Department of Computer Science
Series of Publications A
Report A-2013-1

Term Weighting in Short Documents for Document Categorization, Keyword Extraction and Query Expansion

Mika Timonen

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIII, University Main Building, on January 25th, 2013, at 12 o'clock.

University of Helsinki
Finland

Supervisor
Hannu Toivonen, University of Helsinki, Finland

Pre-examiners
Pekka Kilpeläinen, University of Eastern Finland, Finland
Gaël Dias, University of Caen Basse-Normandie, France

Opponent
Timo Honkela, Aalto University, Finland

Custos
Hannu Toivonen, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

Copyright © 2013 Mika Timonen
ISSN 1238-8645
ISBN 978-952-10-8566-6 (paperback)
ISBN 978-952-10-8567-3 (PDF)
Computing Reviews (1998) Classification: H.2.8, H.3.4, I.5.2, I.5.4
Helsinki 2013
Unigrafia

Term Weighting in Short Documents for Document Categorization, Keyword Extraction and Query Expansion

Mika Timonen

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Mika.Timonen@vtt.fi

PhD Thesis, Series of Publications A, Report A-2013-1
Helsinki, January 2013, 53 + 62 pages
ISSN 1238-8645
ISBN 978-952-10-8566-6 (paperback)
ISBN 978-952-10-8567-3 (PDF)

Abstract

This thesis focuses on term weighting in short documents. I propose weighting approaches for assessing the importance of terms for three tasks: (1) document categorization, which aims to classify documents such as tweets into categories, (2) keyword extraction, which aims to identify and extract the most important words of a document, and (3) keyword association modeling, which aims to identify links between keywords and use them for query expansion.

As the focus of text mining is shifting toward datasets that hold user-generated content, for example, social media, the type of data used in the text mining research is changing. The main characteristic of this data is its shortness. For example, a user status update usually contains less than 20 words.

When using short documents, the biggest challenge in term weighting comes from the fact that most words of a document occur only once within the document. This is called hapax legomena and we call it *Term Frequency = 1*, or *TF=1* challenge. As many traditional feature weighting approaches, such as Term Frequency - Inverse Document Frequency, are based on the occurrence frequency of each word within a document, these approaches do not perform well with short documents.

The first contribution of this thesis is a term weighting approach for document categorization. This approach is directed to combat the TF=1 challenge by excluding the traditional term frequency from the weighting method. It is replaced by using word distribution among categories and within a single category as the main components.

The second contribution of this thesis is a keyword extraction approach that uses three levels of word evaluation: corpus level, cluster level, and document level. I propose novel weighting approaches for all of these levels. This approach is designed to be used with short documents.

Finally, the third contribution of this thesis is an approach for keyword association weighting that is used for query expansion. This approach uses keyword co-occurrences as the main component and creates an association network that aims to identify strong links between the keywords.

The main finding of this study is that the existing term weighting approaches have trouble performing well with short documents. The novel algorithms proposed in this thesis produce promising results both for the keyword extraction and for the text categorization. In addition, when using keyword weighting with query expansion, we show that we are able to produce better search results especially when the original search terms would not produce any results.

Computing Reviews (1998) Categories and Subject Descriptors:

H.2.8	Database management: Data mining
H.3.3	Information storage and retrieval: Query formulation
I.5.2	Pattern Recognition: Feature evaluation and selection
I.5.4	Pattern Recognition: Text processing

General Terms:
Algorithms, Experimentation

Additional Key Words and Phrases:
Keyword Extraction, Query Expansion, Term Weighting, Text Classification, Text Mining

Acknowledgements

This work has been carried out in several projects at VTT Technical Research Centre of Finland. Some of the projects were funded by the Finnish Funding Agency for Technology and Innovation (TEKES), some by VTT and some by Taloustutkimus Oy. The single biggest reason that made this thesis possible is my trip to East China Normal University in Shanghai where I spent six months as a visiting researcher. During that time I was able to fully concentrate on my work and write publications. I wish to thank everyone who made that trip possible.

During my work I have had two supervisors from University of Helsinki: Dr. Roman Yangarber and Prof. Hannu Toivonen. Roman helped me to get this process started and Hannu helped me to get this process finished. I am thankful for their help. In particular, I wish to thank Hannu for the great feedback he gave me regarding the papers and this thesis. He helped me to greatly improve my work and to connect the dots.

In addition, I would not have been able to complete this thesis without the PhD writing leave that VTT generously offers, the people (especially professor Liang He) at ECNU, and my fiancée Melissa who has helped me to brainstorm most of the things described in this thesis.

My gratitude to all my co-authors, Melissa Kasari, Paula Silvonen, Timo Toivanen, Yue Teng, Chao Cheng and Liang He. In addition, I would like to thank my managers and co-workers (in no particular order), Eero Punkka, Lauri Seitsonen, Olli Saarela, Antti Pesonen and Renne Tergujeff for their contributions and support. I would also like to thank Markus Tallgren for believing in me and employing me to VTT. He also helped me to get my work started. Jussi Ahola was a great help in my early research when I was starting to write the papers that eventually lead to this thesis. Without the help and support I received from Jussi this thesis may not have ever been written.

I express my deep gratitude to the pre-examiners of this thesis, Prof. Pekka Kilpeläinen and Prof. Gaël Dias, for their time and feedback that helped improve this dissertation.

Finally, I would like to thank my friend Christian Webb for his help and support especially at the beginning of my studies. He made my life easier when I was new in town and just starting out my studies.

I dedicate my thesis to my family, which includes my fiancée, my parents and my brother who have all supported me tremendously during this project. Without the safety net they offered this process would have been nearly impossible. Their continuous support was the key ingredient that kept me striving toward this goal.

Contents

List of Publications and the Author's Contributions

This thesis consists of four peer-reviewed articles. Three of the articles have been published in refereed proceedings of data mining conferences and one in a refereed information modeling journal. Three of the articles focus on term weighting in two different domains: text categorization, and keyword extraction (Article I, II, III). Article IV focuses on utilization of the keywords in information modeling. My contribution to all of these papers is substantial as I was the main author of each of these papers. These articles have not been included in any other thesis.

Article I

Classification of Short Documents to Categorize Consumer Opinions, Mika Timonen, Paula Silvonen, Melissa Kasari, In *Online Proceedings of 7th International Conference on Advanced Data Mining and Applications*, ADMA 2011, Beijing, China, 2011, pages 1 - 14. Available at `http://aminer.org/PDF/adma2011/session3D/adma11_conf_32.pdf`.

For Article I, I designed and implemented the feature weighting approach and ran the experiments. The contribution of other authors was mainly in experimental setup, writing some small parts of the paper and proof reading.

Article II

Categorization of Very Short Documents, Mika Timonen, In *Proceedings of 4th International Conference on Knowledge Discovery and Information Retrieval*, Barcelona, Spain, 2012, pages 5 - 16.

For Article II, I was the only contributor in this paper.

Article III

Informativeness-based Keyword Extraction from Short Documents, Mika Timonen, Timo Toivanen, Yue Teng, Chao Chen, Liang He, In *Proceedings of 4th International Conference on Knowledge Discovery and Information Retrieval*, Barcelona, Spain, 2012, pages 411 - 421.

For Article III, I designed and implemented the keyword extraction approach. The contribution of other authors was in running the experiments, writing small parts of the paper and creating the test sets.

Article IV

Modelling a Query Space using Associations, Mika Timonen, Paula Silvonen, Melissa Kasari, In *Frontiers in Artificial Intelligence and Applications: Information Modelling and Knowledge Bases XXII*, Volume 225, 2011, pages 77-96.

For Article IV, I designed and implemented the approach for keyword weighting and information modeling with Melissa Kasari. I was the main author of the paper; other authors wrote small parts of the paper.

Chapter 1

Introduction

In this thesis I propose approaches for term weighting in short documents. I focus on three text mining tasks: text categorization, keyword extraction and query expansion. I aim to identify and tackle the challenges of short documents and compare the performance of the proposed approaches against a wide range of existing methods.

Text mining is a process that aims to find and refine information from text. It is a well research field; for instance, during the 1990's and early 2000 text categorization received a lot of attention due to its relevance to both information retrieval and machine learning [43]. News article categorization was one of the focus areas as it provided a large set of data and a standard testing environment [25, 50, 51].

However, with the rise of user-created content on the Internet, the type of interesting texts to be analyzed has changed significantly. As the focus of text mining has been shifting toward Twitter messages, product descriptions and blogs, the traditional datasets, such as Reuters-21578 [26], are no longer as relevant as they once were in the text mining research. When compared to classic text mining datasets, user-generated content has one major difference: its length. For instance, an average Reuters news article holds 160 words [44] but a tweet (Twitter message) contains at most 140 characters; i.e., around 20 words.

In addition to tweets, another relevant source of short documents is market research data collected using surveys and questionnaires that contain both bounded and open ended questions. Bounded questions have a limited set of possible answers while open questions can be answered freely by writing what ever the respondent feels fit. Manual categorization of this data is laborious as a single survey is often answered by thousands of respondents. This brings a need for automatic categorization of the answers.

Short documents are relevant also in information extraction. Product, event, movie and company descriptions are all often short and they contain information that can be relevant in several fields. User modeling, for example, can take the extracted information and use it to build models that indicate the user's interest.

In order to utilize the information from short documents, whether we want to categorize the text or extract information from it, we need to identify which words are the most important within the text. This can be achieved with term weighting.

There are several approaches to term weighting of which the *Term Frequency - Inverse Document Frequency* [42] (*TF-IDF*) is probably the most often used. It is an approach that relies heavily on *term frequency* (*TF*); i.e., a statistic of how many times a word appears within a document. When using TF-IDF, words that occur both only in a few documents within the corpus, and that occur often within a single document, are emphasized. In many cases, TF is a good statistic to measure the importance of a word: if it occurs often, it could be important.

This approach does not work well with short documents. When a document contains only a few words, there are seldom words that occur more often than once within a document. *Hapax legomenon* is a word that occurs only once within a context. In our work, we call this *Term Frequency=1 challenge* or *TF=1 challenge*. As many of the traditional approaches are based on TF, we need to find new ways to weight the terms within the short documents.

The main research challenge I focus on in this thesis is related to the *TF=1* challenge: *How to efficiently weight the terms in short documents?* As term weighting is used as part of other text mining applications, I focus on three separate cases: document categorization, keyword extraction and keyword association modeling.

This thesis is organized as follows. In Chapter 2, I give a detailed description of the research questions discussed in this thesis. In Chapter 3, I summarize the background and the related approaches. In Chapter 4, I answer the research questions by proposing three term weighting approaches to combat the TF=1 challenge in the three text mining fields. In Chapter 5, I present the experimental results and the utilization of the weighted terms. Chapter 6 summarizes the contributions of this thesis. I conclude this thesis in Chapter 7 with discussion of my work.

Chapter 2

Research questions

My work on short documents has focused on several domains and datasets. In this section I describe the research questions we formulated and their background.

My work with short documents started with text categorization. A Finnish market research company had a lot of data that consisted of short documents from their old surveys. As manual categorization of this data has been an arduous task, they wanted to find out if there was a way to automatically process and categorize the survey data. After the first tries we realized that in order to produce good categorization results, we needed to weight the terms efficiently. Due to the length of the documents, the existing approaches, mainly TF-IDF and other term frequency based approaches, did not perform well. From this, we formed the first research question:

1. How to weight terms in short documents for document categorization to overcome TF=1 challenge?

As there are usually no words that occur more than once per document, we need to use an approach that does not rely on term frequency within the document. In addition to term weighting, a smaller challenge is to find a good classifier that is precise and can classify a high percentage of documents. Both of these questions and our contributions are discussed in both Article I and Article II.

In addition to market research data, we had a project that concentrated on other types of short documents: product and event descriptions. We used them to build a recommendation system where the aim was to recommend events such as rock concerts, sporting events and exhibitions. In order to implement a tag-based recommendation system we wanted to tag each event using the keywords found from the descriptions. This motivated the second research question:

2. How to extract the most informative words from a short document?

This challenge focuses on keyword extraction from short documents. As the extracted keywords are used as tags in a recommendation system we need to extract different types of words; some of which are rare and some are common. Therefore, the term weighting approach developed earlier in the context of Question 1 cannot be used alone but we need to find some other complementary methods as well. This is addressed in Article III.

In an ideal case keywords form a precise summary of the document. This information can be used in several ways. One of the applications we have been tackling is a search engine for company's internal documents. However, due to the small number of documents to be queried, sometimes the search produced poor or even no results. In this project we had a set of keywords for each document we wanted to search, so we decided to use them to see if they could be utilized to alleviate this problem. Therefore, one research challenge in the project focused on weighting keywords and keyword pairs and finding a way to utilize them in the search engine. From this, we get the third research question:

3. How to weight keywords and utilize them in a search engine?

This question focuses on using the keywords for search space modeling in a limited sized search engine. The aim is to alleviate the challenge of document retrieval in small search engines such as intranet where the search often produces poor results due to the limited number of documents in the search space. We use an association network to model the associations between the keywords found from the documents and utilize the network in a query expansion method. When building the network, the keywords are weighted to indicate the strength of the association. This is described in Article IV.

Chapter 3

Background

In this chapter I describe the background of our work by giving an overview of the most relevant existing and related approaches on term weighting, document categorization, keyword extraction and query expansion.

3.1 Term weighting

Term weighting, also known as feature weighting when it is not used with text documents, is a method for assessing the importance of each term in the document. A term can be a word or a set of words such as a noun phrase (for example, a proper name). Intuitively, we would like to diminish the impact of terms that are not important in the document (e.g., common verbs, prepositions, articles) and emphasize the impact of others. If this is not done, words like "is", "the", and "in" would have a similar impact with such words as "Olympics", "concert" and "London". When terms are used, for example, in categorization, this would weaken the performance of the classifier considerably. In addition, by removing the unimportant terms the task becomes computationally less demanding [52].

Term Frequency - Inverse Document Frequency (*TF-IDF*) [42] is the most traditional term weighting method and it is used, for example, in information retrieval. The idea is to find the most important terms for the document within a corpus by assessing how often the term occurs within the document (*TF*) and how often in other documents (*IDF*):

$$\text{TF-IDF}(t,d) = -\log \frac{\mathit{df}(t)}{N} \times \frac{\mathit{tf}(t,d)}{|d|}, \tag{3.1}$$

where $\mathit{tf}(t,d)$ is the term frequency of word t within the document d (how often the word appears within the document), $|d|$ is the number of words

Table 3.1: Notations used in this section.

Notation	Meaning	Notation	Meaning
t	Term	$\neg t$	No occurrence of $\neg t$
d	Document	c	Category
$\mathit{df}(t)$	Number of documents with at least one occurrence of t	$\mathit{tf}(t,d)$	Number of times t occurs within a document
$\mathit{ctf}(t)$	Collection term frequency	c_t	Categories that contain t
d_t	Documents that contain t	N	Total number of documents in the collection
$N_{t,c}$	Number of times t occurs in c	$N_{t,\neg c}$	Number of occurrences of t in other categories than c
$N_{\neg t,c}$	Number of occurrences of c without t	$N_{\neg t,\neg c}$	Number of occurrences with neither t or c
N_t	Number of occurrences of t		

in the document, $\mathit{df}(t)$ is the document frequency within the corpus (in how many different documents the word appears in), and N is the number of documents in the corpus. TF-IDF emphasizes words that occur often within a single document and rarely in other documents. Table 3.1 shows the notations used in the equations in this section.

Rennie and Jaakkola [40] have surveyed several other approaches and their use for named entity recognition. In their experiments, *Residual IDF* [9] produced the best results. Residual IDF is based on the idea of comparing the word's observed IDF against predicted IDF ($\widehat{IDF}$). Predicted IDF is estimated using the term frequency and assuming a random distribution of the term in the documents. The larger the difference between IDF and $\widehat{IDF}$, the more informative the word. Equation 3.2 presents how the residual IDF ($RIDF$) is estimated using observed IDF and predicted IDF:

$$\begin{aligned} RIDF(t) &= IDF(t) - \widehat{IDF}(t) \\ &= -\log\frac{\mathit{df}(t)}{N} + \log\left(1 - e^{-\frac{\mathit{ctf}(t)}{N}}\right), \end{aligned} \tag{3.2}$$

where *ctf*(t) is the collection term frequency; *ctf*$(t) = \sum_d$ *tf*(t, d). This approach is similar to TF-IDF as the score will be higher when a word occurs often in a single document. However, this approach tends to give words with medium frequency the highest weight.

Other approaches experimented by Rennie and Jaakkola included x^I metric introduced by Bookstein and Swanson [7]:

$$x^I(t) = N_t - df(t), \tag{3.3}$$

where N_t is the total number of occurrences of t, and $df(t)$ is the number of documents where t appears in. However, according to Rennie and Jaakkola, x^I is not an effective way to find informative words.

Odds Ratio (*OR*), *(Pointwise) Mutual Information* (*MI*), *Information Gain* (*IG*), and *Chi-squared* (χ^2) are other often used approaches. Odds Ratio ($OR(t)$) is used, for example, for relevance ranking in information retrieval [30]. It is calculated by taking the ratio of positive samples and negative samples; i.e., the odds of having a positive instance of the word when compared to the negative [14]:

$$OR(t) = \log \frac{N_{t,c} \times N_{\neg t,\neg c}}{N_{t,\neg c} \times N_{\neg t,c}}, \tag{3.4}$$

where $N_{t,c}$ denotes the number of times term t occurs in category c, $N_{t,\neg c}$ is the number of times t occurs in other categories than c, $N_{\neg t,c}$ is the number of times c occurs without term t, $N_{\neg t,\neg c}$ is the number of times neither c nor t occurs.

Information Gain ($IG(t)$) is often used by decision tree induction algorithms, such as C4.5, to assess which branches can be pruned. It measures the change in entropy when the feature is given, as opposed of being absent. This is estimated as the difference in observed entropy $H(C)$ and the expected entropy $E_T(H(C|T))$ [52]:

$$\begin{aligned} IG(t) &= H(C) - E_T(H(C|T)) \\ &= H(C) - (P(t) \times H(C|t) + P(\neg t) \times H(C|\neg t)) \\ &= -\sum_{i=1}^{m} P(c_i) \log P(c_i) \\ &+ P(t) \sum_{i=1}^{m} P(c_i|t) \log P(c_i|t) \\ &+ P(\neg t) \sum_{i=1}^{m} P(c_i|\neg t) \log P(c_i|\neg t), \end{aligned} \tag{3.5}$$

where $\neg t$ indicates the absence of t, m is the number of all categories, c_i is the ith category.

Chi-squared ($\chi^2(t,c)$) is a traditional statistical test of independence. In feature weighting it is used to assess the dependency of the feature – category, or feature – feature pairs [52]:

$$\chi^2(t,c) = \frac{N \times (A \times D - C \times B)^2}{(A+C) \times (A+B) \times (B+D) \times (C+D)}, \tag{3.6}$$

where $A = N_{t,c}$, $B = N_{t,\neg c}$, $C = N_{\neg t,c}$, and $D = N_{\neg t,\neg c}$. If the χ^2 score is large, the tested events are unlikely to be independent which indicates that the feature is important for the category.

Pointwise Mutual Information ($MI(t,c)$) is similar to the Chi-squared feature selection. The idea is to score each feature - category (or feature - feature) pair and see how much a feature contributes to the pair:

$$MI(t,c) = \log_2 \frac{N_{t,c} \times N}{(N_{t,c} + N_{\neg t,c}) \times (N_{t,c} + N_{t,\neg c})}. \tag{3.7}$$

Bi-Normal Separation (BNS) is an approach originally proposed by Forman [14]. The approach uses the standard normal distribution's inverse cumulative probability functions of positive examples and negative examples:

$$BNS(t,c) = |F^{-1}(\frac{N_{t,c}}{N_{t,c} + N_{t,\neg c}}) - F^{-1}(\frac{N_{\neg t,c}}{N_{\neg t,c} + N_{\neg t,\neg c}})|, \tag{3.8}$$

where F^{-1} is the inverse Normal cumulative distribution function. As the inverse Normal would be infinite at 0 and 1, Forman limited both distributions to the range [0.0005,0.9995].

The idea of BNS is to compare the two distributions; the larger the difference between them, more important the feature. In other words, when a feature occurs often in the positive samples and seldom in negative ones, the feature will get a high BNS weight. In his later work, Forman compared the performance of BNS against several other feature weighting approaches including Odds Ratio, Information Gain, and χ^2 [15]. In his experiments BNS produced the best results with IG performing the second best.

3.2 Document categorization

I now give the background for document categorization, a classification task focusing on text documents.

3.2.1 Classification

Classification is a machine learning task where the aim is to build a model that predicts labels for feature vectors. Text categorization is a classification task where the feature vectors are formed from text documents. Classification can be divided into three steps: 1) feature vector creation, 2) training a classifier, and 3) label prediction. A classifier is trained using a set of vectors called training set.

In the first step, each document forms a feature vector by mapping each term in the document into a feature. When creating a feature vector the document is preprocessed depending on the requirements of the classification task. This may include, for example, stemming and stop word removal. After preprocessing, the features in the feature vectors are weighted so that the most important features are emphasized and less important are removed or their impact diminished.

In the second step the training process takes the set of feature vectors with their labels as its input and outputs the classifier. The classifier is usually a function to predict the classes of the feature vectors. The actual type of the function differs based on the selected classification method. The performance of the classifier is experimented with a test set where the labels are known in advance. However, the classifier does not know these labels. The result of the classification is analyzed and the classifier may be re-built if the results are not satisfactory.

Finally, when the classifier produces satisfying results, it can be used to predict the labels of unlabeled data. This step takes the classifier and the unlabeled data as its input and predicts the category for each of the unlabeled vectors. Output of the process is the classes for each of the vectors.

The process and especially the models differ based on the selected classification approach. A *Naive Bayes classifier* uses a probabilistic function. The idea is to assess the probabilities of each category, and of each feature for each category, and to use the probabilities in classification [39]. Consider the following simplified example, where we have two categories: *car* and *truck*. When there is a document about *cars*, the word *sedan* occurs 80 % of times and the word *trailer* occurs 5 % of times. When a document talks about *trucks*, the word *sedan* occurs 1 % of times and the word *trailer* occurs 70 % of times. For the sake of simplicity, assume that both categories have the same number of documents: $P(car) = P(truck)$.

If an unlabeled document contains the word *sedan*, it is more likely to belong to the category *car* as it is more probable to find this feature value from the category *car* than from *truck*:

$P(car|sedan) = \frac{P(car \wedge sedan)}{P(sedan)} > \frac{P(truck \wedge sedan)}{P(sedan)} = P(truck|sedan)$.

A *k-Nearest Neighbor classifier* (*kNN*) is based on the idea of finding the k nearest training vectors for the test vector and using the categories from those k vectors as the label(s) for the test vector. The distance between the test vector and the training vector can be calculated several ways. It can be the number of matching features (e.g., words), Euclidean distance between the feature vectors or a cosine similarity between the feature vectors. The label can be selected from the closest k neighbors using, for example, weighted voting where each training document gets number of votes that is dependent on the similarity between the two vectors. Other options include using the label that occurs most often among the k neighbors, or using all of the labels among the neighbors. Yang [50] experimented with several statistical classifiers and concluded that kNN produced the best results with their test set of Reuters news articles.

Support Vector Machine classifier (*SVM*) [10] takes the training vectors and aims to find a hyperplane that separates positive and negative samples into different sides of the hyperplane. Usually it is impossible to separate the samples directly using the given data in the given dimensions. For this reason, it is often a good idea to map the original space into a higher-dimensional space where the separation is easier to accomplish [37]. SVM classifiers use a kernel function that maps the features into higher dimensions and creates the hyperplane; this is the model created by the SVM classifier.

3.2.2 Related applications

Yang has compared several of the approaches using Reuters news article data [50, 51]. In these experiments k-Nearest Neighbors (kNN) was one of the top performers. In several other studies Support Vector Machine (SVM) has been reported to produce the best results [22, 25, 51].

Naive Bayes classification has also been able to produce good results. Rennie et al. [39] describe Transformed Weight-normalized Complement Naive Bayes (TWCNB) approach that can, according to them, produce comparable results with SVM. They base the term weighting mostly on term frequency but they also assess term's importance by comparing its distribution among categories. Kibriya et al. [24] extended this idea by using TF-IDF instead of TF in their work.

In text categorization, most research has used text documents of normal length, such as news articles, but there are a few instances that use short documents such as tweets. Pak and Paroubek [35], for example, use linguistic analysis to mine opinions and conclude that when using part-of-speech tagging it is possible to find strong indicators for emotion in text.

Spam detection from tweets is also a popular topic. For example, Moh and Murmann [31] describe an approach for spam detection from tweets that is mostly based on the links between users. They include several features that are related to the user statistics in Twitter. Their approach requires very little analysis of the actual tweets. Benevenuto et al. [4] describe an approach where they use several features from tweets, such as content and user behavior, to classify tweets as spam. These features include the contents of the tweets. McCord and Chuah [28] evaluate the use of several traditional classifiers for spam detection in Twitter. They use user-based and content-based features. User-based features include number of friends (which are known as following and followers in Twitter) and distribution of tweets within a given time period. Content-based features include URLs, keywords and word weights, and hashtags. McCord and Chuah use SVM, Naive Bayes, kNN and Random Forest in their experiments. Random Forest produced the best results in their experiments, with kNN and SVM producing the next best results. Naive Bayes produced clearly the worst results.

Garcia Esparza et al. [13] aim to categorize and recommend tags to tweets and other short messages in order to combat the different tagging conventions of users and to facilitate search. They use TF-IDF term weighting and a kNN classifier with $k = 1$. Ritter et al. [41] describe an approach for modeling Twitter conversations by identifying dialog acts from tweets. Dialogue acts provide shallow understanding of the type of text in question; for example, the text can be identified as being *statement*, *question* or *answer*. They use a Hidden Markov Model to create conversation models and Latent Dirichlet Allocation (LDA) [6] to find hidden topics from the text. The topics are used to follow the dialogue and predict when the topic changes; this helps to identify the different dialogue acts.

Spam detection in tweets is a interesting topic but not directly applicable to categorization of text into several categories. Of the work we have reviewed, only the work done by Esparza et al. [13] uses term weighting and focuses on document categorization. However, they use TF-IDF and kNN which we will show to be ineffective in Article I and Article II.

3.3 Keyword extraction

Several authors have presented keyword extraction approaches in recent years. The methods often use supervised learning. In these cases the idea is to use a predefined seed set as a training set and learn the features for keywords. The training set is built manually by tagging the documents with keywords.

An example that uses supervised learning is called Kea [16, 49]. It uses Naive Bayes learning with TF-IDF and normalized term positions, i.e., the first occurrence of the word divided by the number of words in the text, as the features. The approach was further developed by Turney [47] who included keyphrase cohesion as a new feature. One of the latest updates to Kea is by Nguyen and Kan [32] who included linguistic information such as section information as features.

Before developing the Kea approach, Turney experimented with two other approaches: decision tree algorithm C4.5 and an algorithm called GenEx [46]. GenEx has two components: a hybrid genetic algorithm Genitor, and Extractor. The latter is the keyword extractor that needs twelve parameters to be tuned. Genitor is used for finding these optimal parameters from the training data.

Hulth et al. [21] describe a supervised approach that utilizes domain knowledge found from Thesaurus, and TF-IDF statistics. Later, Hulth included linguistic knowledge and different models to improve the performance of the extraction process [19, 20]. The models use four different attributes: term frequency, collection frequency, relative position of the first occurrence, and part-of-speech tags.

Ercan and Cicekli [12] describe a supervised learning approach that uses lexical chains for extraction. The idea is to find semantically similar terms, i.e., lexical chains, from text and utilize them for keyword extraction as semantic features.

There are also approaches that do not use supervised learning but rely on term statistics instead. KeyGraph is an approach described by Ohsawa et al. [33] that does not use part-of-speech tags, large corpus, nor supervised learning. It is based on term co-occurrence, graph segmentation and clustering. The idea is to find important clusters from a document and assume that each cluster holds keywords. Matsuo and Ishizuka [27] describe an approach that uses a single document as its corpus. The idea is to use the co-occurrences of frequent terms to evaluate if a candidate keyword is important for a document. The evaluation is done using Chi-squared (χ^2) measure. All of these approaches are designed for longer documents and they rely on term frequencies.

Mihalcea and Tarau [29] describe an unsupervised learning approach called TextRank. It is based on PageRank [34] which is a graph-based ranking algorithm. The idea is to create a network where the vertices are the terms of the document and edges are links between co-occurring terms.

A term pair is co-occurring if they are within 2 to 10 words within each other in the document. The edges hold a weight that is received using the PageRank algorithm. The edges are undirected and symmetric. The keywords are extracted by ranking the vertices and picking the top n ones. This approach produced improved results over the approach described by Hulth.

There are some approaches developed that extract keywords from abstracts. These abstracts often contain 200-400 words making them considerably longer than documents in our corpus. One such approach, proposed by HaCohen-Kerner [17], only uses term frequencies to extract keywords. Andrade and Valencia [2] use Medline abstracts to extract protein functions and other biological keywords. The previously mentioned work by Ercan and Cicekli [12] also uses abstracts as the corpus.

Wan and Xiao [48] describe an unsupervised approach called CollabRank that clusters the documents and extracts the keywords within each cluster. The assumption is that documents with similar topics contain similar keywords. The keyword extraction has two levels: first, the words are evaluated in the cluster level using a graph-based ranking algorithm similar to PageRank [34]. After this, the words and phrases are scored at the document level by summing the cluster level saliency scores. In the cluster level evaluation, part-of-speech tags are used to identify suitable candidate keywords. The part-of-speech tags are also used when assessing if the candidate keyphrases are suitable. Wan and Xiao use news articles as their corpus.

3.4 Query expansion

Query expansion is a process that aims to reformulate a query to improve the results of information retrieval. This is important especially when the original query is short or ambiguous and would therefore give only irrelevant results. By expanding the query with related terms the reformulated query may produce good results.

Carpineto and Romano [8] has surveyed query expansion techniques. According to them, the standard methods include: semantic expansion, word stemming and error correction, clustering, search log analysis and web data utilization.

In semantic expansion, the idea is to include semantically similar terms to the query. These words include synonyms and hyponyms. When using word stemming, the idea is to use a stemmed version of the word so that different types of spellings can be found (e.g., singular and plural). Term clustering is a way to find similar terms by using term co-occurrence. Search log analysis is another way of finding similar terms. In this case, the logs are analyzed to identify terms that often co-occur with the given query terms. Finally, web data utilization is an approach where an external data source (e.g., Wikipedia[1]) is used for query expansion. The idea here is to use hyperlinks in Wikipedia to find related topics for the query terms.

Bhogal et al. [5] also reviewed query expansion approaches. They mainly focus on three areas in their review: relevance feedback, corpus dependent knowledge models and corpus independent models. Relevance feedback is one of the oldest methods for expansion. It expands the query using terms from relevant documents. The documents are assessed as relevant if they are ranked highly in previous queries or identified as relevant in other ways (e.g., manually). Corpus dependent knowledge models take a set of documents from the domain and uses them to model the characteristics of the corpus. This includes the previously mentioned stemming and co-occurrence approaches. Corpus independent knowledge models includes semantic expansion and the web data utilization as it uses dictionaries such as WordNet[2] to include synonyms and hyponyms into the search. For more information, we refer the reader to the original articles by Carpineto and Romano [8] and Bhogal et al. [5].

In our work, we focus on term co-occurrence. We use an approach called association network to model the links between the terms. When expanding a query, we need to search the network for the associative terms. For this, we use an idea from spreading activation [11]. Spreading activation is a technique to search a network by starting from a node and iteratively traveling to the neighboring nodes using a predefined condition and the weights between the nodes.

The actual implementation of the spreading activation technique can be done using a best first search approach [36] which is a graph search algorithm that expands the most promising node. The node is chosen according to a specified rule. The idea is to start from the node that maps to the query term and add new nodes (i.e., terms depicted by nodes) to the query using a function $f(t)$. The function selects the best nodes to the query using the weight between the query node and the other nodes. The actual function will depend on the type of graph and the weights. Top n nodes are added to the query where n is a predefined number or the number of nodes that fulfill the function. Our implementation of this query expansion approach is described in Section 4.3.

[1] `http://www.wikipedia.org/`

[2] `http://wordnet.princeton.edu/`

Chapter 4

Term weighting in short documents

In this chapter, I describe our work on term weighting in short documents and propose three novel approaches. Here, a document is considered short when it has at most 100 words. However, most documents used in our studies can be very short, i.e., contain less than 20 words. Twitter messages and market research data are examples of this.

The approaches we have used for term weighting differ depending on the task at hand. The tasks we have studied are document categorization, keyword extraction, and keyword association modeling.

For document categorization we propose two different term weighting approaches that were developed to be used with two different classifiers: a Naive Bayes classifier (Article I) and a Support Vector Machine classifier (Article II).

For keyword extraction we propose an approach that weights the terms on three levels: corpus level, cluster level and document level (Article III). Finally, for keyword association modeling we propose a weighting approach that uses keyword co-occurrence (Article IV). Here the aim is to find strong links between keywords and use them for query expansion.

4.1 Document categorization

Document categorization is one of the main areas where term weighting has been used. The aim is to assess the importance of each word and emphasize the more important words over the unimportant ones. When using only the more important words, the classifier will usually require less computational power than when using all the words.

4.1.1 Approach I: Two level relevance values

The approach presented in this section was introduced in Article I. It uses four components that may not be novel by themselves but by combining them we get a novel feature weighting method. This approach is loosely based on the work by Rennie et al. [39] called Transformed Weight-normalized Complement Naive Bayes (TWCNB).

TWCNB uses Naive Bayes classifier and weights features by: 1) Term Frequency, 2) Inverse Document Frequency, 3) Length Normalization, and 4) Complement class weighting. The first three are standard weighting methods used for example in TF-IDF weighting. The fourth is an approach that uses the distribution among categories. That is, it compares the difference in frequency among positive examples and negative examples. We use this idea of distribution comparison in our work. In addition, we use a component that is similar with Inverse Document Frequency. The other two (1 and 3) are not used in our work.

The aim of our approach is to assess the information value of each word by estimating its relevance in the corpus level and category level. We will define the following statistics to calculate the weights: *inverse average fragment length* where the word appears in, *category probability of the word*, *document probability within the category*, and *inverse category count*. The hypothesis is that as these do not rely on term frequency within a document these are better for weighting the terms within short documents. More detailed descriptions of these components and of the equation for combining them can be found in Article I, pages 7 – 11.

Inverse average fragment length

Inverse average fragment length is based on the assumption that a word is informative when it can occur alone. That is, we assume that on average there are fewer surrounding words around the informative words than uninformative ones.

A fragment is a part of the text that is broken from the document using predefined breaks. We break the text into fragments using predefined stop words and break characters. We use the following stop words: and, or, both. We use the corresponding translated words when the text is not in English. In addition, we use the following characters to break the text: comma (,), exclamation mark (!), question mark (?), full stop (.), colon (;), and semicolon (;). For example, sentence "*The car is new, shiny and pretty*" is broken into fragments "*The car is new*", "*shiny*", "*pretty*".

As an example of our assumption, consider the previous example. Words *shiny* and *pretty* are alone where as words *the*, *car*, *is*, and *new* have several other words in the same fragment. As the words *new*, *shiny*, and *pretty* form a list, they can appear in any order (i.e., in any fragment) whereas the words *the*, *car*, and *is* may not. When we have several documents about the same topic (e.g., car reviews), the same words can occur often. By taking the average fragment length for each word, three words (new, shiny, pretty) will stand out from the less important ones (the, car, is).

The inverse average fragment length $\mathit{ifl}(t)$ for the word t is calculated as follows:

$$\mathit{ifl}(t) = \frac{1}{\frac{1}{|f_t|}\sum l_f(t)}, \tag{4.1}$$

where f_t is the collection of fragments where the word t occurs in, and l_f is the length of the fth fragment where the word t occurs in. In other words, we take the average length of the fragments the word t occurs in. If the word occurs always alone, $\mathit{ifl}(t) = 1$.

If the example sentence occurs two additional times in the form ”*The car is shiny, pretty and new*”, the word *shiny* would have occurred alone once, word *new* two times, and the word *pretty* three times. The unimportant words of this example (the, car, is) occur with three other words in every instance making their inverse average fragment length smaller. In this example the inverse average fragment lengths for each of the words are: $\mathit{ifl}(\mathit{car}) = 0.25$, $\mathit{ifl}(\mathit{is}) = 0.25$, $\mathit{ifl}(\mathit{new}) = 0.5$, $\mathit{ifl}(\mathit{shiny}) = 0.33$, and $\mathit{ifl}(\mathit{pretty}) = 1.0$. As can be seen, this approach gives emphasis on the words that often occur alone.

Inverse category count

Another component of feature weighting for assessing the words in the corpus level is inverse category count. Here, the idea is to emphasize words that occur in fewer categories. That is, the fewer categories include the word, the more informative it is. Inverse category count is defined as:

$$\mathit{icc}(t) = \frac{1}{c_t}, \tag{4.2}$$

where c_t is number of categories where word t occurs in.

If a word appears in a single category, its inverse category count is 1.0 and if it appears in more categories, its inverse category count approaches 0.0 quite quickly. However, this is dependent on the classification task as, e.g., in binary classification the discrimation power of this approach is not strong.

Category probability

The probability of finding the word within a category is the key component of our weighting approach. It is based on the idea that the words that occur often in a single category and rarely in others are the most important ones.

Category probability $P(c|d)$ uses the distribution of the word among the categories. If the word occurs only in a single category, the corresponding category probability is 1. The probability of other categories is 0. This indicates the word's importance for the given category.

The conditional probability $P(c|d)$ is estimated simply by taking the number of documents in the category that contain the word t ($|\{d : t \in d, d \in c\}|$) and dividing it by the total number of documents that contain word t ($|\{d : t \in d\}|$):

$$P(c|d) = \frac{|\{d : t \in d, d \in c\}|}{|\{d : t \in d\}|} = \frac{N_{d,c}}{N_{d,c} + N_{d,\neg c}}. \tag{4.3}$$

Here we use similar notations with Table 3.1, but instead of term counts (e.g., $N_{t,c}$) we use document counts: $N_{d,c}$ which is the number of times document d with word t occurs within category c, and $N_{d,\neg c}$ which is the number of times the document d with word t occurs in other categories than c. This equation corresponds Equation 4 in Article I.

Document probability

The probability that a document in category c contains word t is the final component of the weight:

$$P(d|c) = \frac{|\{d : t \in d, d \in c\}|}{|\{d : d \in c\}|} = \frac{N_{d,c}}{N_{d,c} + N_{\neg d,c}}, \tag{4.4}$$

where $N_{d,c}$ is the number of times document d with word t occurs within the category c, and $N_{\neg d,c}$ is the number of documents without the word t that occur in the category c. This equation corresponds Equation 5 in Article I.

The intuition with this component is that a word is important if it occurs often within the category and unimportant if it occurs seldom. However, if this approach would be used alone it would not find the important words as it would emphasize words that occur often. These words include common verbs ("is"), prepositions ("through"), and articles ("the"). But by combining this probability with category probability, the combination emphasizes words that occur in few categories often. Common verbs, prepositions and articles occur often in several categories which will make their weight small.

Feature weight I

The weight is calculated for each category - word pair separately using the factors described above. I.e., if the word occurs in two different categories its weight is in general different in both of those categories. The weight $w(t, c)$ for word t and category c is the combination of the four component described previously:

$$w(t, c) = (\mathit{ifl}(t) + \mathit{icc}(t)) \times (P(c|d) + P(d|c)). \tag{4.5}$$

The weight has two parts: corpus level, which consists of the average fragment length and inverse category count, and category level, which consists of the two probabilities. The weights are combined in the levels by summing them; this approach was selected as it gives equal emphasis on both components and small values have a lesser effect than, for example, in multiplication.

The two levels are combined by multiplying them. This was selected for the opposite reason; a small score on either of the levels reduces the weight more than two medium sized scores.

The weight is normalized using either l^2-normalization or *category normalization*. The former is the standard vector length normalization:

$$w_{l^2}(t, c) = \frac{w(t, c)}{\sqrt{\sum_{w \in d} w(w, c)^2}}. \tag{4.6}$$

With l^2-normalization the weight of each word is normalized for each document d separately. The normalized weight $w_{l^2}(t, c)$ of the word t is calculated by dividing the old weight $w(t, c)$ of the word t in the document d with the length of the vector of the document d.

Another way to normalize the weight is to use the maximum weight within the category:

$$w_n(t, c) = \frac{w(t, c)}{max_{w_i \in c} w(w_i, c)}. \tag{4.7}$$

The idea here is to divide each weight within a category with the maximum weight of the category. Here the weight of a word remains the same for the category in each document.

4.1.2 Approach II: Fragment length weighted category distribution

The second approach is the result of further development of the first approach. It was introduced in Article II and we call it *Fragment Length*

Weighted Category Distribution (*FLWCD*). As we decided to use a SVM classifier we also decided to make some updates to the initial term weighting method. The result of the update was an approach that substituted the document probability with Bi-Normal Separation and left out inverse category count.

Equation 3.8 in Chapter 3 described how BNS is calculated. It is based on the comparison of two distributions: word's occurrences in positive and negative samples. When compared with document probability $P(d|c)$, in Equation 4.4, both approaches use distribution of the word (or document with the word) in the positive sample, i.e., within the category, but BNS uses also the distribution of negative samples, i.e., documents that do not contain the word. In other words, when BNS compares the difference it emphasizes words that occur often in a single category. If the word is evenly distributed among several categories, BNS produces a smaller weight.

The weight is calculated by multiplying BNS, conditional probability of the category ($P(c|d)$), and inverse fragment length:

$$\begin{aligned} & w(t,c) \\ & = BNS(t,c) \times P(c|d) \times \mathit{ifl}(t) \\ & = |F^{-1}(\frac{N_{t,c}}{N_{t,c}+N_{\neg t,c}}) - F^{-1}(\frac{N_{t,\neg c}}{N_{t,\neg c}+N_{\neg t,\neg c}})| \\ & \times \frac{N_{d,c}}{N_{d,c}+N_{d,\neg c}} \times \frac{1}{\frac{1}{|f_t|}\sum l_f(t)}, \end{aligned} \tag{4.8}$$

where (using the notations from Table 3.1) $N_{t,c}$ is the number of times word t occurs in category c, $N_{d,c}$ is the number of times document with term t occurs in category c, and $N_{\neg d,\neg c}$ is the number of documents that are neither in the category c nor contain the word t. This equation corresponds Equation 11 in Article II.

The benefit of this approach is that it emphasizes the words that occur only in few categories ($P(c|d)$), in short fragments (*ifl*), and often in a single category and seldom in others (BNS).

Normalization can be done using either of the approaches presented in Equation 4.6 or Equation 4.7. Experimental results with both weight functions will be given in Section 5.

4.2 Keyword extraction

In this section I propose an approach for keyword extraction from short documents. This approach was originally presented in Article III. Keyword extraction is the task of finding the most important words of the document. This is useful in several domains: keywords can be used as a summary of the text, or text summarization can find the most important sentences by using the keywords [3]. The keywords can also be used in a tag-based recommendation system as tags. Keywords can also summarize the document collection and they can be used in query expansion.

To extract the keywords, the importance of each word needs to be evaluated. We use three levels of word assessment to identify the keywords: corpus level, cluster level and document level. The idea of multi-level word assessment is based on the work by Wan and Xiao [48]. The utilization of the extracted keywords is discussed in Section 5.3.

4.2.1 Corpus level word assessment

Corpus level word evaluation is described in detail in Article III, page 5. The aim of the corpus level evaluation is to find words that are important in the more abstract level. These words tend to be more common than the more expressive words but they should not be too common either. For example, we want to find terms like ’Rock and Roll’, ’Elvis’ and ’Metallica’ instead of just ’event’ and ’music’. Therefore, we concentrate on words that are neither too common or too rare in the corpus; however, an informative word will more likely be rare than common.

In order to find these types of words we use word frequency in the corpus level (tf_c). As in most cases when using a corpus of short documents, the term frequency within a document (tf_d) for each term is 1. We base our approach on Residual IDF; however, we do not compare IDF against expected IDF but instead we compare IDF against expected optimal IDF for the document collection.

We call the IDF used here *Frequency Weighted IDF* (IDF_{FW}). It is based on the idea of comparing the observed IDF with *Frequency Weight* (FW):

$$IDF_{FW}(t) = IDF(t) - FW(t), \tag{4.9}$$

where $FW(t)$ is the assumed optimal IDF which is described below. The most important terms receive $IDF_{FW}(t) = IDF(t)$, i.e., they receive no penalty with $FW(t)$.

The idea behind IDF_{FW} is to give smaller weights to words when the corpus level term frequency has a larger difference from the assumed optimal frequency n_o. Equation 4.10 shows how FW is calculated:

Table 4.1: Examples how IDF_{FW} changes when $tf_c = df$, $n_o = 93$, and $|D| = 3100$. This corresponds to one of the datasets used in our experiments presented in Article III.

$\alpha \backslash tf_c$	1	5	10	50	95	250	500	1000
1.0	5.06	5.06	5.06	5.06	5.00	2.21	0.21	-1.79
1.1	4.40	4.64	4.74	4.97	4.99	2.06	-0.04	-2.14
1.5	1.79	2.95	3.45	4.61	4.98	1.49	-1.01	-3.51
2.0	-1.48	0.84	1.84	4.16	4.97	0.78	-2.22	-5.22

$$FW(t) = \alpha \times |\log_2 \frac{tf_c(t)}{n_o}|, \tag{4.10}$$

where $tf_c(t)$ is the corpus level term frequency of word t and n_o is the assumed optimal frequency.

The penalty is estimated as IDF but we use n_o instead of the document count N and $tf_c(t)$ instead of $df(t)$. This affects the IDF so that all the term frequencies below n_o will get a positive value, when $tf_c(t)$ equals n_o the value is 0, and when $tf_c(t)$ is greater than n_o the value will be negative. To give penalty on both cases, we need to take the absolute value of the penalty.

Even though FW will be larger with small term frequencies, IDF will be also larger. In fact, when $tf_c(t) = df(t)$ and $tf_c(t) < n_o$, we have $IDF_{FW}(t_1) = IDF_{FW}(t_2)$ even if $tf_c(t_1) < tf_c(t_2)$ for all $tf_c(t) < n_o$. This can be seen in Table 4.1 when $\alpha = 1.0$. We use α to overcome this issue and give a small penalty when $tf_c(t) < n_o$. We have used $\alpha = 1.1$ in our experiments. Table 4.1 shows how the different α values and term frequencies affect the IDF_{FW} scores.

An important part of the equation is the selection of n_o. We use a predefined fraction of the corpus size: $n_o = 0.03 \times N$, where N is the number of documents in the corpus. That is, we consider that a word is optimally important in the corpus level when it occurs in 3 % of documents. We decided to use this number after evaluating the characteristics of our experimental data. This has produced good results in all of the experiments. However, it may be beneficial to change this value when datasets have different characteristics than the datasets described in Article III.

This informativeness evaluation has two features that we consider important: first, in the rare occasions when $df(t) < tf_c(t) < n_o$ these words are emphasized by giving a higher weight. Second, as we consider less frequent terms more informative than more frequent terms, the IDF_{FW}

is smaller when $tf_c(t) = n_o + C$ than when $tf_c(t) = n_o - C$ for any C, $0 < C < n_0$. For example, using the same parameters as in Table 4.1, when $tf_c(t) = n_0 - 43$ we get $IDF_{FW} = 4.97$, and when $tf_c(t) = n_0 + 43$ we get $IDF_{FW} = 3.91$. Here we can see how the less frequent words are emphasized over more frequent ones which is the desirable result in most of the cases when identifying the important words in the corpus level.

4.2.2 Cluster level word assessment

Cluster level assessment is introduced in Article III, pages 5 – 6. In the cluster level we want to emphasize words that occur often within a set of similar documents and rarely in other documents. That is, words that have high category probability $P(c|d)$, shown in Equation 4.3, are important in this level. This includes rare words, i.e., words with small corpus level term frequency tf_c. With event data, these types of words are often performers (”Elvis”, ”The White Stripes”), team names (”Tottenham”, ”FC Barcelona”), and directors and actors (”Tim Burton”, ”Johnny Depp”). These words are important when considering the document or the cluster level document collection as they indicate to the reader the exact contents of the event.

Cluster level evaluation is based on the feature weighting approach presented in Section 4.1.1. However, as the data used for keyword extraction rarely holds labels, we need to use clustering to identify document sets that have similar topics.

We use Agglomerative CompleteLink clustering in our work which is the same approach used by Wan and Xiao [48] in their work for keyword extraction. It is a bottom-up clustering approach where at the beginning, each document forms its own cluster. The clusters are joined iteratively so that in each iteration the most similar clusters are combined as long as the similarity between the two clusters is above a given threshold t_c. The similarity between the clusters c_n and c_m is the minimum similarity between any two documents $d_n \in c_n$ and $d_m \in c_m$:

$$sim(c_n, c_m) = \min_{d_n \in c_n, d_m \in c_m} sim(d_n, d_m), \tag{4.11}$$

where similarity $sim(d_n, d_m)$ is the cosine similarity of the documents. Cosine similarity measures the similarity between two vectors by assessing the cosine of the angel between them:

$$cos(d_n, d_m) = \frac{d_n \cdot d_m}{\|d_n\|\|d_m\|}. \tag{4.12}$$

Here, the dot product of d_n and d_m is the number of matching terms, and $\|d_n\|$ and $\|d_m\|$ is the length of the documents.

The algorithm stops when there are no more clusters to be joined, i.e., if there are no clusters with similarity above t_c. The optimal value for t_c can be found using, for example, cross validation and it varies among datasets.

It is possible to use the term weighting approach without clustering so that each document is considered to belong to its own cluster/category. In this case, the rarest words are emphasized as they occur only in a few documents. This does not produce optimal results but it still can identify important words from text.

We get the cluster level score $s_{cluster}(t, c)$ by using Equation 4.5 presented in Chapter 4. The result of the cluster level evaluation is a score for each word and for each of the clusters it appears in. If the word appears only in a single cluster, the weight will be considerably higher than if it appear in two or more clusters.

4.2.3 Document level word assessment

The final step of the process is to extract the keywords from the documents. This step is described in detail in Article III, pages 6 – 7. In this level we use the word scores from the previous levels. The idea is to extract the words that are found informative on either the corpus level or the cluster level; or preferably on both.

First, to make the scores of corpus and cluster level comparable, the corpus level scores need to be normalized to fall between [0,1]. This is done using Equation 4.7; i.e., the maximum corpus level word score in the document is taken and used as the denominator.

The document level score $s_{doc}(t, d)$ for word t in document d (that belongs to cluster c) is calculated by taking the weighted average of the cluster level score $s_{cluster}(t, c)$ and the normalized corpus level score $s_{corpus}(t)$:

$$s_{doc}(t, d) = \frac{\beta \times s_{cluster}(t, c) + (1 - \beta) \times s_{corpus}(t)}{2}, \qquad (4.13)$$

where β $(0 \leq \beta \leq 1)$ indicates the weight that is used for giving more emphasis to either cluster or the corpus level score. The weighted average is used for two reasons: 1) we get scores that fall within [0,1], and 2) as we want to extract words that are important in either corpus or cluster level, the effect of a low score on either levels does not exclude the word as it would if the scores would be multiplied. That is, we want to emphasize words that have a high weight on either of the levels instead of two medium weights on both levels. The optimal β values can be found using k-fold cross validation. The values we used are described in Article III.

Using this approach, if the document level score is close to 1, the word is informative both in corpus and in cluster level. If the score is close to 0.5, the word is informative either in the cluster level or in the corpus level. Such a word could be, for example, a performer that appears in a single document; this is an important word in the document level.

We noticed when examining the datasets that it is more probable that a keyword occurs often in the beginning of the document instead of the end; this was true especially with the event data. Therefore, we included a *distance* factor $d(t)$ that is based on an idea used in Kea keyword extraction system [16]. The distance is calculated by taking the number of words that precede the word's first occurrence in the document and dividing it with the length of the document:

$$d(t) = 1 - \frac{i(t)}{|d|}, \tag{4.14}$$

where $|d|$ is the number of words in the document and $i(t)$ is the index of word's first occurrence in the document. The index starts from 0 making $d(t) = 1$ for the first word in the document.

We also include part-of-speech tags (POS-tags) in the document level word evaluation as another weighting option for words: different tags get a different POS-weight (w_{POS}) in the final score calculation. This is useful when we want to emphasize different types of words in different domains. For example, in some domains adjectives may be important while in others they are not needed.

The simplest approach is to give weight 1.0 to all tags that are accepted, such as NP and JJ (nouns and adjectives), and 0.0 to all others. To emphasize some tags over the others, weights $w_{POS}(tag_1) > w_{POS}(tag_2) > 0$ can be used. If POS-tags are not available, $w_{POS} = 1.0$ is used for all words.

Finally, all words in the document d are scored by combining $s_{doc}(t, d)$, w_{POS} and $d(t)$:

$$s(t, d) = s_{doc}(t, d) \times d(t) \times w_{POS}(t), \tag{4.15}$$

where $w_{POS}(t)$ is the POS weight for the word t. If t has several POS-tags, the one with the largest weight is used.

Each word t in the document d now has a score $s(t, d)$ that indicates its informativeness for the document. The top k words that receive the highest score are then extracted as keywords from the document. As the documents are short we do not always want to extract all k words as some may have a very small score. Therefore we use a threshold t_d to select n ($n \leq k$) informative words from the document that have a score above t_d. The threshold t_d is relative to the highest score of the document:

$$t_d = \max_{t \in d} s(t, d) \times r \tag{4.16}$$

where r is the predefined cutoff variable. We use $r = 0.5$. That is, the keywords that have a score at least 50 % of the highest score in the document are acceptable.

4.3 Keyword association modeling for query expansion

In this section I describe the approach we have used for modeling associations between keywords and how we used the model for query expansion.

4.3.1 Association modeling

In Article IV, we propose an approach for modeling the relations between keywords by building an association network. The network is used in a search engine for query expansion. Even though keyword association focuses on keyword and keyword pair weighting we consider this a term weighting challenge.

The idea behind association network is to model the relations between concepts. In our case, these concepts are keywords that describe the contents of a document in a concise way. The method is based on the psychological theory that when two concepts appear often with each other, they tend to get a stronger association between them [38]. However, the associations are not the same every time. For example, we may usually associate the concept *car* to *driving*, but we may also think hundreds of other concepts, like *road*, *wheel* and *pavement* among other things.

To model the associations we use a network that consists of nodes and edges. The nodes in the network represent the concepts that can be words, terms or phrases. The nodes are linked together with directed edges that represent the strength of the association. Figure 4.1 from Article IV presents a small example of an association network.

Humans form associations between concepts when experiencing something [18, 38]. This can link together concepts that have no semantic relations between them. The strength of the experience also affects to the strength of the association. In human brain, this can be seen as having more neural pathways between the stronger associations [18].

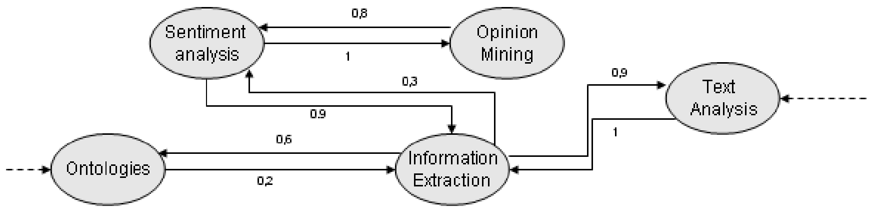

Figure 4.1: An example of an association network.

We used this theory when implementing the association network: an edge between two nodes holds a weight that indicates how strong the association is. We use documents as experiences and the keywords from the documents as the concepts. That is, a document is considered as a single experience that consists of concepts presented as keywords.

From a machine learning perspective, it is usually difficult, if not impossible to identify how strong an "experience" is. Therefore, we use confidence from association rule mining [1] to weight the associations. Confidence is the conditional probability of concept A occurring when the concept B occurs. For instance, when a document's topic is *cars*, it may contain the word *tyres* 25 % of times, making the confidence of *tyres* given *cars* 0.25. This is not symmetric, i.e., the confidence will be different when the topic is *tyres*; *cars* may occur about 50% of the times, making the confidence of *cars* given *tyres* 0.5. This example is arbitrary but it illustrates the antisymmetric properties of confidence and associations.

We use the following equation to compute the confidence of keyword c_2 given the keyword c_1, i.e., link from c_1 to c_2:

$$\mathit{confidence}(c_1 \rightarrow c_2) = \frac{\mathit{freq}(c_1 \cap c_2)}{\mathit{freq}(c_1)}, \tag{4.17}$$

where $\mathit{freq}(c_1 \cap c_2)$ is the number of times c_1 occurs with c_2, and $\mathit{freq}(c_1)$ is the number of times c_1 occurs in total.

In addition to confidence, we included two other features for associations: first, we assumed that the association is stronger between concepts that occur closely together. Second, we want to emphasize newer experiences and make their association stronger; i.e., we assume that the associations gradually deteriorate as time passes. The former is addressed with a distance function that indicates how closely together the concepts were experienced. Distance is an attribute that can vary depending on the data source. In an unstructured text, distance can be measured as the number of words, noun phrases, sentences or even paragraphs between the concepts.

As the concepts we are using are keywords, we use the location of keywords in the keyword list to measure the distance:

$$distance(c_1, c_2) = average(|i_{c_1} - i_{c_2}|), \tag{4.18}$$

where i_{c_n} is the index of the concept c_n in the keyword list. If the concepts always occur next to each other, the distance is 1. As $\log_{10}(1) = 0$, we use 0.01 as the distance factor when $distance(c_1, c_2) = 1$. When *distance*$(c_1, c_2) \geq 10$, we use 1.0 as the distance factor.

Now, the distance factor for concepts c_1 and c_2, when *distance*(c_1,c_2) falls within]1,10[, is:

$$DistanceFactor(c_1, c_2) = \frac{1}{\log_{10}(distance(c_1, c_2))}. \tag{4.19}$$

We also include time function that indicates the age of the concept pairing. If the age of the experience, i.e., document, can be deduced or extracted from the data, we simulate the natural deterioration of neural pathways by using the time function. The time function for concept c_1 takes the age of the documents it appears in:

$$TimeFactor(c_1) = 1 - \frac{\ln age(c_1)}{\gamma}, \tag{4.20}$$

where γ is used to shape the function to produce desirable results, $age(c_1)$ is the average age of the documents the concept appears in. Due to the nature of the data used in our experiments we used years but the age can also be calculated in days. If age is 0, we use 0.01 as the age. For example, if the document was created a month ago, its age is 30 days and *TimeFactor* = 0.94, when $\gamma = 60$. This means that the association has lost 6 % of its strength in one month when 60 days is used as γ. The γ can therefore be seen as an indicator of an old document.

The strength of the association from c_1 to c_2 is the combination of these three components:

$$Strength(c_1 \to c_2) = \frac{freq(c_1 \cap c_2)}{freq(c_1)} \times \frac{(1 - \frac{\ln age(c_1)}{\gamma})}{\log_{10}(distance(c_1, c_2))}. \tag{4.21}$$

In other words, the strength is the combination of association, time and distance.

The strength is normalized to fall between $[0, 1]$ by using the normalization presented in Equation 4.7. The normalization is done to all of the node's out-going edges so that the strongest weight is scaled to 1.0.

Our approach is somewhat similar with TextRank [29] algorithm as we consider co-occurrence as the main component of the weight between keywords. However, we weight the edges differently and the edges are antisymmetric.

4.3.2 Query expansion using keyword models

We use a spreading activation technique that is implemented using best first search. Here, the nodes are the keywords and the weight between two nodes is the association between the two keywords. The weights w fall between $0 \leq w \leq 1$. The best first search uses a function to select the top n nodes from the network by assessing the association between the query node q and the node k. Association between nodes q and k is the maximum value of the product of the weights in any of the paths from q to k.

As an example of spreading activation, consider Figure 4.1. Query *Sentiment analysis* would first expand *Opinion mining* as the weight is $1.0 > 0.9$ (*Information extraction*). Then *Information extraction* is expanded as it is the only neighbor left. From *Information extraction*, *Text analysis* is expanded as when compared to *Ontologies* its weight is larger: $0.9 \times 0.9 = 0.81$ versus $0.9 \times 0.6 = 0.54$. If there were a node from *Opinion mining* with the weight $0.54 < w < 0.81$ it would be expanded after *Text analysis* but before *Ontologies*.

The top n nodes with the highest association are used for expansion, where n is the predefined number of nodes. However, we also use a threshold for expansion; if the association is under the threshold the node will not be added to the expanded list even if there are less than n expanded nodes.

The result of this process is a set of nodes that are used to create a new query. The keyword from each of the nodes is added to the original query and this new query is then used to search the database. The results of the search are then weighted using the weights from the expansion. That is, the results received using the original query terms will get a higher relevance score and will be shown in the result set before the results from the expansion. Biggest benefit from expansion is received when the original search would result documents with poor or no match to the query. In those cases the expansion can produce results that have higher relevance to the original query.

Chapter 5

Evaluation and utilization of weighted features

In this chapter I summarize experimental evaluations of the three term weighting methods presented in the previous chapter. I use three cases: 1) categorization of market research data, 2) keyword extraction from event descriptions, and 3) keyword association modeling for query expansion.

First, in Section 5.1, I describe the short document data sources we have used in our work. In Section 5.2, I describe the categorization of short documents where we use the weighted terms for building the classifier. In addition, I show the experimental results of document categorization. In Section 5.3, I present the evaluation of the keyword extraction and how we used keywords to build keyword models for both user and item modeling, and query expansion.

5.1 Short document data sources

In our work we have several different datasets. Market research data is used in text categorization research and it contains the shortest documents. This data is important as there is a pressing need from the industry for a classifier that can automatically categorize the data as precisely as possible. The data is short as it is written by random respondents who often do not want to use much time when completing a survey. The average word count in the market research datasets we have used is between 4 and 12 words. We also collected a set of tweets from Twitter and use them for experimental evaluation.

Datasets consisting of event, movie and company descriptions are usually proof-read. That is, the information found from the descriptions is

Table 5.1: Characteristics of the datasets used in our work.

Dataset	Number of documents	Average number of words per document	Average TF per document
Market research	7,200	6	1.01
Twitter	2,650	15	1.05
Event descriptions	5,000	32	1.04
Movie descriptions	7,000	63	1.07
Velo data	1,000	80	1.09

often more reliable than, for example, in tweets. We use this type of data for keyword extraction. This type of data is longer than tweets or market research data but it is still short. Their shortness is due to the fact that the data is often accessed using a mobile device and the user wants to find the relevant information quickly. Extracting information from the short descriptions is important for three reasons: 1) this may be the only information we have on the subject, as is the case with the event data, 2) we want to extract the concepts from the documents we know the user has seen (as opposed to the complete descriptions) and use them for building user models, and 3) when a description is written, the writer wants to include all the relevant information about the topic in the description. That is, the short document may contain as much relevant information as the longer version but in a more concise form.

Table 5.1 shows the characteristics of data we have used in our work. As can be seen, even though some of the datasets contain over 50 words per document, it is rare that a word occurs more than once within a single document. For example, even in Velo data (Velo is a company based in Shanghai, China that maintains coupon machines), which contains 80 words per document, the average term frequency is only 1.09.

The market research data, the event descriptions and the Velo data are from actual applications for which these methods have been developed. The Twitter dataset and movie descriptions are public datasets we have used for additional experiments. Next, I will briefly describe each of these datasets.

The market research dataset consists of over ten sets of answers to different questions and surveys. This dataset was received from a market research company located in Finland. This is actual data they have used in their studies. Each dataset contains between 200 and 1,050 documents (i.e., answers). The average number of words per document varies between 3.5 and 12 making the documents very short. The data was collected between 2009 and 2011.

Twitter dataset was downloaded from Twitter in September 2011. We created three different datasets from the data and used them for evaluating the text categorization approaches. We downloaded only tweets that were written in English.

The event dataset consists of approximately 5,000 public events from the Helsinki Metropolitan area. The events were collected between 2007 and 2010 from several different data sources. After preprocessing, where stop words and noisy characters are removed, the documents hold 32 words on average. The average term frequency per document in this dataset was 1.04, i.e., almost all the words occur on average only once within a document.

Velo dataset contains 1,000 descriptions of companies and their products stored in the Velo databases. These descriptions are used in Velo coupon machines in China. The data was gathered in June 2010. The descriptions hold 80 words on average and they are written in Chinese. This data was used for keyword extraction experiments.

Movie descriptions are abstracts downloaded from Wikipedia. The dataset contains approximately 7,000 Wikipedia pages that hold information about different movies. We use MovieLens dataset[1] when selecting the movies: if a movie is found from MovieLens dataset, we downloaded its Wikipedia page. We only use the abstracts found at the beginning of the Wikipedia page. If the abstract is longer than 100 words, we remove the last full sentences to shorten the document under the given limit. The average word frequency per document in this dataset is 1.07. The Wikipedia pages were retrieved in May 2010.

When we compare these datasets to the Reuters-21578 set, we can see that the statistics differ greatly. The Reuters set contains 150 words per document on average after the stop words are removed and its average term frequency is 1.57 [44].

We use a dataset of 9,000 project descriptions for keyword association modeling. The descriptions have a short abstract what the project is about and a set of keywords that describe the concepts of the project.

[1]`http://www.grouplens.org/node/12`

5.2 Categorization of short documents

One of the ways a market research company collects data from consumers is using online questionnaires. Often these questionnaires hold open ended questions; these questions do not have a predefined set of answers but the respondent can write anything they feel fit. These answers are usually labeled manually by market research professionals by reading each of the answers and selecting suitable labels [45]. As each questionnaire can get thousands of answers this is an arduous task.

Our aim has been to automatize the labeling process by using supervised learning approaches. When implementing a classifier the biggest challenge has been to weight the terms to emphasize the most important words. This is important to get more precise classification results; if each word would have the same impact in classification the common words would misdirect the classifier and produce poor results.

The process of classification is done in three steps: first, the features (i.e., words) are weighted and the weighted features are used to build the feature vectors that are used for training the classifier. The second step, classifier training, depends on the selected classification method. Finally, the test documents, i.e., documents that do not have a label, are classified using the trained classifier.

We experimented with two different classifiers for market research data. First, we used a Naive Bayes like classifier that uses the weights to assign a category for each text fragments [45]:

$$class(f_n) = \arg\max_c \Big(\sum_{i=1}^{|f_n|} w_i(c) + \sum_{i=1,j=2}^{|f_n|} w_{i,j}(c)\Big), \tag{5.1}$$

where f_n is the nth fragment of document d, w_i is the weight of the ith word in f_n, and $w_{i,j}$ the weight of the word pair i, j in f_n for the category c. The weight for the word pair is the summed weight of both words in the category.

The idea is to sum the weights of each word and word pair in the fragment for each category and assign the fragment to the category that receives the highest score.

The second classifier we used was a Support Vector Machine classifier that uses a linear kernel. We used an implementation called SVM^{light} [23] of the classifier. We used the feature weighting approach described in Section 4.1.2 and l^2-normalization.

In addition to market research data, we also experimented by categorizing Twitter messages [44]. The Twitter data we collected hold 15 words

Table 5.2: Comparison of the feature weighting methods from Article II. Compared approaches are Approach I (A I), Approach II (A II), Bi-Normal Separation (BNS), Chi-squared (χ^2), Information Gain (IG), Odds Ratio (OR), Residual IDF (RIDF), and Term Frequency - Inverse Document Frequency (TFIDF).

Dataset	A II	A I	BNS	χ^2	IG	OR	RIDF	TFIDF
Market	**0.72**	**0.70**	**0.70**	**0.66**	**0.65**	**0.70**	**0.67**	**0.67**
Twitter	**0.73**	**0.69**	**0.64**	**0.70**	**0.59**	**0.67**	**0.46**	**0.47**
Average	***0.73***	***0.70***	***0.67***	***0.68***	***0.62***	***0.69***	***0.57***	***0.57***

per message on average. We use SVM classifier with the feature weighting approach presented in Section 4.1.2.

The experiments were made by dividing the data into two sets: training set and the test set. The test set was held separate during the whole training process, i.e., the parameter tuning was done using only the training set. Both the market research and the Twitter datasets were manually annotated for categories. However, we did conduct cross-annotator evaluation for these datasets.

We use F-score as the metric for comparing the results. This is the standard measure for test accuracy used in several research articles in this area [14, 15, 22, 50, 52]. It is the harmonic mean of precision (the number of correct categories out of all the predicted categories) and recall (the number of correct categories out of all the categories of the document). As we want to emphasize precision, we use the $F_{0.5}$-metric:

$$F_{0.5} = (1 + 0.5^2) \times \frac{\text{precision} \times \text{recall}}{0.5^2 \times \text{precision} + \text{recall}}. \tag{5.2}$$

The emphasis on precision was done due to the real world requirements of the market research company. As they stated, the classifier can miss some classifications but the ones it makes should be accurate.

We compared the performance of the following term weighting approaches: Approach I, Approach II, BNS, Chi-squared, Pointwise Mutual Information, Information Gain, Odds Ratio, Residual IDF, TF-IDF, and Term Frequency. We used 15 different datasets of the two domains as the test sets. We used SVM classifier in the comparison.

Table 5.2 shows the results of the evaluation. In the table, Market shows the average $F_{0.5}$-scores of the experimental evaluation of the market research data, and Twitter the averages of Twitter data. Average is the average of the two test cases. We omitted Mutual Information and Term

Frequency from Table 5.2 due to space limitations but they can be found from Table 4 in Article II. The results of Mutual Information is on the level with Information Gain and the results of Term Frequency are on the level with TF-IDF.

The comparison showed that Approach II produced the best results when used in a Support Vector Machine classifier. In Market research experiment there is only a little variance between the results of different approaches. In our opinion, this is due to the shortness of the documents. When the documents are short enough, the impact of term weighting is smaller as there are only a few features for SVM. However, when the documents are a bit longer, like in Twitter experiment, the impact of the term weighting is greater. This can also be seen when comparing the results of market research test set of different length (Article II).

As selecting the best classifier is an important challenge, we compared three classifiers in Article I and extended the comparison in Article II with more test sets and more classifiers. The compared classifiers were: k-Nearest Neighbors, two Naive Bayes classifiers and a Support Vector Machine classifier. The average score of the classifiers using Approach II in the two domains were: SVM 0.73, Naive Bayes 0.64, kNN 0.57, and TWCNB (Naive Bayes) 0.23 (Article II). TWCNB is the Naive Bayes approach we used as the starting point in our work in Article I. Its poor result is most probably due to its strong relation to term frequency.

Even though this comparison does not include all the possible classifiers, it is safe to assume that SVM is the best option for short document categorization in most cases as it is the best classifier in several other domains and usually with long documents as well [22].

5.3 Keyword extraction and keyword models

Next, I present the experimental evaluation of the proposed keyword extraction approach and two ways we have utilized the extracted keywords. The first use is for user and item profiling for recommendation, where the extracted keywords are used as tags. This was done as a more objective evaluation of the extracted keywords. The second use is for creating keyword association models that are used in a search engine. The aim is to expand a search so that highly associative keywords are added to the query.

The approaches we use for both keyword extraction and keyword association modeling require some parameter selection. Even though the selected parameters have an effect on the results I leave the more detailed evaluation on their effects out of scope of this thesis. The parameters we used and the reasons for their selection can be found from Article III and Article IV.

5.3.1 Experimental results of keyword extraction

The evaluation of the proposed keyword extraction approach is described in Article III. In this section I give a short summary of the evaluation.

The evaluation was done by comparing the keyword extraction precision of several approaches. We used a number of documents from different datasets that were manually tagged for keywords. We evaluated the annotator agreement rate which in the end was quite low, between 64 % and 70 % depending on the dataset. We updated the annotations after the disagreements were resolved. We used F-score as the metric of accuracy.

We included the following keyword extraction approaches into our evaluation: CollabRank, KeyGraph, χ^2 based keyword extraction from Matsuo and Ishizuka [27], Chi-squared feature weighting, and TF-IDF. We use the extraction of all nouns and adjectives as the baseline in our studies. This baseline was selected as it is the simplest approach and can be effective if the document is so short that it will produce only a few keywords.

The performance comparison is done using three different datasets. The datasets we used were: event descriptions in Finnish, Velo coupon descriptions in Chinese, and Wikipedia movie descriptions in English. From each of these datasets we manually picked keywords for 300 documents to be used as a gold standard. As the keywords were picked manually, they tend to be subjective; i.e., someone's opinions for the most important words for the document.

Table 5.3 shows the results of this experiment. As can be seen, with movie descriptions our approach (IKE) produced the best result with F-score of 0.57. Second best were Chi-squared feature weighting and TF-IDF with the score of 0.35. Rest of the approaches produced scores under 0.3. With event descriptions our approach produced the F-score of 0.56. TF-IDF and Chi-squared tied again for the second with the score of 0.49. CollabRank scored 0.46. Rest were under 0.40. With Velo data our approach produced again the best result with the F-score of 0.31. Chi-squared was second with the score of 0.26 and the rest were under 0.25. The baseline results were: 0.22, 0.39, 0.15 for movies, events and Velo, respectively. In this comparison our approach clearly out-performs the competition.

We can see that the baseline does not produce good results. That is, by extracting only all the nouns and adjectives is not beneficial. As was expected, a term frequency based approach (TF-IDF) does not produce very good results when compared to our approach. The poor results for KeyGraph and Matsuo's keyword extraction approaches are the result of

Table 5.3: F-scores for each of the method in keyword precision experiment. Chi-squared is the traditional feature weighting approach, and χ^2 KE is the keyword extraction approach presented by Matsuo [27]. Due to the Chinese character set, we were unable to evaluate KeyGraph and Matsuo's χ^2 keyword extraction approach on Velo data. This table was originally presented in Article III.

	IKE	Collab-Rank	Chi-squared	TF-IDF	KeyGraph	χ^2 KE	Baseline
Wikipedia	0.57	0.29	0.35	0.35	0.22	0.21	0.22
Events	0.56	0.46	0.49	0.49	0.36	0.35	0.39
Velo	0.31	0.18	0.26	0.22	-	-	0.15

the extraction from a single document at a time; as the documents are short the documents do not contain enough information for these two approaches.

5.3.2 User and item modeling for recommendation

As the manually tagged keywords used in the keyword extraction evaluation were subjective, we also wanted to do an objective evaluation of the extracted keywords. Therefore, we used the keywords as tags for user modeling and recommendation, and report the recommendation results. In Article III we evaluated the keyword extraction by using the keywords for user and item (item is an abstract term used to depict the domain in question, e.g., movies) modeling. The aim was to carry out an objective performance comparison between several competing approaches by using the models for recommendation.

As the document set, we used movie descriptions downloaded from Wikipedia that were at most 100 words long. We extracted the keywords from the descriptions using the previously described keyword extraction approach. The keywords were then used as tags for the movies, i.e., the keywords formed the item model for the movies. Each movie has now a set of tags that consists of keywords. We used MovieLens data for creating the user model. MovieLens data holds movie ratings from users that indicate their feelings towards the given movie. The ratings fall between 1-5, where 1 indicates dislike. We take the ratings and add tags from the item model to user model based on the user ratings. For example, if the user has rated a movie with five stars (out of five), the tags from the movie are added to the user model with the maximum weight (1.0).

Formally, the item model (where items are movies) consists of the movie title m, and a set of tags T_m linked to the title. The tag set T_m consists of tags $t \in T_m$ extracted from the movie description d_m. User model U_n for the user n is created by using the set of tags T_m from each movie the user has rated. The tags are added to the user model by giving each of the tags $t \in T_m$ a weight w_t which corresponds the user's average rating for the movies where the tag appears in. The user model consists of tag-weight pairs $(t, w_t) \in U_n$.

The user model can be used for recommendation by scoring each of the movies the user has not seen by using the item model T_x (of the unseen movie) and comparing it with the user model U_n. The score for a movie x the user n has not seen is:

$$s(x, n) = \sum_{t \in T_x} w_t : (t, w_t) \in U_n. \tag{5.3}$$

That is, the score is the sum of the weights of the matching tags. The movies with the highest scores are then added to the top-n list of movies in descending order.

We used this approach when evaluating the performance of the keyword extraction approaches. The keywords were extracted using the approach described in Section 4.2. For comparison purposes we also built models using the other keyword extraction approaches used for evaluation in the previous section. The performance was evaluated by using the models for recommendation: a test set was built by taking movies the user had rated highly (these movies were not included to the training set) and adding them to a set of randomly selected movies the user has not seen.

The recommendation score for each movie was the combination of precision and coverage. Precision indicates the number of movies that are recommended to the user are actually movies the user has seen and liked. Coverage is the number of user models created from the keywords that can be used for recommendation. We included coverage as some of the approaches produce models that do not produce any recommendations as they are too narrow.

Table 5.4 from Article III shows the results of the recommendation experiment. Our approach produced the recommendation score of 0.43, which is the best in the comparison. Chi-squared keyword extraction produced 0.29, and the rest of the scores were between 0.24 and 0.26. The baseline produced the score[2] of 0.35. This experiments shows that our approach clearly out-performs the competition in this objective experiment as well. The baseline produced quite a good result which is due to the fact that its coverage is high even though its precision was quite low.

[2]In Article III, there is an error with this score. The correct score is presented here.

Table 5.4: Comparison of user models for recommendation when extracted keywords are used for user modeling. This table was originally presented in Article III. Our method is shown in the column IKE.

	IKE	Collab-Rank	Chi-squared	TF-IDF	KeyGraph	χ^2 KE	Baseline
Precision	0.55	0.41	0.84	0.86	0.30	0.33	0.39
Coverage	0.75	0.59	0.27	0.29	0.86	0.85	0.89
Total Score	**0.41**	**0.24**	**0.23**	**0.25**	**0.26**	**0.28**	**0.35**

5.3.3 Keyword association modeling for query expansion

We use keywords to build an association network for query expansion in a business intelligence search engine. The idea behind query expansion using association network is to include new search terms to the query by finding the corresponding node for the query term from the association network and using spreading activation described in Section 4.3.2.

In order to use the query expansion technique we need to build a model that describes the associations between the keywords. We weight these associations using the approach described in Section 4.3.1. The result of this process is an association network.

To evaluate the association network we used a set of project descriptions as the document set. These documents contained manually annotated keywords, i.e., we did not use a keyword extraction approach for tagging the documents. Table 5.5 (from Article IV) shows an example set of associations received using this approach. The table shows strength of the association both with and without the time factor from Equation 4.20. This indicates the effect of the time factor; for example, association between *gprs* - *umts* drops 6.5 % from 1.0 to 0.935 due to the fact the topic is old and not relevant any more as there are no new projects related to those topics.

We did not evaluate the performance of the query expansion formally as our focus was on association model creation and keyword weighting. Our query expansion evaluation focused on gathering comments from the search engine users about the search results they received. The users were co-workers from VTT that were doing project planning and needed some business intelligence information. The comments were mostly positive (Article IV) but as the users may have been biased (less likely to give negative feedback to a co-worker) this is only a small indication of the feasibility of the system.

We evaluated the implemented network by manually going through the links and weights. Our findings are shown in Table 5.6 from Article IV.

Table 5.5: A sample of keyword associations from Article IV. The last column indicates the weight after the age has been factored in. The age indicates years since the project ended.

Concept (from)	Concept (to)	Weight without age	Final weight
satellite picture	satellite image	1.0	1.0
building information modelling	safety	1.0	1.0
waste combustion	biomass	1.0	0.98
ontology	reasoning	1.0	0.97
rfid tag	barcode	1.0	0.96
pulping process	pulping industry	1.0	0.95
gprs	umts	1.0	0.94
road	asphalt	1.0	0.92
competitor survey	SME	1.0	0.91
sun	isotropy	1.0	0.91
rime	ice formation	1.0	0.90
screwdriver	hand saw	1.0	0.90
polymer	plastic	1.0	0.85
apms	paper machine	0.96	0.89
sea level	climatic change	0.93	0.88
felling	pulpwood	0.90	0.85
mobile telephone	local area network	0.90	0.85
lightweight concrete	stiffness	0.90	0.81
aerial photography	aerial survey	0.83	0.76
online measurement technology	high pressure	0.71	0.65
atmosphere	scanning	0.63	0.58
testing methods	failure	0.55	0.50
process simulation	processes	0.52	0.49
rye	wheat	0.42	0.45
energy conservation	fuel consumption	0.22	0.21
food processing	electric device	0.09	0.07
enzyme	health care	0.01	0.02

Table 5.6: Results of the network evaluation originally shown in Article IV. Positives were considered as correctly weighted, negatives incorrectly weighted. Higher and lower indicates whether the negatives should be valued higher or lower.

Weight	Positives	Negatives	Higher	Lower
1.0	92%	8%	0%	100%
0.3 - 0.7	60%	40%	85%	15%
0 < 0.1	45%	55%	100%	0%

Considering whether a link between two nodes is relevant or if the weight is correct is difficult. However, this evaluation still gives some indication about the feasibility of the network. In our assessment, among the links with strong relation the link itself is often feasible even if the weight might be too strong.

We also estimated that weaker links are less likely to be feasible. However, even among the links with small weights there are still a high percentage of feasible links left. As can be seen from Table 5.5, this approach can produce links that are good for query expansion. For example, when querying *gprs*, expanding the query with *umts* is feasible. The same applies, for example, with *satellite picture* and *satellite image* pair.

Chapter 6

Contributions of this thesis

The focus of this thesis is to tackle the TF=1 challenge of short documents. In this thesis I aimed to answer three research questions:

1. How to weight terms in short documents for document categorization to overcome TF=1 challenge?
2. How to extract the most informative words from short documents?
3. How to weight keywords in short documents and utilize them in a search engine?

To answer these questions we have developed term weighting approaches that utilize such word statistics as average fragment length, category count, distribution of categories of the word, and distribution of the word within the category. We also utilized Bi-Normal Separation that weights the terms based on the distribution of a word within positive and negative samples. In addition, we have used word's frequency among all the documents, its location in the document and keyword co-occurrences as the components in our term weighting approaches.

Our approaches to these challenges have been presented in Articles I - IV and summarized in Chapters 4 and 5. Next, I give a short summary of those answers.

Question 1. How to weight terms in short documents for document categorization to overcome TF=1 challenge?

I proposed two approaches for term weighting that are aimed to be used with two different classifiers. The approaches are based on each term's distribution within a category and among categories. These weighting approaches are described in detail in Article I and Article II. The proposed approaches have a distinct benefit when compared to more traditional approaches such as TF-IDF: they do not rely on word's term frequency within a document, overcoming the TF=1 challenge. Our experiments (Section 5.2) show that the proposed approaches produced promising results when categorizing short market research data and tweets with TF=1.

Question 2. How to extract the most informative words from short documents?

I proposed an approach for extracting keywords from short documents (Section 4.2). This approach weights the words on three levels and extracts the words with the highest weight. When compared with the competing keyword extraction approaches our approach produces the best results in both of our experiments (Section 5.3.1 and 5.3.2). The approach and the experiments are described in detail in Article III.

Question 3. How to weight keywords in short documents and utilize them in a search engine?

I proposed an approach for weighting the links between keywords (Section 4.3). This produces a model of the keyword relations called an association network. I presented how this network is used for query expansion (Section 5.3.3). This is originally presented in Article IV. We evaluated the resulting network by manually assessing links in the network. The results showed that most links with a high weight were correct. In addition, a large percentage of medium to low weights were also feasible (Section 5.3.3). In our opinion, this shows that the network is feasible and can be used for query expansion.

Chapter 7

Discussion and conclusions

In this thesis I have addressed the challenge of term weighting in short documents. The main issue I focused on is the fact that a word seldom occurs more than once in a single document. This makes document level word assessment, i.e., taking only the words within the document into consideration, ineffective and raises the need for novel term weighting approaches in short documents.

We have developed the term weighting methods for two different text mining cases: categorization of market research data and keyword extraction from event, movie and company descriptions. In text categorization, the motivation for our work came from a market research company that needed software for automatic categorization of their surveys. We have proposed two approaches (Article I and Article II) which both have been used by the market research company. Currently they are using Approach II with a SVM classifier.

The solution could be further improved in various aspects. When using SVM as the classifier the processing time is considerably longer than when using, for example, a Naive Bayes classifier. This is due to the binary classification process used in the solution. Especially cases where there are thousands of answers and tens of categories, the processing time is too long. Even though the proposed approaches produce the best results among the experimented approaches in terms of F-scores, the results of some of the datasets are not good enough for real use. In our opinion, the experimental evaluation should be extended by including various types of datasets, by performing cross-annotator validation and by assessing the impact of each component in the weighting methods. This way we may find the reasons for poor performance in some cases and get a stronger validation for the proposed approaches.

Another interesting task for future is to experiment with datasets of longer texts. In this work we used only very short documents as the main focus was on market research data. The longer documents could be, for example, product and event descriptions that contain less than 100 words. In addition, comparing the proposed approach with longer data (e.g., Reuters-21578) could be interesting.

To extract keywords from short descriptions we developed a keyword extraction approach that evaluates the words on three levels. The experiments produced promising results. The main reason for the good performance was that it extracts keywords of two different levels: lower abstraction level keywords such as the performer of the event (e.g., *Rolling Stones*) and more general keywords such as the type of the event (*Rock concert*).

Currently, the proposed approach is in use for keyword extractions from tv-show descriptions where the keywords are used as tags in a prototype recommendation system. In the future we will experiment with Approach II for the cluster level word evaluation instead of Approach I.

The third weighting approach I proposed in this thesis is used for keyword association modeling. The aim is to create an association network that can be used in query expansion using spreading activation technique. We did not evaluate the query expansion formally as we focused our efforts on model evaluation.

We have used this approach for domain modeling in a prototype business intelligence search engine. This modeling approach may not be feasible in a large search engine as defining the associations between keywords of wide range of domains is difficult. Currently, we are adopting this approach to be used in combination with a fuzzy ontology in a search engine that focuses on information about chemicals. The aim is to find associations between keywords related to the chemical domain and link the keywords to a fuzzy ontology. This model is then used for query expansion.

The main lesson I have learned when working with all of these approaches for term weighting in short documents is that approaches based on term frequency within a document, such as TF-IDF, do not perform well with short documents. As TF has been the corner stone for the most previous approaches, this finding is important.

The experiments presented in this thesis support this argument: in all of the experiments our approaches clearly out-perform the TF based approaches. In addition, when we compared the performance against a wide range of other relevant approaches, all of our approaches out-performed the competition. Therefore I can conclude that the novel term weighting methods presented in this thesis perform well for the tasks they were designed for: assessing the weight or importance of words in short documents.

In the future, the focus of term weighting research should be on finding a "TF-IDF" approach for short documents; i.e., an approach that is universally effective and can be used with many languages and with a wide variety of data sources. In this thesis I have laid the first stepping stones on this road. I hope that some, if not most, of the components we have used in our work can be utilized as the basis for future work in this area.

As the short document data sources such as Twitter continue growing, one thing is for certain: short documents will continue to offer great new challenges to feature weighting and text mining in general for years to come.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93), USA*, pages 207–216, May 1993.

[2] M.A. Andrade and A. Valencia. Automatic extraction of keywords from scientific text: Application to the knowledge domain of protein families. *Bioinformatics*, 14:600–607, 1998.

[3] E. D' Avanzo and B. Magnini. A keyphrase-based approach to summarization: the LAKE system at DUC-2005. In *Document Understanding Workshop (DUC'05), Canada*, October 2005.

[4] F. Benevenuto, G. Mango, T. Rodrigues, and V. Almeida. Detecting spammers on Twitter. In *Seventh annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS'10), USA*, July 2010. Available at http://ceas.cc/2010/papers/Paper%2021.pdf (accessed 10.10.2012).

[5] J. Bhogal, A. Macfarlane, and P. Smith. A review of ontology based query expansion. *Information Processing & Management*, 43(4):866 – 886, 2007.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] A. Bookstein and D. R. Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 5(25):312–318, 1974.

[8] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, 44(1):1:1–1:50, January 2012.

[9] K.W. Clark and W.A. Gale. Inverse Document Frequency (IDF): A measure of deviation from Poisson. In *Third Workshop on Very Large Corpora, Massachusetts Institute of Technology Cambridge, USA*, pages 121–130, June 1995.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[11] F. Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.

[12] G. Ercan and I. Cicekli. Using lexical chains for keyword extraction. *Information Processing and Management*, 43(6):1705–1714, 2007.

[13] S. G. Esparza, M. P. O'Mahony, and B. Smyth. Towards tagging and categorization for micro-blogs. In *Proceedings of 21st National Conference on Artificial Intelligence and Cognitive Science (AICS'10), Ireland*, September 2010. Available at http://irserver.ucd.ie/handle/10197/2517 (accessed 10.10.2012).

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

[15] G. Forman. BNS feature scaling: an improved representation over TF-IDF for SVM text classification. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08), USA*, pages 263–270, October 2008.

[16] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99), Sweden*, pages 668–673, August 1999.

[17] Y. HaCohen-Kerner. Automatic extraction of keywords from abstracts. In *7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'03), United Kingdom*, pages 843–849, September 2003.

[18] D. Hebb. *The organization of behavior: a neuropsychological theory.* New York: Wiley, 1949.

[19] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Conference on Empirical Methods in Natural Language Processing (EMNLP'03), Japan*, pages 216–223, July 2003.

[20] A. Hulth. Enhancing linguistically oriented automatic keyword extraction. In *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04), USA*, pages 17–20, May 2004.

[21] A. Hulth, J. Karlgren, A. Jonsson, H. Boström, and L. Asker. Automatic keyword extraction using domain knowledge. In *Proceedings of Second International Conference on Computational Linguistics and Intelligent Text Processing (CICLing'01), Mexico*, pages 472–482, February 2001.

[22] T. Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. In *10th European Conference on Machine Learning (ECML'98), Germany*, pages 137–142, April 1998.

[23] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical, pages 41–56. MIT Press, 1999.

[24] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes. Multinomial Naive Bayes for text categorization revisited. In *17th Australian Joint Conference on Artificial Intelligence (AI'04), Australia*, pages 488–499, December 2004.

[25] A. Krishnakumar. Text categorization building a kNN classifier for the Reuters-21578 collection, 2006. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.135.9946 (Accessed 10.10.2012).

[26] D. D. Lewis. Reuters-21578. http://www.daviddlewis.com/resources/testcollections/reuters21578/ (accessed 10.10.2012).

[27] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIR'03), USA*, pages 392–396, May 2003.

[28] M. McCord and M. Chuah. Spam detection on Twitter using traditional classifiers. In *Proceedings of 8th International Conference on Autonomic and Trusted Computing (ATC'11), Canada*, pages 175–186, September 2011.

[29] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, (EMNLP'04), A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Spain*, pages 404–411, 2004.

[30] D. Mladenic and M. Grobelnik. Feature selection for unbalanced class distribution and Naive Bayes. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Slovenia*, pages 258–267, June 1999.

[31] T. Moh and A. J. Murmann. Can you judge a man by his friends? - enhancing spammer detection on the Twitter microblogging platform using friends and followers. In *Proceedings of 4th International Conference on Information Systems, Technology and Management (ICISTM'10), Thailand*, pages 210–220, March 2010.

[32] T. D. Nguyen and M. Kan. Keyphrase extraction in scientific publications. In *Proceedings of 10th International Conference on Asian Digital Libraries Asian Digital Libraries (ICADL'07), Vietnam*, pages 317–326, December 2007.

[33] Y. Ohsawa, N. E. Benson, and M. Yachida. KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Proceedings of IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL'98)*, pages 12–18, April 1998.

[34] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford, 1998.

[35] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'10), Malta*, May 2010.

[36] J. Pearl. *Heuristics - intelligent search strategies for computer problem solving*. Addison-Wesley series in artificial intelligence. Addison-Wesley, 1984.

[37] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing (3rd ed.)*, chapter 16.5. Support Vector Machines. New York: Cambridge University Press, 2007.

[38] J. Raaijmakers and R. Schiffrin. Search of associative memory. *Psychological Review*, 8(2):98–134, 1981.

[39] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the poor assumptions of Naive Bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03), USA*, pages 616–623, August 2003.

[40] J. D. M. Rennie and T. Jaakkola. Using term informativeness for named entity detection. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05), Brazil*, pages 353–360, August 2005.

[41] A. Ritter, C. Cherry, and B. Dolan. Unsupervised modeling of Twitter conversations. In *Human Language Technologies - Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL'10), USA*, pages 172–180, June 2010.

[42] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[43] F. Sebastiani. Text categorization. In Laura C. Rivero, Jorge Horacio Doorn, and Viviana E. Ferraggine, editors, *Encyclopedia of Database Technologies and Applications*, pages 683–687. Idea Group, 2005.

[44] M. Timonen. Categorization of very short documents. In *Internation Conference on Knowledge Discovery and Information Retrieval (KDIR'12), Spain*, pages 5–16, October 2012.

[45] M. Timonen, P. Silvonen, and M. Kasari. Classification of short documents to categorize consumer opinions. In *Online Proceedings of 7th International Conference on Advanced Data Mining and Applications (ADMA'11), China*, December 2011. Available at http://aminer.org/PDF/adma2011/session3D/adma11_conf_32.pdf (accessed 10.10.2012).

[46] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[47] P. D. Turney. Coherent keyphrase extraction via web mining. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03), Mexico*, pages 434–442, August 2003.

[48] X. Wan and J. Xiao. CollabRank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of 22nd International Conference on Computational Linguistics (COLING'08), United Kingdom*, pages 969–976, August 2008.

[49] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries (DL'99), USA*, pages 254–255, August 1999.

[50] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[51] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99),USA*, pages 42–49, August 1999.

[52] Y. Yang and J.P. Pedersen. Feature selection in statistical learning of text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), USA*, pages 412–420, July 1997.