expensive. Before the SVD can be applied every document must be loaded and word-space vectors be constructed since the method builds on analysis of all information. If new language data is encountered later the whole LSA process need to be redone from the beginning since the new material might change what parts of the data that the SVD should truncate. The new data can be folded into the index without redoing the LSA process, but then it impossible to know if the word-space faithfully represents the new data or not (Sahlgren 2005).

There are other ways to produce a representation of word use. One of the better known is Hyperspace Analogue to Language (HAL). Sahlgren, in his Ph.D thesis, ”The Word-Space Model” reviews these alternatives and what kind of information they will gather with different kinds of contexts (Sahlgren 2005).

1.2.2 Projections to Reduce Dimensionality

In the real world everyone are familiar with the idea that a picture gives us something that can represent the real object. The picture is 2-dimensional while the real object is 3-dimensional. Mathematically what has happened during the creation of the picture is that a vector projection has been performed. The projection gives a mapping between the points in the 3-dimensional space and where they should end up in a 2-dimensional space to give minimal difference in appearance between object and picture.

If the object is represented as a set of vectors, these vectors form a matrix. The vector projection is executed by a matrix multiplication with a matrix with suitable dimensionality and content. If the original data is located in a \( s \times d \) matrix, a multiplication with a \( d \times k \) matrix will produce a new matrix that is \( s \times k \) in dimension. This occurs because the rows of the first matrix will be multiplied with the columns of the second matrix. Figure 1 demonstrate the effects of a correctly executed multiplication. Intuition suggests that any projection that reduces the dimensionality greatly must lose most of the information in the data, fortunately this intuition is not correct when it comes to sparse matrices.

\[
\begin{pmatrix}
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
-2 & 3 \\
2 & -1 \\
1 & -1 \\
-1 & -1
\end{pmatrix}
= 
\begin{pmatrix}
-1 & 2 \\
0 & 3 \\
3 & -2 \\
-3 & 5
\end{pmatrix}
\]

*Figure 1 Example of a matrix multiplied with a random matrix that reduces the number of columns from 4 to 2*

1.2.2.1 Random Projections

Any projection will not keep the information even if it is a sparse matrix, but if the projection matrix is filled with random numbers with suitable properties the risk that sparse data is damaged during the projection is very small. A more formal explanation of the phenomena is that if we form two vectors in high dimension by randomly choosing components the vectors become approximately orthogonal. Since the random projection is only approximately
Introduction

orthogonal the distance between points will become distorted when the projection is applied to the data.

For the random projection to be successful for Information Retrieval purposes the distortion caused by the random projection must not add any bias to the compacted word space. An example on what conditions is needed to achieve this is that Kaski (2005) has demonstrated what happens when the components of the projection vectors are random numbers between -1 and +1, the mean of the projection vectors is zero and the inner product of the vectors is used for comparison. The distortion to the inner product due to random interference become zero on average and have a variance that is less than 2/d, where d is the dimensionality of the final word space. The next sections explain how these ideas are employed in the method Random Indexing.

1.3 Introduction to Random Indexing

Bag-of-concepts is a phrase used to describe another solution to the word-space dimension reduction problem. The method is called Random Indexing and employs a random projection to compress the data. It was developed by Pentti Kanerva (2000) and Magnus Sahlgren (2001).

The reason the phrase bag-of-concepts is used to describe the method is due to similarity to how an LSA improved word-space performs in the synonym-finding part of the TOEFL test (Test Of English as a Foreign Language).

LSA merges words into concepts to reduce the dimension of the vector space and Random Indexing borrows this terminology even though the connection between the method and concepts of the words is less obvious for Random Indexing. Interestingly enough the Random Indexing method and LSA both perform comparable to foreign applicants who really take the synonym-finding test of the TOEFL test (Kanerva 2000).

1.3.1 What Random Indexing Aims to Solve

A human can often deduce the meaning of a word by looking at the words that occur close to it. For instance the words "stable", "riding","fodder" and "saddle" gives a pretty good description of what a horse really is. The bag-of-words approach is built on the idea that a text as a whole is represented by the words in the text. Yet it is easy to imagine one or a couple of the words missing from a certain text, not all texts about horses need to contain the word "saddle" even though the human observer recognizes the connection with saddle when encountering the word horse. From a linguistic perspective “saddle” is a word found in the linguistic context of the word horse. The linguistic context is those words that help the reader to determine the meaning of a word. It seems plausible that words that occur close to a word often are part of the word's linguistic context.

In theory the programmer could with regular bag-of-words vectors build "bags" that each contain all words encountered close to a specific word. These vectors of occurrence could be used to compare if two words are similar in use or not. In practice the computational cost to process and store the many contexts is extremely large. A sampling of the document collection used in this project reveals that just a few thousands of short texts gives about 30 000 unique words, and the full document collection contains about 300 000 unique words. The concern that the vectors storing the information of the bags might not fit in the computer memory is very real.

1.3.2 Basic Description

With Random Indexing each word is represented by two vectors. One of the vectors contains a randomly assigned label. The random label is a vector filled mostly with zeros, except a handful of +1 and -1 that are located on random indexes. This kind of distributed representation can be generated on the fly the first time a new word is encountered.
The second vector is used to record the encountered contexts of the word. What should be included in the context of a word is determined by forming a context window that contains the words directly before and after the word in the text. If a word is found in the context window of a certain word the random label of the found word is added to context vector of the word to mark the connection with the word. Two words will have similar context vectors if the words appear in similar contexts in the text.

Random interference between the labels will mean that all words will have some similarity, but the impact from this is small provided that the dimension used is sufficiently large compared to the corpora loaded into the index.

1.3.3 Parameters

One parameter of the Random Indexing method is the length of the vector containing the random label. The number of entries in this vector will decide the dimension of the final index and thus the storage requirements. The number of nonzero entries in the random labels is another parameter that has an impact on how the random interference will be distributed over the random labels.

How to weight the words in the context window to approximate the real linguistic context constitute a very important parameter that has a great impact on the performance of the final Random Index. A major part of employing Random Indexing for a language technology task is to determine what weighting scheme and what context window size should be used by Random Indexing software during construction of the index.

1.3.4 Random Indexing Compared to LSA

LSA can theoretically operate on arbitrary language units (words, phrases, clauses or sentences) but is limited in practice by available memory and the fact that the method requires much processing power. A typical use of LSA is to reduce a word-space that is of the word-by-document kind to a concept-by-document matrix. Random Indexing is different in that it might be configured to handle a more wide range of tasks with the same resources. One possible mode of operation is to consider the whole document to be the context. In this case the method approximates a regular bag-of-words word-space. The dimension reduction is comparable to the one that LSA produces, but the method will not combine synonyms into concepts like LSA does, but just reduce the needed storage space.

With a more narrow context Random Indexing can assemble information about how the words are used in all documents. In this case the Random Indexing method produce an index that is of the word-by-word kind. This kind of word-space answers the question to what degree two words are similar.

Sahlgren (2005) give a good introduction to the terms paradigmatic and syntagmatic that describe the nature of information found in such word-spaces. Words are syntagmatic related if they occur together in same document. Paradigmatic related means the words occur in similar contexts. A typical relation that is paradigmatic in nature is a pair of synonyms. Sahlgren's analysis identifies LSA as approximating paradigmatic relationships since it transforms a word index with syntagmatic word-to-document relations to concepts that are paradigmatic in nature. Random Indexing with a narrow context window extracts the paradigmatic relationships directly.

This report will focus on using narrow context windows since Oscar Täckström in his master's thesis (Täckström 2005) at KTH found that this approach showed promise for better results for categorization of text categories where ordinary bag-of-words fail.

Intuitively it sounds plausible that key difference that might speak to Random Indexing's benefit is that LSA either combines two words or leave them completely untouched. If there is a set of five words that are related to different degrees, LSA may join some of these words (depending
on the target dimensionality) but can not reveal how similar these words are or if other words are almost as similar. With Random Indexing the data representation allows evaluation of to what degree any word pair is similar.

1.3.5 Benefits of Random Indexing

The key benefit of the random mapping idea in general is that the dimensionality of the final vector that describes the document will not depend on the number of documents or words that have been indexed. With Random Indexing we have the additional benefits in that the method is incremental, and that there is no need to sample all texts before results can be produced.

In essence Random Indexing is a method to compact sparse raw data to a smaller representation. Like the random mapping idea proposed by Kaski (1998) it builds on matrix multiplications with matrices with random numbers. The difference is that Random Indexing makes use of the fact that matrix multiplication is distributive. This means that instead of assembling a matrix containing all contexts and compressing it with a random projection we can incrementally perform the multiplication with the random projection and end with the same final result. A more detailed description of Random Indexing can be found in Chapter 3 of this report.

1.4 Language preprocessing

A number of different methods exist for making the language data easier to process for the computer in natural language tasks. The following are those of major importance to this project.

1.4.1 Stemming and lemmatizing

Stemming and lemmatizing (Jurafsky 2000) are two different approaches to deal with the same problem. In an information retrieval situation inflected forms of the word will make the words look different even though they describe the same thing. Examples would be that it does not make sense to differ between the singular form *car*, and the plural form *cars*, or the words *run*, *running* and *ran* when evaluating the general meaning of the text.

When doing lemmatizing each word is changed to its correct linguistic root, that is the lemma form. Stemming means a more simple approach where any suffix caused by the inflection form is simply cut. Stemming does not require the found stem to be a proper word, while lemmatizing requires the lemma form to be a real word. For instance stemming would reduce *cycle* and *cycling* to the stem *cycl*, but would fail to deal with the above example of *run* and *ran*.

On the positive side stemming means that words like calculate and calculation will receive a common stem, something that might be beneficial to information retrieval since these words are strongly related in terms of information. If stemming or lemmatizing is preferable it is very much a question of what natural language that will be processed and what the intended application is. Rosell (2005) reports a 4% increased quality in clustering of Swedish texts by the use of stemming.

1.4.2 Distinguishing Words and TF-IDF/LogIDF Weighting

Not every word in a text has equal importance for delivering information. Function words for instance like *and*, *or* and *the* only give information combined with other words within the actual phrase. Even ignoring such words there are differences in distinguishing power between different words, the word *chemistry* is more narrow in meaning than *science* and thus works better to distinguish texts from each other even though chemistry is a kind of science.

A number of different kinds of weighting methods have been developed to address this concern. Such weighting can be local within the document or global over all documents. One global method that is of particular interest to this project is the Inverse Document Frequency (IDF), that divides the number of documents with the number of documents that contain the word.
many cases the IDF itself gives a too large emphasis to words that appear in very few texts and an alternative is LogIDF that uses the logarithm of the IDF to lessen the impact of very rare words.

A local weighting method is Term Frequency (TF) that uses the frequency of the term in each individual document. This method can be motivated by the observation that a text that is really about chemistry will probably use the word *chemistry* more times than one that touches it while it discusses something different. It is possible to capture this by weighting the words with their frequency in the document. A potential problem is that the term frequency will add emphasis to longer documents since these are longer and the term frequency is therefore usually divided by the length of the document to improve the quality of the results in Information Retrieval situations.

TF and IDF are normally combined. Some authors use the term IDF to also refer to when the logarithm is used, to avoid confusion this report will use the notation LogIDF.

### 1.5 Clustering

Clustering in general concerns algorithms that are used to divide data into groups so that the members of a group are similar to each other, but not similar to the rest of the data. The data processed can be anything, for instance pictures, but in this project the focus will be on the application of clustering algorithms on text.

Clustering algorithms differ from classification algorithms in that clustering is an unsupervised process. A classification task is supervised in the sense that the user must define in advance what categories the algorithm should use. In many situations discovery of the structure of the data is the actual goal and any attempt to anticipate the categories in advance will hide the true nature of the data.

#### Examples of Text Clustering

Text clustering has not reached widespread commercial use so far, even though there are some quite promising results. A few examples of current and possible use.

- Google does employ text clustering in their Google news service. This service assembles links from major news sites and group the results together based on the content (http://www.news.google.com).
- Attempts to in advance cluster the data that internet search engines process to lower the response delay or raise quality of the results have not been successful. On the other hand clustering the output from the search engine has been shown to improve the quality of the answers (Hearst and Pedersen 1996).
- Clustering of free text survey answers from customer input programs or medical monitoring programs has been proposed as a good application of text clustering. (Rosell 2005)

#### 1.5.1 Clustering Algorithms

A number of clustering algorithms exist, but most of them are not efficient enough to be applied on large data sets. The following clustering algorithms have been used in this project. The description of the methods is based on a survey of available clustering methods (Jain et al 1999).

##### 1.5.1.1 Clustering Algorithm: K-Means

The most popular clustering algorithm for much data is K-Means. The algorithm begins by randomly selecting k texts as starting clusters and assigning each text to the closest cluster. In each consecutive iteration it calculates the average centre of each cluster and reassigns those
texts that are more similar to another cluster centre. The algorithm does a local hill climbing from the random selected documents at the start and will thus not converge to the same clustering on each run.

The complexity is $O(n \times k \times i)$, where $n$ is the number of documents, $k$ is the number of clusters and $i$ is the number of iterations. One of the main benefits of the algorithm is that there often are acceptable results after a limited number of iterations. The comparably low complexity, $O(k + n)$ in memory consumption, and easy implementation explains why K-Means is popular. When the number of elements to cluster is large K-Means is usually the only practical choice.

1.5.1.2 Clustering Algorithm: Agglomerative

Agglomerative clustering starts by making every text into a cluster on its own. Then it merges those clusters that are most similar. This process is repeated until everything has been connected into a tree, and then the desired number of clusters are extracted by splitting the top connections. The results of the clustering has been evaluated at certain numbers of clusters since there is no simple way to present the resulting tree in a written report.

The complexity of agglomerative clustering is $O(n^2 \times \log(n))$ and the memory requirement is $O(n^2)$. A number of alternatives exist for determining which clusters are candidates to be merged. When using so called Single Link the most similar pair of texts from the two clusters determine how similar the clusters are. When using Average Link the average of the clusters determines if two clusters are similar. Finally Complete Link means that the least similar pair of texts from the two clusters determine how similar the clusters are.

1.5.2 Clustering Evaluation

The results of the clustering can be evaluated by internal or external measures. Internal measures investigate how well the clustering algorithm used the available information during the clustering process. External measures compares the clustering to a previous classification, known to be reliable.

There are many reasonable classifications of a corpus of texts. Which one serves the purposes best depends very much on the intended use of the classification. External measurements test how well a clustering compares to only one such classification. A match between clustering and classification is a strong hint that the clustering might be useful in the same situation as the classification.

1.5.2.1 Entropy

Entropy is an external measurement that can be used to compare different clusterings. If there is a probability $p(j)$ that a random text taken from a cluster belongs to category $j$, the entropy of the cluster can be calculated with the following formula:

$$\text{entropy for a cluster} = - \sum_{\text{categories}} p(j) \cdot \log_2(p(j)).$$

The worst possible entropy for a cluster occurs when each category are equally represented in the cluster. The total entropy of the clustering is the entropy for all clusters combined and weighted by the percentage texts located in each cluster. For a theoretical perfect clustering the entropy is zero.

1.5.2.2 Normalized Mutual Information

Entropy tells how much error the clustering caused, from a practical viewpoint the end user is more interested in how much information was gained. Normalized Mutual Information (NMI) is an external measurement that indicates how much information that is shared between the clustering result and the reference classification.
Let \( m_{ij} \) = number of texts that belong both in cluster \( i \) and category \( j \), let \( n_i \) = number of texts in cluster \( i \), let \( o_{j} \) = texts in category \( j \), let \( k \) be the number of categories and let \( g \) be the number of clusters

\[
\text{NMI} = \frac{\sum \sum m_{ij} \log \left( \frac{m_{ij}}{n_i o_{j}} \right)}{(2 \log (gk))}.
\]

For a theoretical perfect clustering the Normalized Mutual Information will be 1, and for a total failed one with random assignment to the clusters, it will be zero.
2 Methodology

This chapter presents the goal of the project, existing programs that has been employed and how the work has been organized.

The previous chapter introduced the need for computer aided classification of unknown texts and how computationally efficient models of language content can be created. The clustering algorithms presented can potentially solve the problem of classification aid with unknown categories even though results so far has not been perfect. Looking at good results of bag-of-concepts representations like Random Indexing in other natural language processing tasks, the question arises if such representation would benefit text clustering too.

2.1 Goal of the Project

The goal of this project is to investigate how a document representation based on Random Indexing will perform in a standard text clustering task.

2.2 Existing Programs and Servers

Random Indexing has not been used for clustering purposes earlier. Fortunately there exists a GPL implementation of Random Indexing named JavaSDM (SDM – Spare Distributed Memory) developed by Ph.D Martin Hassel from KTH which produces well documented xml files (Hassel 2006).

The source code and documentation can be downloaded from [http://www.nada.kth.se/~xmartin/java/JavaSDM/](http://www.nada.kth.se/~xmartin/java/JavaSDM/). The package also includes services such as connection with language resources at KTH, automatic codepage detection (a codepage define the mapping between binary code and country specific letters) and removal of html elements.

It is not obvious which clustering algorithm makes the best use of a Random Index representation. My supervisor Magnus Rosell has provided slightly experimental Java classes that can perform clustering and evaluation using K-Means and Agglomerative clustering algorithms. The Agglomerative algorithms are considerably slower than K-Means. They are included in this project since they are a hierarchical approach instead of partitional like K-Means and there is hope they perhaps could utilize the data in a different way.

The school of Computer Science and Communications on KTH has developed a service name Granska that among much else provides lemmatizing of Swedish words. A presentation of this research project can be found at [http://www.csc.kth.se/tcs/projects/granska/index-en.html](http://www.csc.kth.se/tcs/projects/granska/index-en.html).

2.3 Evaluation Set

The question of what set of texts that should be used for evaluation of the method is always very important for clustering. Random Indexing needs only raw texts to assemble the context vectors of the words, but with a little caveat that the language needs to have some structure on the word order. On the other hand there is need to have some kind of categorization to compare against so that Normalized Mutual Information can be calculated.

2.3.1 KTH News Corpus

The KTH News Corpus was selected as the source of documents. This is a set of Swedish newspapers articles collected from the web (Hassel 2001). One of the benefits of this choice is that for two of the papers, “Aftonbladet” and “Dagens Nyheter”, the articles has the category
encoded in the file name. This categorization is based on what section of the paper the article was originally published in. Having such classification available directly allows for easy evaluation. The downside of this choice is that the topic of the article and what section of the newspaper it was published in might not match.

2.4 Programming

The JavaSDM package produces a Random Index that provides comparison of similarity between words and access to the raw data of the index. Unfortunately the speed of assembling the index is limited by the very slow lemmatizing server and this makes it impractical to build the representation from scratch every time.

It is reasonable to expect that numerous clusterings with different parameters might be needed before it is possible to deduce if a certain index is useful. With this in mind the choice was made to create a standalone program called TokenfileCreator to produce suiting data files for the clustering program. A standalone program to load the Random Index and produce the needed files also made it much more simple to adjust to improved versions of the two program packages.

2.4.1 Initial Obstacles

As the construction of the program TokenfileCreator progressed, several obstacles were discovered. A number of improved versions of JavaSDM and the clustering software was tested before everything was resolved.

2.4.1.1 Memory constraints

The JavaSDM package generally produces files that are rather large. An obstacle of technical nature was that allocating more memory to Java than 600 MB caused system failure with core dump of the kernel on the regular terminals at KTH. Moving the experiments to the CSC server Swift that has 4 processors and 16 GB of memory eventually solved this particular problem. Moving to Swift also had the benefit that large data files could be kept in memory instead of filling the user quota.

How much memory the JavaSDM needs depends on the parameter selection for the Random Indexing. The evaluation of the data representation performed at this stage of the project suggested the need for a dimension that consumes considerable memory. Later analysis indicates that the real requirement on the parameter selection is much more modest, and the method is stable even with less resources. The experiment chapter includes a more detailed analysis of the performance of the JavaSDM version used in these early experiments, see Section 5.5.

2.4.1.2 Failed Language Preprocessing

JavaSDM contacts language resources at KTH like the server Granska to perform language preprocessing. These are of excellent capability, but not always very stable due to them being run on servers that are dedicated to other things. From time to time the server will stop responding for a number of seconds when it is busy with more important work. The server also sometimes silently drops working on previous requests for preprocessing.

With TokenfileCreator this caused problems since the preprocessed language data is needed to build document representations. In some cases TokenfileCreator would encounter words that did not exist in the index because the server had failed when JavaSDM tried to lemmatize the word. In other cases the number of files added to the data file would differ between the experiments using the same file list since the server would be down when a particular document was handled by TokenfileCreator. The final solution to this problem was a new version of JavaSDM that can save the preprocessed data to an arbitrary folder for later reuse.
2.4.1.3 Swedish Letters and File names

One problem that caused confusion was missing categories from the final clustering. Examination identified the problem was caused by JavaSDMs codepage detection mechanism by mistake being applied to the filenames of the files processed. This caused Java to be unable to find any file name with non English letters in the file name. The “Aftonbladet” corpus have two such categories, *nöje* (entertainment) and *världen* (world).

2.4.2 The Real Obstacle

All problems mentioned in the Initial Obstacles list caused problems, but it is worth pointing out that, looking at the whole project the difficult problem is not the programming task. Loading a Random Index and to produce a xml file is a pretty straightforward task. The real difficulty has been to understand how document representations should be constructed to include useful information for a clustering task.

2.5 Work Flow

The bulk of the work in this project has been spent on one thing: trying to understand why the final clustering fails. Since the number of possible parameter choices are far too many for it to be possible to test every alternative, some kind of theory about how the different parameters effect the final result is desired. The solution attempted in lack of such theory has been sampling how the clustering results change with some of the parameters adjusted. The intention has been to determine combinations of parameters that perform better and use these to eventually form a theory of how the parameters interact.

2.5.1 Sampling Resulting Quality and “Hill Climbing”

The essence of the method to sample the results in a small neighbourhood is calculating an approximation of the derivative for a certain set of parameters. The approximate derivative can then be used to determine what change of parameters will result in a “hill climb” and better final result. The downside of the method is that the sampling points need to be close to give an accurate view of the parameters. During the project a massive number of parameter choices has been investigated. Chapter 5 that presents the actual experiments covered in this report only includes those parameter choices that were determined to be useful to building a theory of how the parameters interact.

2.5.2 Inconsistent Results and the Solution

In practice many of the experiments have been downright confusing. A typical clustering experiment ended with a number of clusters that had a few economy texts and all the other texts assembled in a very large cluster with around 80% of the texts. The average result from the first months was a few percentages improvement over a truly random clustering. Different parameter changes did effect the final results, but the benefits have been very small and parameters failed to deliver the expected improvement suggested from standalone tests.

Analysis at the completion of the project suggests that most of the difficulties encountered can be traced back to a few programming errors in the JavaSDM and that the clustering program was designed to not normalizing the centroids that describe the average of a cluster. When these errors had finally been identified and corrected there was a massive improvement in quality of the clustering. Unfortunately this means that all experiments done before the last weeks of the project are not reliable. Limited time prevents all experiments from being repeated, as the full planned project time had already been used up. The decision of what experiments that should be repeated are based on what experiments showed most promise earlier.
2.5.3 Roadmap of the Report

The roadmap for the report is as follows:

- Chapter 1 is intended to present the problem domain, terminology and the theoretical background needed to understand the goals of the project.

- Chapter 2 presents the goal of the project, method used and roadmap of the rest of the report.

- Chapter 3 presents theory for the Random Indexing method. Random Indexing has been used in many successful situations, but the theoretical understanding and investigations how the representation performs with real language data is limited. This chapter focuses on those aspects that have turned out to be interesting for this project.

- Chapter 4 discusses how Random Indexing word representations can be used to build text representations. Most of the difficult design choices and questions that have taken long time to resolve is covered in this chapter.

- Chapter 5 includes selected experiments with results, discussion and analysis that illustrate the theories from chapter 4. The chapter also describes the evaluation set in more detail and a practical investigation of how different versions of the JavaSDM uses allocated dimensionality.

- Chapter 6 brings conclusions and suggestions for future research.
3 Random Indexing

This chapter presents the theory of the Random Indexing method in more detail and aspects of the parameter choices.

The point with Random Indexing is that it allows us to reduce the dimension of a word-space at little computational cost. The efficiency of the method gives us freedom to investigate and compare many different kinds of contexts around words down at the word level. The problematic part of using the method is that what should be the context of a word is not given. Since other dimension reduction methods typically do not operate at the word level, but instead perform statistical analysis to find the concepts, there is little prior research of what the context for a word should look like to aid different linguistic tasks.

3.1 Theory

The word-space model in general builds on the idea that it is possible to detect difference in meaning between texts by measuring differences in the set of words used by the texts. Random Indexing applies the same reasoning at the word level. The meaning of a word is not fully contained in the words that it usually occurs with, but it is reasonable to assume that two words are similar in meaning if the words are used together with the same set of words. From a linguistic perspective there is hope that the words found close to a word approximate the real linguistic context of the word that for obvious reasons would be useful for clustering purposes.

The goal of the word-space methods is to produce numerical vectors that are aimed mostly in the same direction if the words are similar and in different directions in other cases. The actual location of a particular word is not interesting in itself unless compared with other words (Sahlgren 2005). From this follows that distortion of locations in the word-space are acceptable if the approximate distances between words are kept.

3.1.1 Representation

Each word encountered during Random Indexing processing will be given an index vector with a dimensionality that is predetermined, typically between a couple of hundreds to a few thousand elements. The length of the index vector decides the final dimensionality of the index. These vectors are filled with zeros, except for a handful of randomly distributed +1s and -1s. The index vectors produced in this fashion are only almost orthogonal, but distances between words are approximately kept provided the dimensionality of the vector is large enough.

How many randomly +1s and -1s that are used in the index vector is a free parameter except they must be an equal number of each kind. JavaSDM defaults to 4 positive and 4 negative. This parameter has an impact on how the random noise of the index is spread over the words. Very few non zero entries give fewer words with random disturbances, but greater disturbances when they happen. There is probably some optimum level, but in practice users of Random Indexes settle for some choice and stick to it as long as the random disturbances are limited.

Each word is also given a context vector that records the words encountered in the context window of the word. If a particular word is found inside the context window its index vector is added to the context vector of the word. By letting the context window slide through the text all contexts of all words are recorded in the index.

3.1.2 Different Kinds of Similarities

One important thing to observe is that due to its discovery method Random Indexing will not be able to tell what kind of similarity it records (Sahlgren 2005). An example would be that the word pairs good and bad or rich and poor have the opposite meaning, they are antonyms, not
synonyms. Experience suggests that such words often will occur in similar looking contexts. We can often replace the word with its antonym without making any further adjustments to the sentence structure.

Even if it is expected the antonym pair to be similar to each other due to this phenomenon it is reasonable to expect that there will be some difference. A word with negative implications like “catastrophe” will probably occur with other words that are negative in meaning.

### 3.1.3 Function Words

Function words, that is words like and, the and or do contribute very little information unless looking at phrases. With Random Indexing the high frequency nature of the function words raises additional problems. There is much more statistical use of function words and their high frequency mean that they occur in the context window of many words. It is the index vectors of the words found in the context window that are combined to described distributional profile of the word.

In essence Random Indexing replace the encountered word with the sum of the index vectors of all words encountered at any time close to word. For words that are descriptive this approach is exactly what is desired since there is hope these words describe the concept of the word. For function words this approach is more questionable since all documents will receive the same massive set of function words since these occur with most other words. This potentially affects the average similarity between documents since all texts will receive some similarity to other texts from each occurrence of a function words. Filtering function words as described in section 3.5 of the report is most likely essential for a functional operation of a Random Index.

### 3.2 Context Window Size

Every word found will be described by the contexts on both sides of the word. The context window used in the JavaSDM ignores sentence boundaries. The size of the left and right context can be determined independently, and are thus free parameters. Random Indexing can use a non symmetrical context window, but I have not found any theory nor any actual experiments that supports this to be beneficial.

A very small window can be expected to miss important parts of context while a too large window will make too many words associated with each other.

As a baseline in the experiments 4+4 was initially used since this was the default in JavaSDM when the project started. Actually further research has shown this value might be suboptimal as a starting point. Karlgren and Sahlgren (2001) has done experiments on the TOEFEL-test (test of English as a foreign language) and investigated the impact of different window sizes. The conclusion from that investigation indicates that when proper stemming had been done the 3+3 gave a 72% success rate compared to the 66% achieved for 4+4. With a more crude truncation approach or no attempt at stemming at all, the best results were achieved with a window size of 2+2.

There is of course no guarantee that the optimal window size is the same in Swedish and English (or for that matter for other English corpora or sets of Random Index parameters). For a more detailed examination of how window size affects the clustering see experiments in chapter 5 of the report. When the experiments were repeated after bug fixes the size 3+3 was used.

### 3.2.1 Different Types of Weighting Schemes

A weighting scheme can be used to give the words in the context window a different impact on the word's context vector. There are three major types of weighting. With constant weighting all the words are given the same importance. A second type of weighting is that words can be given a weighting factor that depends on the distance to the focus word in the middle of the context
window. This counters the effects of a large context window since the impact of the outer words become dampened. Thirdly words can be given a weight that depends on how unique they are.

### 3.2.2 Earlier Use of Weighting Schemes

Constant weighting and a number of dampened weighting schemes has been used in many previous experiments. One interesting result is that constant weight in the context window performs better than dampened weight with synonyms and association tests between different words (Sahlgren 2006). Synonyms and association between words are likely to be beneficial to clustering and thus constant weighting has been used as the baseline in the experiments.

Weighting based on the uniqueness of a word has not been tested before this project started, but a recent Australian study investigates bad performance from Random Indexing when it is used on very large corpora and employs weighting with TF-LogIDF to improve the result (Gorman and Curran 2006).

### 3.3 The Cosine Similarity

The similarity between vector directions can be investigated by comparing the angle between the vectors. With vectors pointing in the same direction the angle is zero, and if the vectors are opposite the angle is 180 degrees. Since the index in the vectors represents sets of words the angle can be used to approximate the similarity between words in a word space.

Working with actual angles is not very intuitive, the meaning of a 12 degree difference between two sets of words requires some serious explanation. The practice is, therefore, to not compare the angles directly, but instead the cosine value of the angles. This gives 1 in similarity score between word vectors that are identical, and a lower score if there is a difference. As an extra benefit the cosine of the angles can easily be computed as the inner product of the vectors after they have been normalized.

In standard use of word-spaces reported in literature the cosine similarity will be zero for documents that are not similar and 1 if the documents are exactly the same. This builds on the assumption that all the entries in word-space representation are positive, for instance they constitute term frequency.

A word-space model like Random Indexing uses random weights that are on average zero. This means the cosine measure can get a negative measurement when two words happen to have overlap in the random index context vectors but with opposite sign.

### 3.4 Dimensionality

A very common question on the topic of word-space models is if the dimensionality has any connection to features of the natural language itself. With Random Indexing there is no clear connection of this kind. On the other hand the dimensionality does quite obviously matter. It is impossible to discover differences in similarities if these are smaller than the distortions caused by the Random Index process that is dependent on the dimensionality.

In practice all experiments with Random Indexing report that at some point increased dimensionality stop affecting the actual quality of the result. The dimensionality where this saturation effect happens differs between experiments. Hassel (2007) reports that he has tested dimensionalities between 250 and 1000 for his HolSum software without detecting any real difference in performance. It seems like a reasonable hypothesis that clustering and summarization deal with texts in about the same way and thus 1000 in dimensionality has been chosen as a baseline in the experiments even though some sources report some small improvement to be seen at higher dimensionalities.
3.4.1 Dimensionality Linked to Document Representation Errors

Keeping in mind that Random Indexing previously has not been used for clustering there is some interest to measure what dimensionality choice is needed for a clustering to be successful and at what point a saturation effect occurs. Unfortunately the limited time given to work on the project means that this question has had to be given low priority.

How to measure the quality of the representation in a task like this is actually not trivial. As mentioned in chapter one of this report Kaski (1998) has shown the random distortion to be zero on the average and with a variance less than 2/dimensionality when the dimensionality is large, for the random projections he uses.

It is important to note that Kaski’s result is for singular word pairs. It is an open question what level of random distortion is acceptable to a context vector that describe the whole document based on the context vector of singular words. How the statistical profiles for a number of terms, that all are semantically connected, does combine is far from obvious.

One possible measure of random distortion for a whole document is to compare the document’s representation with words that do not exist in the document. Words that occur in similar contexts as the words in the document will be similar to these words, but it is reasonable to assume some words have no connection with the document.

These words with no connection would normally be expected to get a cosine similarity score of zero, but with Random Indexing they sometimes end up with a negative score due to opposite sign on the same index in the word context vector. Experiments with the dimensionality parameter indicate that the absolute value of those cosine scores that are negative have a value that is proportional to the ratio of information and dimensionality in the Random Index.

Unfortunately there is no theory of how these kind or errors would translate into reduced performance of the clustering and limited time has prevented a detailed study of how the clustering is affected.

3.4.2 Negative Similarity Caused by Dimensionality Size

A small informal investigation shows that for tested choices of dimensionality the observed negative cosine scores are small. A few numbers may help to demonstrate how Random Index performs with real language data. In these experiments one of the text sets intended for the clustering with 2500 texts and about 30000 unique words was examined. Function words and words that appear in one or two documents were removed in order to reduce the running time to an acceptable level.

With around 9000 words left the similarity between all words becomes easier to calculate. Unfortunately the number of possible word pairs is still massive, about 1120 megabyte of data if the similarities themselves also should be stored.

To work around the memory requirement a program was developed that stores the found similarities in a number of files on the disc. For instance a word pair with 0.23 in similarity would be stored in a file with the extension 02 while 0.199 would be stored in a file with the extension 01. The results can still in some cases become too massive to be viewed in any ordinary editor, but the file with negative cosine score can be opened without problem.

Table 1 give the average results for those word pairs that have a negative similarity. The table include dimensionalities between 1000 and 6000 since this is the range typically reported in previous experiments on Random Indexing.
### Table 1 Number of Words with Negative Similarity

<table>
<thead>
<tr>
<th>Dimensionality</th>
<th>Number of words pairs with negative cosine score</th>
<th>Average cosine score for those with a negative score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.07%</td>
<td>-0.017</td>
</tr>
<tr>
<td>2000</td>
<td>0.03%</td>
<td>-0.013</td>
</tr>
<tr>
<td>4000</td>
<td>0.02%</td>
<td>-0.009</td>
</tr>
<tr>
<td>6000</td>
<td>0.02%</td>
<td>-0.008</td>
</tr>
</tbody>
</table>

### 3.5 Preprocessing Before Random Indexing

JavaSDM has been configured to remove a list of common words assembled by Docent Jussi Carlgren at Swedish Institute of Computer Science (SICS). Besides function words the list also contains a number of words that are believed to provide little information about the topic of the document.

JavaSDM also supports contacting the Granska service at KTH that can provide lemmatizing for Swedish texts. Rosell (2005) argues that stemming might be more beneficial to clustering than lemmatizing, but with no direct support in JavaSDM for stemming for Swedish the decision was made to stick with lemmatizing the texts.
4 Document Representations

This chapter presents the model for how a document representation can be built. The design choices are explained, but actual effects of using these ideas will be presented in chapter 5.

The Word-space model itself builds on the idea that it is possible to detect difference in meaning between documents by measuring differences in what words the documents use. Random Indexing apply the same reasoning to the words themselves. If combing the Random Index context vector for many words intuition suggests that it turns into a bag-of-concepts instead of bag-of-words, but should be able to proceed in the same way. The final similarity between two documents can be calculated by the cosine similarity of the documents' representation vectors.

A representation of this kind can be built by scanning the text and combining the context vectors for the words found. A document representation of this kind becomes a very dense vector with the same dimensionality as the context vectors of the Random Index.

The most basic way to combine the context vectors is addition of the raw values of the context vectors. The benefit of this approach is the great simplicity. There is no need to value the importance of different features of the words found. Alternatively, the individual weights in the context vectors can also be scaled by some number before they are added to the document representation. This allows us to construct a document representation that favour other features than large term frequency.

An explanation of why raw addition of the context vectors might not work is that each vector contains the words that occur with the word. If a particular word exists in the context vector of many words it will be added many times to the document's representation. Looking at the final context vector it is impossible to tell if a word is strongly represented because some words in the text have a strong connection with the word or if all words have a weak connection with the word.

4.1 Normalization

One tricky design choice is if the context vectors of the words should be normalized before they are added to the document representation. Often in an information retrieval situation the user want the text vectors normalized since otherwise there is a bias for long texts due to their higher frequency count. With Random Indexing it is also true that some words have a higher frequency count, but this is from more frequent use in all documents and not because a particular document is longer. It is not given that it is beneficial to give a word that occur often equal importance to a word that is used a small number of times.

Detailed study reveals that Normalization serve a slightly different purpose for Random Indexing. It is not document length that Normalization targets but rather what part of the usage of the word that the representation focus on. With Normalization the impact of a particular context to a word's context vector is decided by how common it is among all contexts for that particular word. Normalization will promote the dominant use of the word and suppress less common meanings because the weight of 1 is spread proportionally over all contexts of the word. Without Normalization the context vectors of the words are proportional to how frequently the word is compared to all other words.

An example would be that the phrase living in a yellow submarine is probably pretty rare compared to other uses of the word yellow. With Normalization the word submarine will be given little importance in the context vector and the connection between yellow and submarine is limited. Without Normalization the connection between yellow and submarine becomes
greater since the phrase is commonly enough quoted to signal a connection between the words if it is compared to less common words than *yellow*.

The decision about whether to normalize only applies when combining the context vectors of many words before comparison with the cosine measure. The cosine measure itself will always normalize the vectors when it calculates its score. Only when dealing with the sum of a number of words context vectors is there a choice if to normalize or not part way through the process.

### 4.2 Preprocessing of Numbers

It is questionable to what degree a number in itself gives content information. To learn that two documents contain the number 5 will most likely not tell us anything about what the documents are about. On the other hand to learn that two documents contain the number 2008 does allow us to guess pretty well that they are about the same year. The choice was made in this project to filter out all numbers to keep things simple. This filtering was done during creation of the document representations since JavaSDM does not have any direct support for such filtering. An alternate idea that has not been explored would have been to replace all numbers with a special marker that signals the sentence had a number at that position.

### 4.3 Filtering of Low Frequency Words

Words that only appear in very few of the documents can for obvious reasons be negative to clustering. The ideal words to work with from a clustering perspective are those that are used in every text about a certain topic. Words that are used in a few of the texts risk making those texts similar, but prevent the clustering algorithm to merge them with other groups of texts that are about the same topic. For a standard word-space representation that works by comparing frequency counts this is a very real concern.

If the same situation appears for Random Indexing based clustering is not clear. Synonyms will be similar even if one of them is rare, and words that are not frequently used have context vectors with the random labels of few other words and are thus unlikely to produce much random interference. On the other hand there is little statistical evidence for the rare words which potentially makes the estimation of their similarity with other words uncertain. Sahlgren and Carlgren (2001) report that in their TOEFL test they received the best results when they removed words which appeared in one or two documents.

### 4.4 Filtering of High Frequency Words

Function words are removed by the JavaSDM package itself, but other words that are more frequent than the average can also create problems. Intuitively this is easy to understand; a word that exists in more categories than one will make it more difficult for the algorithm to tell the categories apart. The problem is that this observation gives little input into what frequency limit that is preferable.

Additional support for the idea that high frequency words might create problem is the results shown by Gorman and Curran (2006). They demonstrate that frequency scaling in the form of TF-LogIDF is needed to address the problem that Random Indexing does not perform well with a corpus of many millions of words. Without such a scaling Random Indexing stop benefiting from a larger corpus when the corpus size passes 10 million words. With TF-LogIDF scaling they report a 100% improvement of accuracy on a 2 billion word corpus.

The corpus used in this project is of course much smaller, yet the fact that TF-log-IDF solves the problem for large corpus gives a strong clue about a weakness in the Random Indexing method. Essentially with Random Indexing the context vector contains all words that appear
around the word. With a very large corpus the statistical evidence of the more frequent words means that they start to dominate the context vectors.

The reason why this might concern this project, even though the newspaper articles only contain a few hundred thousand words, is that the use of words in a text are most certainly not independent. When adding together the context vectors of many related words the impact of highly frequent words is increased since there is an overlap between the context of words that appear together. Normalization and scaling the weights with LogIDF are two ways to address this problem, but filtering highly frequent words could also help.

4.5 Scaling with LogIDF

Adjusting the components of the vector representation by the LogIDF of the word is a common method within Information Retrieval (see Section 1.4.2) to improve results. There is an obvious connection between the LogIDF and filtering of low and high frequency words. The LogIDF will focus the representation on those words that have a very low frequency and lessen the impact of high frequency words. Beyond that it is very difficult to say anything in advance about the effect of scaling the weights with LogIDF.

4.6 The Effect of Repeated Words

If the same word is found a number of times in the document adding it every time will affect how strongly the word's context is recorded in the document representation. The basic effect is that adding the context vector again and again approximate a scaling by term frequency (see Section 1.4.2). In the information retrieval situation the term frequency is highly useful since a text that uses the term many times is more likely to be on topic for the search. Often this is combined with a Normalization of the length of the text to prevent longer texts from scoring better due to them being longer.

The crucial question at this point is if scaling by term frequency aids text clustering based on Random Indexing. An argument in favour of term frequency scaling is that such scaling will make a document using many distinguishing words have a representation more focused on those distinguishing words. On the other hand it is possible that the documents that benefit most from this would end in the right cluster anyway due to them having many words that mark them as belonging to the category. What happens with documents that are difficult to classify is what decides the outcome of the clustering and it does not seem certain that scaling by term frequency aids the classification of those documents.

A reasonable hypothesis is that this kind of approximation of term frequency is beneficial only if it is combined with scaling with the LogIDF, but beyond that there is little that can be said in advance as to how term frequency interact with other parameters in this clustering task.

4.7 Similarity Signatures or Random Index Data

When a set of context vectors has been assembled for the texts the only real design choice left is how to present the information to the clustering software.

4.7.1 Random Index Data

The most straightforward way to represent the texts is to adopt the actual Random Indexing context vector for the document as the representation. This give a vector of the same length as the Random Index dimensionality.

The downside of this kind of representation is that it suffers from the same concerns that are encountered when words are added to the individual texts. If a particular word is found in many
documents and has a context vector that has some random disturbance, repeated addition might make the disturbance more dominant. There is no real theory that directly suggest that the above concern by necessity is a real problem, but it is clear that it is difficult to tell what happens if a cluster contain very many documents.

4.7.2 Similarity Signatures

One alternate way to represent the texts for the clustering software is to produce a representation that describe the documents relation to other documents instead of the actual text contents. One way to produce such representation is to describe all documents by a vector that indicate how similar the document is to rest of the documents. This kind of representation gives a signature that describes the document. The length of the signature is proportional to the number of texts that are to be clustered.

The resulting representation is with Similarity Signatures a point in a highly dimensional space where texts that are similar to each other are closely located. The points also have the property that two texts are similar not only due to how similar the texts are, but also due to what degree they are similar to the same set of texts.

One of the more obvious downsides of this kind of representation is that the representation grows with the number of documents. Another concern is that this kind of representation perform a second projection beside the random projection used by Random Indexing. This is something that potentially makes it more difficult to analyse what the resulting program does mathematically.

The reason to still consider this kind of representation even with these concerns is that this representation has one attractive property. Calculation of the similarity between pairs of text will mean avoiding running into the problem of not knowing what happens when adding a large set of Random Index representations that might contain the same random disturbance.
5 Experiments, Results and Discussion

This chapter describes the experiments of the project and the motivation for choosing these particular experiments.

Preferably the set of experiments performed should include every possible combination of parameters if the interaction of parameters is not fully understood. Limited time has prevented such an exhaustive test, instead a number of experiments have been selected that hopefully will highlight interesting features of Random Indexing based document representations.

5.1 Evaluation Set

Originally the project plan included experiments on texts from both “Aftonbladet” and “Dagens Nyheter” from KTH News Corpus. With the discovery that all experiments had to be redone close to the project’s completion the choice was made to ignore the articles from the newspaper “Aftonbladet” to limit the number of experiments. The choice of only “Dagens Nyheter” is mostly arbitrary, but there is hope that the articles from this newspaper are more well written since most readers consider this to be the more serious newspaper of the pair.

5.1.1 The Effect of Document Length

Texts with just a few words are inherently difficult to cluster with any representation that compares raw occurrence of the words to determine similarity. With longer texts there is reasonable hope that most texts will have some set of words that is typical for the category. If the text has only a few words the likelihood that the text features any of the typical words for the category is by necessity smaller.

It is not clear if this limitation applies equally to a Random Indexing based clustering attempt. With Random Indexing each word carries information about how it is used in general and thus have more statistical evidence than what is found in the short text. To limit the extension of the project the decision was made to remove texts shorter than 20 words to simplify comparison of the results.

5.1.2 Slow Performance

A practical problem with performing the experiments was the design choice to build separate programs to handle the random index and the clustering. Random Indexing is very effective computationally and a full index can be built in the matter of minutes (in practice the program spends most of the time waiting for the lemmatizing software to respond).

The available clustering software on the other hand was not designed around manipulation of Random Indexing vectors, but instead focuses on effective processing of regular bag-of-words representations. The clustering software could be configured to import foreign numerical data, but with a severe hit on performance.

In practice the greater collection of articles from “Aftonbladet” could not be clustered in reasonable time. To overcome this problem the decision was made to use random extracts of 2500 texts that have been used for the same purpose in earlier experiments by my supervisor Magnus Rosell. To simplify comparison between clusterings of articles from “Aftonbladet” and “Dagens Nyheter” a program was constructed that could generate random extracts of 2500 texts from the “Dagens Nyheter” part of the corpus.
5.1.3 Categories and Text Sets

After removal of short articles the corpus contained 6954 texts from the newspaper "Dagens Nyheter". These are divided over the categories finance, national, culture, sports and international. From this set of texts random selections were done with the aid of a program written in Python. Table 2 describe the names of the text sets that have been used in the experiments found later in this chapter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Selection method</th>
<th>Number of texts</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomDN1</td>
<td>Dagens Nyheter</td>
<td>Random sampling</td>
<td>2500</td>
</tr>
<tr>
<td>RandomDN2</td>
<td>Dagens Nyheter</td>
<td>Random sampling</td>
<td>2500</td>
</tr>
</tbody>
</table>

5.2 Fundamental Parameters

It seems plausible that some parameters for the Random Indexing process have more impact on the final clustering result than other parameters. The size of the context window is clearly of great importance since it dictates what words are used to describe the context of a word. The weighing scheme that determines what level of importance is given to the words in the context window is also potentially very important.

5.2.1 Context Window Size

Three context window sizes have been selected as interesting. JavaSDMs default context window size, with three words to the left and three to the right, are the obvious baseline. Experiments by Sahlgren (2005) indicate that even smaller window sizes only benefit part-of-speech tagging but in general were negative for other language tasks. Sahlgren only submits his results as examples to back his main thesis, but the results seem consistent enough to support the conclusion that it seems unlikely that even smaller window sizes would benefit a clustering task.

The second window size selected is six words to the right and six words to the left. The idea is to capture the whole sentence when the sentence is short. It is worth noticing that increases in window size affects dampened and non-dampened weighting schemes differently. The edge of the context window will with dampening have smaller and smaller impact as the window size grows. Without dampening there is equal importance to words directly adjacent to the word and those farther away no matter the window size.

The third window size used is ten words to the right and ten to the left. Since the preprocessing for Random Indexing will remove all sentence markers, this mean a significant part of the context window might contain words from the sentences around the sentence with the focus word.

5.2.2 Examined Weighting Schemes

What weighting scheme to use for the Random Indexing is an important design choice. The following weighting schemes have been used in the experiments.

- Inverse dampening: weight = \( \frac{1}{d} \)
- Exponential dampening: weight = \( 2^{1-d} \)
- Constant: weight = 1
5.3 Main Experiment

A number of experiments have been performed. Since it is useful to have a baseline to compare the results against, the final experiment is reported first followed by the rest of the experiments.

5.3.1 A Random Index Representation vs a Standard Representation

The most interesting question is without doubt how a Random Index based representation performs compared to a representation that uses only the actual words in the text.

5.3.1.1 Experiment Setup

The cluster sizes 5, 10, 15, 25, 50 and 100 have been investigated. One set of tests used Random Indexing with the best set of parameters found in other experiments. The other set of tests used the same cluster sizes, but with the lemmatized word lists. In both cases the clustering algorithm is K-Means.

The parameter choices used for Random Indexing are a dimensionality of 1000, the weighting scheme Incremental LogIDF, filtering of words that appear in less than 3 documents, scaling of weights with TFIDF and Normalization and a context window that goes 10 words in every direction. Cluster centroids are built using the method Similarity Signatures or with Random Index Data (see Section 4.7).

A possible problem in this comparison is that parameters for clustering with a regular representation have not been as extensively customized as the Random Index parameters have been.

5.3.1.2 Results

The results of the experiment are presented in table 3. The measure used for comparison of quality of the clustering is NMI (normalized mutual information). For each NMI there is also a noted standard deviation in the column Std.

An estimation of the reliability of the measurement is that statistically the true value of NMI can be expected to be found within a confidence interval of two standard deviations away from the average NMI with a 95% probability.

If two measurements have confidence intervals that are overlapping that means it is impossible to tell if the number of performed measurements are too few for an accurate comparison of the methods or if the methods themselves give equal quality.
Table 3 Clustering achieved in the main experiment

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Textset</th>
<th>Random Indexing Similarity Signatures</th>
<th>Random Indexing Random Index Data</th>
<th>Ordinary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NMI</td>
<td>Std</td>
<td>NMI</td>
</tr>
<tr>
<td>5</td>
<td>RandomDN1</td>
<td>0.446</td>
<td>0.007</td>
<td>0.453</td>
</tr>
<tr>
<td>10</td>
<td>RandomDN1</td>
<td>0.451</td>
<td>0.018</td>
<td>0.423</td>
</tr>
<tr>
<td>15</td>
<td>RandomDN1</td>
<td>0.440</td>
<td>0.008</td>
<td>0.435</td>
</tr>
<tr>
<td>25</td>
<td>RandomDN1</td>
<td>0.418</td>
<td>0.006</td>
<td>0.420</td>
</tr>
<tr>
<td>50</td>
<td>RandomDN1</td>
<td>0.386</td>
<td>0.002</td>
<td>0.385</td>
</tr>
<tr>
<td>100</td>
<td>RandomDN1</td>
<td>0.363</td>
<td>0.002</td>
<td>0.385</td>
</tr>
<tr>
<td>5</td>
<td>RandomDN2</td>
<td>0.423</td>
<td>0.006</td>
<td>0.414</td>
</tr>
<tr>
<td>10</td>
<td>RandomDN2</td>
<td>0.429</td>
<td>0.019</td>
<td>0.397</td>
</tr>
<tr>
<td>15</td>
<td>RandomDN2</td>
<td>0.418</td>
<td>0.011</td>
<td>0.413</td>
</tr>
<tr>
<td>25</td>
<td>RandomDN2</td>
<td>0.403</td>
<td>0.006</td>
<td>0.391</td>
</tr>
<tr>
<td>50</td>
<td>RandomDN2</td>
<td>0.376</td>
<td>0.003</td>
<td>0.369</td>
</tr>
<tr>
<td>100</td>
<td>RandomDN2</td>
<td>0.353</td>
<td>0.002</td>
<td>0.347</td>
</tr>
</tbody>
</table>

5.3.1.3 Analysis

The performance of Random Index based clustering is found to be equal to the regular clustering for 5, 10 and 15 clusters. For 25 clusters the Similarity Signature representation is better while the Random Index Data representation is better for RandomDN1 but overlapping for RandomDN2. For 50 and 100 clusters the Random Index based clustering performs better. In general the representation that uses Random Index Data seems to have a slightly higher standard deviation than the Similarity Signature representation, but this might be a statistical fluke.

To speculate about possible reasons why the Random Indexing based representation has “problems” with few clusters is difficult without more information about the used representation. One possible way to evaluate what the Random Index representation really contains is to examine the average similarity between categories.

Table 4 presents the results of a small experiment that examine the average similarities for the dataset RandomDN1, weighting scheme LogIDF and a context window size of 10+10.
Table 4 Average similarity between categories

<table>
<thead>
<tr>
<th>Average similarity</th>
<th>Finance</th>
<th>National</th>
<th>Culture</th>
<th>Sports</th>
<th>International</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance</td>
<td>0.745</td>
<td>0.740</td>
<td>0.741</td>
<td>0.731</td>
<td>0.719</td>
</tr>
<tr>
<td>National</td>
<td>0.740</td>
<td>0.842</td>
<td>0.823</td>
<td>0.816</td>
<td>0.811</td>
</tr>
<tr>
<td>Culture</td>
<td>0.741</td>
<td>0.823</td>
<td>0.834</td>
<td>0.823</td>
<td>0.797</td>
</tr>
<tr>
<td>Sports</td>
<td>0.731</td>
<td>0.816</td>
<td>0.823</td>
<td>0.855</td>
<td>0.787</td>
</tr>
<tr>
<td>International</td>
<td>0.719</td>
<td>0.811</td>
<td>0.797</td>
<td>0.787</td>
<td>0.811</td>
</tr>
</tbody>
</table>

One conclusion that is fairly obvious from the averages presented in table 4 are that the line between categories is in some cases very narrow. International texts are on average quite similar to other international texts and national texts. Finance texts are just barely more similar to each other than to their similarity to national and culture texts.

It seems plausible that these similarities show why the Random Index based representation have problems with a small number of clusters. The question about why this happen is difficult to answer. One possibility is that the categorization used is faulty in the sense that the culture department of the newspaper might review a book about foreign events. Another possibility is that some words are commonly used in both parts of the newspaper and make the categories very similar.

5.4 Complementary Experiments

The goal in these experiments has been to try to identify design aspects that highlight how a document representation based on Random Indexing should be designed. In most cases the measurements have been averaged to find the general tendency and they are thus mostly useful to compare different methods towards each other.

5.4.1 How to Combine Words

One important design aspect is the question of how to combine the words to get the best possible final clustering. If different weighting schemes benefit from different kinds of scaling is an open question. Preliminary testing that has been done gives an indication that this is not the case, but there was no time to repeat these experiments during the time allocated for the project (see section 2.5 for details).

5.4.1.1 Experiment Setup

The weighting schemes Constant and LogIDF has been used in these experiments. The examined methods are:

- Normalization (see section 4.1)
- Removal of high frequency words (see section 4.4)
- Scaling with LogIDF (see section 4.5)
- Removal of repeated words (see section 4.6)

Adjustment of how many documents a word must appear to be seen as useful as described in section 4.3 is a parameter that has not been investigated in these experiments. The decision to exclude adjustments of this parameter was based on that it showed limited impact on the clustering quality in the initial experiments. Unfortunately there is no way to know if this result still holds for the better Random Indexing representation used in the final experiments.
For removal of high frequency words three levels of filtering have been examined. The baseline is no such filtering. By examining the word frequency list for the documents two more filtering levels were decided. A manual inspection of the words with higher frequency count than 400 (16% of the total documents) indicates that none of these can easily be associated with any of the categories. This filtering level removes about 50 unique words from the documents. Secondly, the filtering level 250 was selected. It is 10% of the total number of documents. The word list contains around 50 words in the frequency interval 250 to 400.

All measures have been calculated over 10 runs of K-Means for the text sets RandomDN1 and RandomDN2. The clustering sizes 5 and 10 have been used. The context window size used is 10+10. Cluster centroids are built using the method Similarity Signatures (see Section 4.7).

5.4.1.2 Result

As a general baseline to these experiments raw addition of the word vectors without any scaling gives a Normalized Mutual Information of 0.061 for Constant as the weighting scheme and 0.109 for LogIDF as the weighting scheme (both with a standard deviation of 0.006). Most of the examined methods improve the result greatly.

The positive news from this experiment is that none of the proposed scaling methods seems to contribute negatively to the clustering result. The bad news is that the number of performed clusterings are too limited to allow full evaluation of the investigated parameters. In many cases the uncertainty intervals are overlapping.

Normally it is impossible to know if overlapping intervals are caused by a fundamental randomness of the method or if they are caused by the measures being calculated over too few clusterings. In this case a limited analysis is possible due to the observation that the methods perform very consistently over all the experiments. One example would be that Normalization is the top alternative within each group even though all the uncertainty intervals in the group are overlapping.

With the pattern of how the methods perform in mind the choice has been made not to present the individual clustering results, but rather averages calculated over groups of scaling methods. The benefit of this approach is that the uncertainty in form of the standard deviation lessens when considering more clustering runs. In practice averages have been created for all possible combinations of the four methods. Those comparisons that produced any significant result are covered in table 5, 6, 7, 8 and 9.

Table 5 shows that Normalization of the words context vectors when building the text representation give substantial improvement to the quality of the clustering. Normalization gives the best absolute result with the weighting scheme LogIDF even though the weighting scheme Constant get a larger relative improvement.

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Textset</th>
<th>Normalized</th>
<th></th>
<th>No Normalization</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average NMI</td>
<td>Std</td>
<td>Average NMI</td>
<td>Std</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN1</td>
<td>0.353</td>
<td>0.004</td>
<td>0.248</td>
<td>0.004</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN2</td>
<td>0.321</td>
<td>0.005</td>
<td>0.225</td>
<td>0.002</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN1</td>
<td>0.445</td>
<td>0.003</td>
<td>0.377</td>
<td>0.005</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN2</td>
<td>0.428</td>
<td>0.003</td>
<td>0.351</td>
<td>0.006</td>
</tr>
</tbody>
</table>
Experiments, Results and Discussion

Table 6 demonstrates that scaling the words context vectors with the LogIDF contributes positively to Normalization when the weighting scheme Constant is used. With the weighting scheme LogIDF the confidence intervals are overlapping. This has been marked in the table with a star.

Care should be taken not to give too much importance to these results since it is unclear if any of the other investigated methods address the same problems in a raw Random Indexing representation. Comparison with the results reported in table 7, 8 and 9 is probably a good idea before any conclusions are drawn from this table.

Table 6 The average effect of LogIDF added to Normalization for the weighting schemes Constant and LogIDF

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Text set</th>
<th>Scaled by LogIDF and Normalized</th>
<th>No scaling with LogIDF but Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average NMI Std</td>
<td>Average NMI Std</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN1</td>
<td>0.372 0.006 0.334 0.005</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN2</td>
<td>0.341 0.008 0.300 0.005</td>
<td></td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN1</td>
<td>0.449 0.005* 0.440 0.004*</td>
<td></td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN2</td>
<td>0.433 0.005* 0.423 0.003*</td>
<td></td>
</tr>
</tbody>
</table>

Table 7 show the result of comparison of LogIDF scaling compared to Normalization. The result is clearly that Normalization gives greater benefit if you must choose only one of them.

Table 7 The average effect of LogIDF compared to Normalization for the weighting schemes Constant and LogIDF

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Text set</th>
<th>Scaled by LogIDF</th>
<th>Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average NMI Std</td>
<td>Average NMI Std</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN1</td>
<td>0.248 0.004 0.334 0.005</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN2</td>
<td>0.225 0.002 0.300 0.005</td>
<td></td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN1</td>
<td>0.377 0.005 0.442 0.003</td>
<td></td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN2</td>
<td>0.351 0.006 0.423 0.003</td>
<td></td>
</tr>
</tbody>
</table>

Table 8 shows the affect of high frequency word filtering alone. From these values it is quite clear that adjustment of high frequency filtering alone is not a method that can outperform Normalization.
Table 8 The average effect of filtering high frequency words alone for the weighting schemes Constant and LogIDF

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Text set</th>
<th>No filtering</th>
<th>250 filtering</th>
<th>400 filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average</td>
<td>Std</td>
<td>Average</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN1</td>
<td>0.068</td>
<td>0.009</td>
<td>0.299</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN2</td>
<td>0.055</td>
<td>0.008</td>
<td>0.270</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN1</td>
<td>0.012</td>
<td>0.011</td>
<td>0.410</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN2</td>
<td>0.010</td>
<td>0.006</td>
<td>0.367</td>
</tr>
</tbody>
</table>

Table 9 shows the results of the combination LogIDF and filtering of high frequency words. Those measurements that have overlapping uncertainty intervals with corresponding measurements for only filtering out high frequency words (table 8) have been marked with a star.

Table 9 The average effect of filtering high frequency words combined with LogIDF scaling for the weighting schemes Constant and LogIDF

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Text set</th>
<th>No filtering +</th>
<th>250 filtering +</th>
<th>400 filtering +</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average</td>
<td>Std</td>
<td>Average</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN1</td>
<td>0.119</td>
<td>0.003</td>
<td>0.308*</td>
</tr>
<tr>
<td>Constant</td>
<td>RandomDN2</td>
<td>0.107</td>
<td>0.002</td>
<td>0.280*</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN1</td>
<td>0.324</td>
<td>0.012</td>
<td>0.413*</td>
</tr>
<tr>
<td>LogIDF</td>
<td>RandomDN2</td>
<td>0.276</td>
<td>0.010</td>
<td>0.390*</td>
</tr>
</tbody>
</table>

Experiments have been performed on combining filtering of high frequency words and Normalization. Combining these two methods fails to give any detectable advantages over just Normalization. Experiments have also been performed in combining removal of repeated words with the other methods without any detectable improvement on the final result.

5.4.1.3 Analysis

Normalization is as demonstrated in table 5 clearly the method that gives best results, but the other tables indicate that a combination of the others can perform equally well. The fact that Normalization has such a strong effect on the final result suggests that the dominant use of a word (see Section 4.1) is beneficial to clustering purposes. It is at this point unclear if this is because the dominant use of the word contains more important information for the deciding the topic of the text or if focus on the dominant word as a secondary effect suppress the noise level in the representation.

Approximation of term frequency by adding repeated words contributes nothing to the quality of the final clustering in these experiments. The fact that it takes extra programming effort to
Experiments, Results and Discussion

filter repeated words suggests that keeping the words might be preferable. Section 6.3.4 further
discuss possible experiments about the affects of repeated words for long texts.

Study of table 8 and table 9 give some insight in the effect of high frequency words in the
representation. Filtering high frequency words is in itself tricky in the absence of a theory of
what words are negative to the clustering. The good news is that such filtering only seems to be
needed if the user want to avoid using Normalization for some reason.

The most interesting observation to make from table 9 is without doubt that after removal of the
most frequent words LogIDF gives very little added benefit. This observation suggest the idea
that the weight of the descriptive words is of little importance for the stability of the clustering.
It is at this point of course impossible to know if this holds true in general. In section 6.3.5 there
is a discussion about the need to research what types of words are negative to text clustering.

The most interesting thing to be learned from table 6 is actually the lack of improvement for
scaling the weight with LogIDF when the weighting scheme LogIDF is used. For the weighting
scheme Constant there is support for that scaling the weights with LogIDF is good. It seems
possible that this might indicate that the weighting scheme LogIDF and the scaling the weights
with LogIDF achieves the same effect. If this is true, it is really a rather surprising result. The
weighting scheme builds on the idea that common words are bad candidates to represent the
concept of the word. The scaling method on the other hand builds on the idea that concepts that
are used in many documents are not very descriptive. Intuition suggests that these uses of
LogIDF are applied to different problem domains.

One possible explanation as to why scaling with LogIDF fails to perform as expected is that
there might be some concepts that are common enough for them to receive a low Inverse
Document Frequency even though they are useful to determine the topics of the text. Another
real possibility is that the Random Index is built over too few texts for the Inverse Document
Frequency to be reliable. It is possible to imagine that the statistical evidence is good enough to
benefit the weighting scheme, but when using both weighting scheme and scaling method the
combination gives cause of random noise and no real extra benefit to the representation.

Finally, it is worth to notice that the weighting scheme Constant generally performs worse than
LogIDF, but the difference is small with the right combination of scaling methods. All things
considered it seems that the weighting scheme LogIDF and Normalization combined with
possibly removal of repeated words should be the standard method to combine words into a
representation. Scaling the vector components with LogIDF is something that merits future
research even if it fails to contribute much in these experiments.

5.4.2 Does Dampening Improve Clustering Quality?

One of the motivations for looking at narrow contexts around a word is that it seems intuitively
plausible that words that occur closer to each other have a stronger connection than words that
occur only in the same document. A dampened weighting scheme means extra emphasis on the
narrow context window. If clustering benefits from such emphasis is difficult to know. Sahlgren
reports that part-of-speech is the only examined language task that gains significantly from
aggressive dampening. This suggests that a non-dampened weighting is preferable when
building document representations for text clustering.

5.4.2.1 Experiment Setup

A test to determine the effect of the dampening must examine the size of the context window
since the effect of the dampening is larger at the edges of a large context window. Sahlgren
examines only constant dampening and exponential dampening since he argues that other
weighting schemes with dampening are just variations of exponential dampening. For a context
window size of 3 words this is approximately true, but with a much wider context window
exponential dampening will have a weight at outer parts of the context window that is negligible
compared to a less aggressive dampening.
Figure 2 compare the exponential and inverse dampening to each other. It is quite clear from this figure that these weighting schemes are quite different. At a distance 10 steps from the focus word inverse dampening gives a weight of 10% compared to a constant weighting scheme.

Exponential dampening on the other hand is down to 0.2% of what a constant weighting gives, or 2% of what the inverse weighting scheme gives. It seems dangerous to assume that such a large difference in weight will not have an impact on how the Random Index encode the word contexts.

Six tests have been performed. Constant dampening has been compared to inverse and exponential dampening for the context window sizes 3+3, 6+6 and 10+10. Also LogIDF dampening has been compared to LogIDF Inverse and LogIDF Exponential for the same context window sizes. Secondary parameters used are Normalization, filtering of word occurring in 1 or 2 documents and filtering of high frequency words (with the limits 400, 250 and no filtering).

The motivation to include filtering of high frequency words in this experiment was because of the concern that high frequency words might possibly have a greater impact for large context windows since the larger window is more likely to capture many of these words. Cluster centroids are built using the method Similarity Signatures (see Section 4.7).

5.4.2.2 Results

The measures have been calculated over 10 K-Means clusterings for 5 and 10 clusters. Table 10, 11 and 12 contain the results for different context window sizes for the weighting schemes Constant, Inverse and Exponential.

The first weighting scheme in a column is marked as better if a 95% confidence interval can be constructed for the NMI that is not overlapping for all examined levels of high frequency word filtering. In no situation is the second weighting scheme in a column better than the first one.

For LogIDF, Exponential LogIDF and Inverse LogIDF it is not possible to construct any 95% confidence intervals that does not overlap.

Table 10 Comparison of Weighting Schemes for 3+3 context windows. A blank table cell indicates that the intervals are overlapping.

<table>
<thead>
<tr>
<th>Context window</th>
<th>Textset</th>
<th>Clusters</th>
<th>Constant vs Inverse</th>
<th>Constant vs Exponential</th>
<th>Exponential vs Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>3+3</td>
<td>RandomDN1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3+3</td>
<td>RandomDN2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3+3</td>
<td>RandomDN1</td>
<td>10</td>
<td>Better</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3+3</td>
<td>RandomDN2</td>
<td>10</td>
<td>Better</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experiments, Results and Discussion

Table 11 Comparison of Weighting Schemes for 6+6 context windows. A blank table cell indicates that the intervals are overlapping.

<table>
<thead>
<tr>
<th>Context window</th>
<th>Textset</th>
<th>Clusters</th>
<th>Constant vs Inverse</th>
<th>Constant vs Exponential</th>
<th>Exponential vs Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>6+6</td>
<td>RandomDN1</td>
<td>5</td>
<td>Better</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6+6</td>
<td>RandomDN2</td>
<td>5</td>
<td>Better</td>
<td>Better</td>
<td></td>
</tr>
<tr>
<td>6+6</td>
<td>RandomDN1</td>
<td>10</td>
<td>Better</td>
<td>*</td>
<td>Better</td>
</tr>
<tr>
<td>6+6</td>
<td>RandomDN2</td>
<td>10</td>
<td>Better</td>
<td>Better</td>
<td>Better</td>
</tr>
</tbody>
</table>

*The overlapping intervals that prevents 95% confidence interval is in this case that the Exponential clustering for high level filtering of 400 for some reason has an unusual large standard deviation compared to the other clusterings.

Table 12 Comparison of Weighting Schemes for large context windows. A blank table cell indicates that the intervals are overlapping.

<table>
<thead>
<tr>
<th>Context window</th>
<th>Textset</th>
<th>Clusters</th>
<th>Constant vs Inverse</th>
<th>Constant vs Exponential</th>
<th>Exponential vs Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>10+10</td>
<td>RandomDN1</td>
<td>5</td>
<td>Better</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10+10</td>
<td>RandomDN2</td>
<td>5</td>
<td>Better</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10+10</td>
<td>RandomDN1</td>
<td>10</td>
<td>Better</td>
<td>Better</td>
<td>Better</td>
</tr>
<tr>
<td>10+10</td>
<td>RandomDN2</td>
<td>10</td>
<td>Better</td>
<td>Better</td>
<td>Better</td>
</tr>
</tbody>
</table>

A number of observations are possible from the results in these tables. Clustering with more clusters than categories in general reduces the risk that a cluster ends with texts from more than one category. Something that is interesting is that the different weighting schemes are not able to take advantage of the extra clusters to the same degree. For instance this can be seen for Exponential compared to Inverse where the former always is better than the latter for 10 clusters even though they have overlapping intervals for 5 clusters.

The idea that high frequency words might be a problem for the non-dampened weighting scheme Constant is not supported by these experiments. Constant and Exponential are equal for 3+3 context windows, but Constant is better for larger window sizes since it is always better than the weighting scheme Inverse while Exponential is only better sometimes.

One result that it is interesting, but not directly visible from the table 10, 11 and 12, is that the weighting schemes seems to suffer differently from the presence of high frequency words. Table 13 presents an alternate view of the experiments that allows evaluation of the general pattern of the weighting schemes performance.

The table presents the average NMI for the weighting schemes, context window sizes and number of clusters. The NMI is averaged over the different filtering levels: no filtering, 250 and 400. To reduce the size of the table each average is presented with a 95% confidence interval instead of a separate standard deviation column.
Table 13 demonstrates that the weighting scheme Constant on average is better than both Inverse and Exponential weighting schemes for the context window size 3+3. The reason it is not listed as better in table 10 is that with no filtering of high frequency words it has overlapping confidence intervals. The same apply for comparisons between Exponential and Inverse.

5.4.2.3 Analysis

The motivation for these experiments was to examine if dampening is beneficial to the clustering. Logic suggest that if the dampening in the context window degrade the information that is used by the clustering algorithm, the aggressive exponential dampening would lose most information. Likewise the reverse situation would suggest that the exponential weighting would gain the advantage. The results fails to support any of these cases. The general pattern is that the non-dampened clustering performs best, but that exponential dampening performs better than the less aggressive inverse weighting scheme.

One interesting observation is that a greater context window seems to benefit the weighting scheme Constant. This result hints that the outer part of the context window contain words that aid correct clustering of the texts. If LogIDF gains similarly it is not clear from these experiments, but there seems to be a tendency for this. It is also worth to notice that even though the weighting scheme Constant gains quality from a larger context window the result is still not be comparable to what the LogIDF weighting schemes achieve.

In hindsight it is obvious that the great difference in weight between the selected weighting schemes might be what makes it difficult to draw any conclusions. The experiment aimed to examine different levels of dampening in the weighting scheme, but the selected weighting schemes have great differences in weights. The general tactic employed in this project with no theories of how the parameters interact is to sample the clustering quality for parameter choices that are close. Essentially this allows an estimation of the derivative for the neighbourhood of the sample points and what parameter adjustment that will bring us to a local maximum.

The contradictory result of the experiment might be caused by weighting schemes and window sizes being so different that they interact with different features of the documents. It is for
instance possible that there between the Constant and Inverse weightings schemes is a dampening rate that gives even better results than Constant. The reverse situation is also equally possible. A better experimental setup would probably have been to use comparably more similar window sizes (6,7 and 8) and weighting schemes more similar to each other.

One final thing that might be worth noticing is that there might be a tendency that the non-dampened LogIDF weighting scheme is better than the two dampened variants. No real conclusions can be drawn since the confidence intervals are overlapping, but this tendency matches the good performance of Constance compared to Inverse and Exponential.

5.4.3 Can Incremental LogIDF Replace LogIDF?

One of the benefits of Random Indexing is that it is an incremental method that allows us to add new data into the existing index without invalidating previous results. The results of Gorman and Curran (2006) that demonstrate the need for IDF to improve the performance of Random Index for large corpora is therefore discouraging. Calculation of the LogIDF requires sampling of all documents; something that would prevent incremental updates of the index.

One possible solution to the problem would be to have a server that provides statistical measurements for the used language so that data can be added incrementally. On the hand it is interesting to know if weighting scheme like incremental LogIDF could give acceptable results. For convenience the formula for the weights for LogIDF are repeated below. Let $m$ be the number of documents indexed so far and let $w$ be the number of documents that contain the word so far.

Incremental LogIDF: weight = $1 + \log \left( \frac{m}{w} \right)$

The reason to include a +1 in the formula is to make sure that the index learns something from the first documents where the IDF ratio is close to 1.

5.4.3.1 Experiment Setup

Clustering has been performed with K-Means for the weighting scheme LogIDF and Incremental LogIDF. Normalized mutual information has been calculated over ten runs of K-Means for 5 cluster.

The word representations has been scaled by LogIDF and Normalized. Filtering of words that appear in less than three documents has been done. There has been no removal of repeated words and no filtering of high frequency words. Cluster centroids are created either with Similarity Signatures or with Random Index Data (see Section 4.7).

5.4.3.2 Results

The compared weighting schemes perform equally well. The measurements can be found in table 14. This experiment features a rather large standard deviation compared to other experiments since the presented averages are calculated only over ten K-Means clusterings with 5 and 10 clusters respectively instead of being averages over a greater number of individual experiments.
Table 14 Comparison of an incremental and a standard weighting scheme for Similarity Signatures and Random Index Data

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>Cluster centroid</th>
<th>RandomDN1</th>
<th>RandomDN2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NMI</td>
<td>Std</td>
</tr>
<tr>
<td>LogIDF</td>
<td>Similarity Signature</td>
<td>0.442</td>
<td>0.037</td>
</tr>
<tr>
<td>LogIDF</td>
<td>Random Index Data</td>
<td>0.426</td>
<td>0.032</td>
</tr>
<tr>
<td>Incremental LogIDF</td>
<td>Similarity Signature</td>
<td>0.446</td>
<td>0.007</td>
</tr>
<tr>
<td>Incremental LogIDF</td>
<td>Random Index Data</td>
<td>0.453</td>
<td>0.009</td>
</tr>
</tbody>
</table>

5.4.3.3 Analysis

In the performed experiments there are no detectable difference between the clustering performance for the two weighting schemes. The included +1 in the Incremental LogIDF formula is included to make sure that the index acquires information from the first documents encountered when the total numbers of documents are too close to 1 for any learning to happen.

When indexing large set of texts the first documents are probably of little importance for the stability of the statistical evidence of how the words are used. In these experiment only 2500 documents are used which mean that many words are only used in a handful of documents. These words are also of necessity those words which get a very high inverse document frequency, and thus believed to be the most descriptive words for content classification purposes.

When clustering much larger sets of texts it seems plausible that the +1 is not needed, since those words that appear only a handful of times most likely are misspelled ordinary words. Yet further experiments are needed before it can be decided how much text is needed for this to be true.

One interesting result that was discovered by mistake when doing these experiments is that clustering RandomDN2 with an index that use the words statistics from RandomDN1 does not really have any impact on clustering quality. I leave it to those who perform additional experiments on this subject to evaluate if this implies anything about how the random index and the LogIDF interact.

5.4.4 Is 1000 in Dimension Acceptable?

From a practical point of view any choice of dimensionality that avoids significantly distorting the results is a good one. To determine what minimal choice of dimensionality satisfies this is a far too time consuming process for this report. On the other hand the validity of the rest of the experiments depend on the choice of dimensionality not distorting the results.

5.4.4.1 Experiment Setup

The weighting scheme LogIDF with a 10+10 context window was used to create document representations based on Random Index with the dimensionalities 500, 750, 1000, 1250 and 1500. Words with less than 3 occurrences were filtered. The words representations was also scaled with LogIDF and Normalized. Clustering was done with K-Means with measures averaged over ten runs.

Cluster centroids have been built using both the method Similarity Signatures and the method Random Index Data(see Section 4.7).
5.4.4.2 Results

The Normalized Mutual Information is about 0.423 for both text sets, all dimensionalities and both methods to build cluster centroids. Comparing the standard deviations reveals that all these are approximately equal (between 0.008 and 0.002) and all 95% confidence intervals are overlapping. Furthermore, there is no set order between the dimensionalities.

5.4.4.3 Analysis

A dimensionality choice between 500 to 1500 is clearly sufficient for these experiments since this experiment would have shown if dimensionality was a limiting factor.

5.4.5 Agglomerative Clustering

Agglomerative clustering works by combining clusters that are similar while K-Means uses the squared error to determine which texts should be moved from their current cluster. These two approaches are fundamentally different and might utilize the representation in different ways.

5.4.5.1 Experiment Setup

With no randomness involved in the agglomerative clustering process one single clustering is sufficient. Clustering has been performed on RandomDN1 and RandomDN2 with LogIDF as a weighting scheme. The weights have been Normalized and scaled with LogIDF. Words that appear in 1 or 2 documents have been filtered from the representation.

Due to the narrow lines between categories that are discussed in section 5.3.1.3 there are concerns that the final mergers of algorithm might distort the results. With this in mind clusterings have been performed with 5, 10, 15, 25, 50 and 100 clusters.

Finally there is interest in how Agglomerative clustering interacts with document representations created by Similarity Signatures (see Section 4.7). Agglomerative clustering works by building a matrix with the similarity between all texts and what happens when the similarity is calculated over signatures containing similarity ratings with the text set is far from obvious. For this reason all clusterings have been done with both ways to build cluster centroids to allow comparison.

5.4.5.2 Results

Table 15 presents the results of clustering in the Single Link mode. Single Link means that the similarity between clusters is decided by the most similar member. In practice this mean that a cluster that has grown large is more likely to become larger since it covers more of the available space. The very low NMI can easily be understood by looking at the resulting clustering. As an example, for Random Index Data and 100 clusters the largest cluster ends with 95% of the total texts.
Table 15 Normalized Mutual Information for Agglomerative Clustering with Single Link

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Similarity Signature</th>
<th>Random Index Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RandomDN1</td>
<td>RandomDN2</td>
</tr>
<tr>
<td>5</td>
<td>0.005</td>
<td>0.001</td>
</tr>
<tr>
<td>10</td>
<td>0.006</td>
<td>0.005</td>
</tr>
<tr>
<td>15</td>
<td>0.007</td>
<td>0.007</td>
</tr>
<tr>
<td>25</td>
<td>0.009</td>
<td>0.010</td>
</tr>
<tr>
<td>50</td>
<td>0.015</td>
<td>0.015</td>
</tr>
<tr>
<td>100</td>
<td>0.024</td>
<td>0.026</td>
</tr>
</tbody>
</table>

Table 16 gives the result for Agglomerative clustering with the mode Complete Link. Complete Link means the similarity between clusters is determined by the least similar texts in the clusters. A comparison how the NMI varies for different number of clusters may be slightly confusing since the NMI does not increase monotonic. Investigation of the entropy show that it increased monotonic in all cases.

One thing of obvious interest is that the representation using Similarity Signatures often performs better than the Random Index Data representations, but with the exception for 15 clusters.

Examples of what constitutes large clusters for Complete Link is rather interesting. For 100 clusters, RandomDN1 and Random Index Data as method of building representations the largest cluster has 625 texts (42.6 comes from the National section of the newspaper). The second largest cluster has 374 texts (98.1 from the Sports section of the newspaper). The third cluster has 248 texts (58.1% from the National section of the newspaper). The fourth cluster has 104 texts (89.4% from the Finance section of the newspaper). Twenty-five of the clusters have one single text. Forty-four clusters with more than one text contains only a single category.

Table 16 Normalized Mutual Information for Agglomerative Clustering with Complete Link

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Similarity Signature</th>
<th>Random Index Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RandomDN1</td>
<td>RandomDN2</td>
</tr>
<tr>
<td>5</td>
<td>0.177</td>
<td>0.084</td>
</tr>
<tr>
<td>10</td>
<td>0.169</td>
<td>0.101</td>
</tr>
<tr>
<td>15</td>
<td>0.156</td>
<td>0.148</td>
</tr>
<tr>
<td>25</td>
<td>0.341</td>
<td>0.281</td>
</tr>
<tr>
<td>50</td>
<td>0.344</td>
<td>0.328</td>
</tr>
<tr>
<td>100</td>
<td>0.331</td>
<td>0.313</td>
</tr>
</tbody>
</table>
Table 17 give the clustering results for Agglomerative clustering in the Average Link mode. Average Link means the similarity of two clusters is decided by the averages of the clusters.

Examples of what constitutes a large cluster for Average Link is that for 100 clusters RandomDN1 the largest cluster has 2016 texts (28.8% from the Sports section of the newspaper). The second largest cluster has 109 texts (96.3% from the Finance section of the newspaper). Fifty eight of the clusters have only one single text.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>RandomDN1</th>
<th>RandomDN2</th>
<th>RandomDN1</th>
<th>RandomDN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.009</td>
<td>0.009</td>
<td>0.006</td>
<td>0.002</td>
</tr>
<tr>
<td>10</td>
<td>0.060</td>
<td>0.065</td>
<td>0.008</td>
<td>0.006</td>
</tr>
<tr>
<td>15</td>
<td>0.072</td>
<td>0.062</td>
<td>0.010</td>
<td>0.009</td>
</tr>
<tr>
<td>25</td>
<td>0.127</td>
<td>0.130</td>
<td>0.011</td>
<td>0.012</td>
</tr>
<tr>
<td>50</td>
<td>0.240</td>
<td>0.127</td>
<td>0.011</td>
<td>0.012</td>
</tr>
<tr>
<td>100</td>
<td>0.253</td>
<td>0.241</td>
<td>0.093</td>
<td>0.104</td>
</tr>
</tbody>
</table>

5.4.5.3 Smarter Agglomerative Clustering

The attempts with Agglomerative clustering in this project have not been very successful. One possible way to improve the results is most likely to adjust the representation so that there are more differences between categories. On the other hand the results from the clustering attempts in this project suggest a much easier solution.

The problem for agglomerative clustering in this project is clearly that once a cluster grows above a certain limit it become statistically close to most single texts. It is not necessary to solve that problem if the large clusters are ignored when small clusters are merged.

In this experiment a Agglomerative clustering was performed with the additional rule that large clusters should be temporarily excluded from the clustering iterations until enough of the clusters have reached the same size.

Table 18 presents the result for Single Link, Complete Link and Average Link with cluster centroids assembled using the Random Data method. Using the method Similarity Signatures to build cluster centroids gives identical results. The additional rule added in this experiment greatly improves the quality of the results, even though the results is still not comparable to K-Means.
5.4.5.4 Analysis

The total conclusion from this experiments must be that good results with Agglomerative clustering are difficult to achieve for Random Index based representations. That Single Link fail to work at all for the regular Agglomerative algorithm is somewhat expected, but Average Link and Complete Link both give rather bad results compared to what intuition suggest.

One surprising detail is that for the case Average Link there is a very large difference in performance between using Similarity Signatures or Random Index Data when building the cluster centroids. Perhaps this is a consequence of the small difference in similarity between the categories that is demonstrated in Section 5.3.1.3. Without further study of why the average difference between categories is small it is difficult to speculate as to why the Similarity Signature alternative performs better.

The advantage for the method Similarity Signatures is absent from the Agglomerative clustering when using the additional rule that small clusters should be joined before larger clusters. This is something that support the idea from Section 4.7.2 about the method Similarity Signatures possibly reducing problems caused by many common words between documents.

In the absence of further experiments no real conclusion can be drawn from these experiments. If experiments on customized word lists are performed as suggested in Section 6.3.5, it might be worthwhile to evaluate the results using the Agglomerative algorithms for both representations.

5.5 Analysis of JavaSDMs Use of Dimensionality

The JavaSDM package is one of the cornerstones of this project. With this in mind proper functionality of this package is of obvious concern. These experiments investigate how the program package makes use of the dimensionality allocated to the index.

5.5.1 Statistical Terminology

With a true random projection it cannot be expected that each index will be used with equal frequency. This is a direct effect of the number of words added being bounded. It is very unlikely that by chance picking each index the same number of times when the chance for a particular index is very small and the number of words added is limited. The spread of the
selection of indexes is mathematically described by the variance. It tells how collected the data values are around the average. The variance lessens as the number of added words increases since with an infinite number of added words there will be a uniform probability distribution.

5.5.2 Reproducible Random Numbers

The programmer of a Random Index implementation may for debugging purposes find it worthwhile to have results that are reproducible. Normally this is the case with random numbers on a computer since the computer only has generated numbers that are only pseudo random. JavaSDM can not use this particular trick since it has been designed to be run on computers with more than one processor and will in those cases create the index in an order that is not equal every time. The work around used in JavaSDM is to base the generated “random numbers” on the word being entered into the Random Index. Still any attempt at having reproducible random numbers really needs to prove the random nature of said numbers. The need for this was discovered during the project when the results of certain experiments lost quality at higher dimensionality contrary to what the theory for random projections suggests.

5.5.3 Programming Errors

Careful evaluation of all involved steps during the building process revealed that the cause of the problem was how the random labels for words were constructed. The employed version of JavaSDM used the actual word in production of the random label, unfortunately the algorithm did not add enough randomness. At most dimensionality choices this would not matter, but in some it gave similarities between words that were in the range -0.8 to +1 instead of the expected 0 to +1. The negative scores occurred for about 5% to 10% of the word pairs, but having such large negative scores most certainly put all the reported similarities in doubt. Fortunately upgrading to the latest version of JavaSDM solved the problem since this version uses a more clever version of the code for generating reproducible pseudo random numbers.

5.5.4 Experimental Detection of Faulty Random Numbers

Upgrading to the latest JavaSDM version removed the encountered large negative similarities. On the other hand the basic situation is still unsatisfactory since the error only happen at some dimensionalities and evaluation of the absolute value of words with a negative similarity as done in Section 3.4.2 takes extensive time since it compares all words towards each other. The average frequency of how often the indexes are used and the variance of this measurement is on the other hand easy to compute.

A simple program was constructed that evaluates what indexes every word uses. The average activation of each index could thus be determined. Table 19 demonstrates the results of a experiment on Random Indexes with a dimensionality of 2000 and about 9000 added words using both versions of the JavaSDM and an equal set of normal pseudo random numbers.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old JavaSDM</td>
<td>37.04</td>
<td>171.2</td>
</tr>
<tr>
<td>New JavaSDM</td>
<td>37.49</td>
<td>36.9</td>
</tr>
<tr>
<td>Random Numbers</td>
<td>32.92</td>
<td>32.3</td>
</tr>
</tbody>
</table>
A simple baseline to compare against is the measurement for the same number of pseudo random numbers. The new JavaSDM has a higher variance than the random numbers, but not by very much. If this kind of test is be used for real evaluation of the quality of the Random Index it is of course imperative that comparison is done towards the average variance of a number of random number sets.

5.5.5 Effects of the Faulty Random Numbers

As described in section 2.5 two major programming errors distorted the results in the first run of experiments of the project. The second problem besides the lack of randomness was failure to normalize the cluster representations during the clustering. The net effect of these two errors in combination was to give clusterings that barely surpassed a pure random clustering. An estimate would be that on average each error removed about 0.17 of a typical NMI of 0.40. Any attempt to create reproducible random numbers should also include a comparison of the results with a true random approach to verify the correctness of the implemented algorithm.
6 Conclusions

This chapter examines the found results and tries to draw conclusions from the project and proposes what future research that looks promising.

A number of methods to improve the results of a Random Index based clustering has been discovered. Using Normalization of the words added to the representation and a weighting scheme with LogIDF scaling seems, from these experiments, to be the preferable choice. Filtering of high frequency words proves to be a possible way to improve the results in some cases, but suffers from difficulties to know where to draw the line for the filtering and the frequency of words itself is a weak indicator of how useful the word is for clustering purposes.

For a number of other methods the benefit is limited, or the area has not been enough investigated for proper evaluation. Examples of these are filtering of low frequency words, scaling by LogIDF, approximating term frequency by including the words context vector for the word every time it is encountered, dampening included in the weighting scheme, and optimum choice of dimensionality.

6.1 Main Contribution

The mission of the project was to examine if clustering can be performed successfully with a Random Index based representation. The overall conclusion must be that this is the case. The Random Indexing representation performs comparable to the regular clustering and the many possible parameter choices that give about the same final result indicate that there is a reasonable chance to achieve an acceptable result even without difficult fine tuning of the parameters.

6.2 Other Important Contributions

This project has revealed a few ideas that I think merit a special mentioning here.

6.2.1 Incremental LogIDF

One of the major benefits of using a method like Random Indexing is that the method is incremental. When new data is discovered it can be added to the index without making the previous partial results invalid.

Gorman and Curran (2006) report that IDF weighting is needed to make Random Indexing perform acceptably for very large corpora. This is encouraging since it extends the tasks that can be handled by Random Indexing, but also disappointing since it removes the benefit that the method is incremental.

I suggest that the possibility of using Incremental LogIDF instead of accurate statistics is very promising for the use of Random Indexing in real world situations.

6.2.2 Using Normalization When Combining Words

The idea that Normalization of the context vectors is beneficial when many words are combined is as far as I can tell a genuine discovery done in this project. A possible reason why nobody has investigated this in detail before is that normal use of Random Indexing includes comparison with the cosine measure. Since the cosine measure itself includes Normalization of the two vectors involved it is easy to view Normalization only as a part of measurement of similarity and not connected to the Random Indexing process itself.
To use scaling with Normalization as a tool to focus the representation of the major concept of each word is something that I think can have impact on other areas than clustering.

### 6.2.3 Evaluation of Random Index Quality

How to evaluate the quality of a Random Index is a difficult topic. I suggest that the idea to monitor the performance of the random index by evaluating the absolute value, sign and frequency of words that are expected to get zero in similarity is a powerful way to find out if the index lacks quality. Words that are unrelated will only be affected by the noise level of the index. Programming errors or bad parameter choices can be detected as greater than expected negative cosine measurements.

### 6.3 Suggested Research

In this section I present a number of experimental ideas that have either been discovered after the project plan was decided or have been excluded to keep the scope of the project reasonable. The ideas are not listed in any particular order.

#### 6.3.1 Analysis of Context Vectors

By visual inspection it is pretty simple to differentiate between the dense context vector for a function word and that of a word that is more distinguishing. Words with a few repeated contexts and such words that appear in many different contexts also seem to be possible to find by visual inspection.

It seems plausible that it would be possible to develop tools to automatically analyse to what degree a context vector is distinguishing. The imprint in the context vector of a word does not only contain evidence of the number of times the word occurs, but also an imprint of the number of different contexts. The document representation could possibly further be refined by allowing the representation building software to modify the encoding based on this information.

A possible benefit of this approach might be that parameters like stopwords, low frequency filtering, high frequency filtering and similar parameters that all depend on the domain of the texts and the language used can be replaced by the analysis of measurements of the context vector.

The way to start such experiments would be to generate a random index and compare all words to each other. Those words with a very high set of similar words, or a very high average similarity to other words, would be candidates to have their context vector analysed.

#### 6.3.2 Evaluation Against Proper Categories

The main evaluation used in this work is comparison with the newspaper section used for the articles. The problem is that this classification suffers from severe problems. Texts about events like the Al-Qaeda attack 9/11 can be found in all parts of newspapers and in essence the classification is based on the part of the paper in which the journalist is employed and not the actual contents of the texts.

The lack of quality of the categorization is something that affects both the Random Index based representation and the ordinary representation. On the other hand it could happen that one of the representations is more negatively affected from the imprecise categorization.

#### 6.3.3 Domain Specific Texts

It would be very interesting to do clustering with Random Indexing for domain specific texts. It might be possible that a newspaper contains too many different subjects for Random Indexing that by nature requires much raw data to get the needed statistical evidence to function properly. One way around this would be to use a random index generated over a larger corpora, another
Conclusions

would be to cluster a more limited domain. Clustering of a domain specific set of texts might
give a different view on how well the Random Index based clustering performs.

6.3.4 Removal of Repeated Words for Long Texts

This project failed to demonstrate any difference in clustering quality after removal of repeated
words. The motivation to perform this particular experiment was the concern that repeated
addition of a word with a random disturbance could possibly degrade the accuracy of the
similarity measurement. With this in mind it would be very interesting to perform experiments
for very long texts to see if the existence of repeated words have any impact in this case.

6.3.5 Customized Stopword Lists

If “Wednesday” is a word that aids in determining if two texts come from the same category or
just make unrelated texts similar is an open question. Intuition suggests that it might be the latter
case because a human reader cannot get any clue of the contexts of the text from this word
alone.

Investigation shows that the word “Wednesday” is not particularly common, it occurs in around
10% of the texts. Words that are obviously quite strong in terms of information, like for instance
“police”, occur more often so filtering by how often the word occurs have severe disadvantages.
When using standard information retrieval it makes sense to include every word. If the user
knows that the event happened on a “Wednesday” he can narrow down the search by including
“Wednesday” in the search query and it in other cases it does not matter.

It seem plausible that these kinds of words are downright destructive for clustering purposes
since clustering determines similarity based on all words. A customized stop word list will
probably improve results.

A personal speculation is that Random Indexing based clustering will gain more from a word
list that is customized for clustering purposes compared to regular clustering. The reason for this
thought is that for ordinary clustering the similarity increases only if the word is present, while
words with Random Indexing are described by the words around them. Words that are used in
many different unrelated contexts might in practice add much noise instead of just fractioning
the individual category into two parts.

6.3.6 Compound Splitting

About 20% of the content words in Swedish are found in the solid compounds. An example of a
solid compound would be that the Swedish words for “ill” and “caretaker” can be combined into
one solid compound that is the Swedish word for “nurse”. Rosell (2005) has shown that splitting
the solid compounds that exist in Swedish improve clustering results. This result is actually
rather fascinating since the downside of splitting the compounds is that with a bag-of-word
representation there is loss of the information that the words are strongly coupled in the text,
something that can certainly be argued to be a very strong part of the solid compound's
meaning.

With a Random Index based representation that uses a narrow window around the word, the
index will use the other part of the compound to describe the word. The information hidden in
the solid compound is put to use without a loss of the connection with the other part of the solid
compound. This could perhaps give the clustering algorithm a larger improvement than the
benefit to a representation not based on Random Indexing.

6.3.7 Using a Premade Random Index

In this project the texts have been clustered using only the words found in the actual documents.
An alternative would be to build a Random Index from a larger set of texts to give additional
statistical evidence that might aid in uncovering more latent connections between words.
Looking at the results of the experiments in this project it is probably imperative that such Random Index uses LogIDF as weighting scheme (or possibly Incremental LogIDF).
Bibliography


