Informativeness-based Keyword Extraction from Short Documents

Mika Timonen[1,3], Timo Toivanen[1], Yue Teng[2], Chao Cheng[2] and Liang He[2]
[1]*VTT Technical Research Centre of Finland, PO Box 1000, 02044 Espoo, Finland*
[2]*East China Normal University, Institute of Computer Applications, No.500 Dongchuan Road, 200241 Shanghai, China*
[3]*Department of Computer Science, University of Helsinki, Helsinki, Finland*

Keywords: Keyword Extraction, Machine Learning, Short Documents, Term Weighting, Text Mining.

Abstract: With the rise of user created content on the Internet, the focus of text mining has shifted. Twitter messages and product descriptions are examples of new corpora available for text mining. Keyword extraction, user modeling and text categorization are all areas that are focusing on utilizing this new data. However, as the documents within these corpora are considerably shorter than in the traditional cases, such as news articles, there are also new challenges. In this paper, we focus on keyword extraction from documents such as event and product descriptions, and movie plot lines that often hold 30 to 60 words. We propose a novel unsupervised keyword extraction approach called Informativeness-based Keyword Extraction (IKE) that uses clustering and three levels of word evaluation to address the challenges of short documents. We evaluate the performance of our approach by using manually tagged test sets and compare the results against other keyword extraction methods, such as CollabRank, KeyGraph, Chi-squared, and TF-IDF. We also evaluate the precision and effectiveness of the extracted keywords for user modeling and recommendation and report the results of all approaches. In all of the experiments IKE out-performs the competition.

1 INTRODUCTION

As there are more and more user created content on the Internet, short documents have become an important corpus in several text mining areas. The most relevant sources of short documents currently are product descriptions, Twitter messages, consumer feedback and blogs. In most cases, these documents have less than 100 words and contain only a few sentences. For example, Twitter messages contain at most 140 characters (around 20 words).

Keyword extraction, also known as keyphrase extraction[1], is an area of text mining that aims to identify the most informative and important words and/or phrases, also called terms, of the document. It has uses in several different domains, including text summarization, text categorization, document tagging and recommendation systems.

The challenge with keyword extraction is similar with the challenge of feature weighting in text categorization as both aim to assess the impact of the words in the document. In text categorization the weights are used when training the classifier whereas keyword extraction uses the weights to find the most important words. Timonen (Timonen et al., 2011a; Timonen, 2012) identified differences between categorization of short and long documents. These differences are relevant with keyword extraction also. The most obvious difference comes from the number of words in each document and in the whole corpus. This results in a challenge identified by Timonen as *TF=1 challenge*; i.e., each word occurs only once in a document. Because of this challenge, approaches that rely on the difference between term frequency and document frequency become less effective (Timonen, 2012).

The traditional keyword extraction approaches often rely heavily on term frequency. For example, Term Frequency - Inverse Document Frequency (TF-IDF), KeyGraph from Ohsawa et al. (1998), and a Chi-Squared based approach from Matsuo and Ishizuka (2003) rely on word co-occurrence and word frequencies. All of these studies have focused on longer documents such as news articles or scientific articles. For example, the traditional test set of news articles is the Reuters news archive[2] which contains 160 words per document on average.

[1]In this paper, we use the term keyword extraction to refer both keywords and keyphrases

[2]http://www.daviddlewis.com/resources/testcollections/reuters21578/

Timonen M., Toivanen T., Teng Y., Chen C. and He L..
Informativeness-based Keyword Extraction from Short Documents.
DOI: 10.5220/0004130704110421
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (SSTM-2012), pages 411-421
ISBN: 978-989-8565-29-7

Challenge of extracting keywords from short documents is not a trivial one. The simplest approach may be to extract all the words, or only all the nouns from the document. However, this is rarely beneficial as it results extracting words that are significantly less informative than others. In Section 4 of this paper we show that these irrelevant words reduce significantly the precision of user models and recommendation. In addition, the reduction in time consumption in applications that use the extracted keywords is significant when there are less keywords. For example, when using an approach for domain modeling and query expansion that maps each of the co-occurring keywords together, each additional keyword will greatly increase the time consumption (Timonen et al., 2011b).

In this paper, we propose a novel unsupervised approach for keyword extraction from short documents. The aim of our work is to use the extracted keywords in user models and in a recommendation system. We have based our work on CollabRank keyword extraction approach by Wan and Xiao (2008), and feature weighting in short documents by Timonen et al. (2011a). The idea is to assess the importance (informativeness) of each word of a document on three levels: corpus level, cluster level and document level. In corpus level analysis we want to find words that are informative when considering all the documents. To find words that are informative in a smaller set, we cluster the documents and evaluate the informativeness inside these clusters. The aim is to group similar documents together and find words that are common within the cluster and uncommon outside the cluster. Finally, the informative words within the document are extracted using the results of the two previous levels. We call this approach *Informativeness-based Keyword Extraction* (*IKE*).

We evaluate our approach using three different datasets: movie descriptions, event descriptions, and company descriptions. We use data of three different languages: English, Finnish and Chinese. We manually annotated approximately 300 documents from each dataset and compared the precision and recall of the extracted keywords from our approach against several other keyword extraction methods such as CollabRank, KeyGraph, TF-IDF and Matsuo's Chi-Squared. The problem with this evaluation was that the annotated keywords were subjective to the annotator, which resulted under 70 % agreement rate among annotators. Therefore we conducted another experiment where we used the extracted keywords for user modeling and evaluated the recommendation precision with the models. This evaluation was objective and demonstrates the utilization of the extracted keywords. In all of the experiments our approach outperformed the competition.

This paper makes the following contributions: (1) a novel keyword extraction approach for short documents based on three level word analysis, (2) a novel approach for corpus level word informativeness assessment for short documents, and (3) a comprehensive evaluation of keyword extraction approaches using short documents as the corpus.

This paper is organized as follows: in Section 2 we discuss related approaches. In Section 3 we describe our approach. In Section 4 we perform experimental evaluation and compare the results against other keyword extraction approaches. We conclude the paper in Section 5.

2 RELATED WORK

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 of documents as a training set and learn the features for keywords. The training set is built by manually tagging the documents for keywords.

One approach that uses supervised learning is called Kea (Frank et al., 1999; Witten et al., 1999). It uses Naive Bayes learning with Term Frequency - Inverse Document Frequency (TF-IDF) and normalized term positions as the features. The approach was further developed by Turney (2003) who included keyphrase cohesion as a new feature. One of the latest updates to Kea is done by Nguyen and Kan (2007) who included linguistic information such as section information as features.

Before developing Kea approach, Turney experimented two other approaches: decision tree algorithm C4.5 and an algorithm called GenEx (Turney, 2000). GenEx is an algorithm that has two components: 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. (2001) 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 (Hulth, 2003, 2004). The models use four different attributes: term frequency, collection frequency, relative position of the first occurrence, and Part-of-Speech tags.

Ercan and Cicekli (2007) 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.

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. (1998) that does not use POS-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 (2003) describe an approach that also 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.

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 approach, which is presented by HaCohen-Kerner (2003), uses term frequencies and importance of sentences for extracting the keywords. Later, HaCohen-Kerner et al. (2005) continue the work and include other statistics as well. Andrade and Valencia (1998) use Medline abstracts for extracting protein functions and other biological keywords. Previously mentioned work done by Ercan and Cicekli (2007) also uses abstracts as the corpus. Finally, SemEval-2010 had a task that aimed at extraction of keywords from scientific articles. Kim et al. (2010) presents the findings of the task.

Keyphrase extraction is some times used as a synonym to keyword extraction but it can also differ from it by aiming to extract *n-grams*, i.e., word groups that are in the form of phrases (e.g., ”digital camera”). Yih et al. (2006) present a keyphrase extraction approach for finding keyphrases from web pages. Their approaches is based on document structure and word locations. This approach is directed to keyphrases instead of just words as it aims to identify not just words but word groups. They use a logistic regression for training a classifier for the extraction process. Tomokiyo and Hurst (2003) present a language model based approach for keyphrase extraction. They use pointwise Kullback-Leibler -divergence between language models to assess the informativeness and ”phraseness” of the keyphrases.

Wan and Xiao (2008) 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 keywords are extracted in two levels. First, the words are evaluated in the cluster level using graph-based ranking algorithm similar to PageRank (Page et al., 1998). After this, the words are scored on the document level by summing the cluster level saliency scores. In the cluster level evaluation POS-tags are used to identify suitable candidate keywords; the POS-tags are also used when assessing if the candidate keyphrases are suitable. Wan and Xiao use news articles as their corpus.

Assessing the term informativeness is an important part of keyword extraction. Rennie and Jaakkola (2005) have surveyed term informativeness measures in the context of named entity recognition and concluded that Residual IDF produced the best results for their case. Residual IDF is based on the idea of comparing the word’s observed Inverse Document Frequency (IDF) against predicted IDF ($\widehat{IDF}$) (Clark and Gale, 1995). Predicted IDF is assessed using the term frequency and assuming a random distribution of the term in the documents (Poisson model). If the difference between the two IDF measures is large, the word is informative.

Finally, Timonen et al. (2011a) have studied categorization of short documents. They conclude that the existing approaches used for longer documents do not perform as well with short documents. They propose a feature weighting approach that is designed to produce better results with short documents. We will describe this approach in Section 3 as we base the cluster level word evaluation on their work.

3 KEYWORD EXTRACTION FROM SHORT DOCUMENTS

We have based our approach on the previous works done by Wan and Xiao (2008), and Timonen et al. (2011a). From the former we use the idea of multi-level word assessment through clustering, and from the latter we use the term weighting approach. The term weighting approach described by Timonen et al. is designed for short documents which made it relevant for this case also.

The extraction process has two steps: (1) preprocessing that includes document clustering, and (2) word informativeness evaluation. The latter is divided into three levels of evaluation: corpus level, cluster level and document level, and it aims to identify and extract the most informative words of each level. The input for the process is the set of documents (corpus). The process produces a set of keywords for each document as its output.

3.1 Problem Description

We define a short document as a document that contains no more than 100 words, which is equal to a very short scientific abstract. Depending on the dataset, the documents are often much shorter: for example, Timonen (2012) use a Twitter dataset that holds 15 words on average. In our work, we concentrate on event and movie descriptions that have 30 to 60 words per description. These word counts are considerably less than in corpora previously used in keyword extraction.

The descriptions contain information about events or movies in a concise way. In the case of events, the information consists of type of the event, possible performers, and other information that may be relevant for the reader. The movie descriptions hold information about the movie such as plot, actors, director and genre of the movie.

The aim is to extract the relevant information from the description. The following is an example of an actual event description: ”International contemporary art exhibition from the collection of UPM-Kymmene. The collection focuses on German art from this century. It consists of paintings and drawings from several internationally noted artists such as Markus Lupertz, A.R. Penck and Sigmar.” We want to extract the following words to capture the key information of this event: ”contemporary”, ”German”, ”art”, ”paintings”, ”drawings”, ”Markus Lupertz”, ”A.R. Penck”, ”Sigmar”.

Due to the large variation in content and the arduous task of building a comprehensive training set, we focus our efforts on unsupervised approaches. The biggest challenge with unsupervised learning is the fact that most words occur only once per document. In fact, the more often a word occurs in a document, less informative it usually is. This makes the traditional approaches that rely on term frequency less effective.

During our initial evaluations we noticed that document frequency alone does not find important words. Some informative words, such as the performer of the event may occur only once in the whole corpus. However, some important words such as the event type (e.g., ”rock concert”) may occur often within the corpus. Both of these are important for the document, and for the reader, but this information cannot be captured using only document frequency. In the following sections we propose an approach that takes these challenges and requirements into consideration when extracting keywords from short documents.

3.2 Preprocessing and Clustering

The first task of the extraction is preprocessing. The text needs to be cleaned from noise, which usually consists of uninformative characters, and stop words. For this we use a filter that removes words with less than 3 characters, and a stop word lists. In addition, we need to identify important noun phrases such as names. That is, we group the first and last names together to form phrases. It should be noted that there are freely available named entity recognition software available for English but for Finnish we had to implement our own. For this, we use a naive approach: if two or more consecutive words have a capital letter as its first letter, the words are tagged as a proper noun group (e.g., ”Jack White”). In addition, if there is a connecting word like ’and’ between two words that start with a capital letter, we also tag them (e.g., ”Rock and Roll”). For Chinese, we do not use noun phrase tagging. We do not use other noun phrase identification.

The term evaluation approach that we will describe in the next section requires clustering of the documents. We use Agglomerative (CompleteLink) clustering, which produced the best results for Wan and Xiao (2008). CompleteLink is a bottom-up clustering approach where at the beginning, each document forms its own cluster. The most similar clusters are joined in each iteration until there are at most k clusters. The similarity between the clusters c_n and c_m is the minimum similarity between two documents d_n ($d_n \in c_n$) and d_m ($d_m \in c_m$):

$$sim(c_n, c_m) = \min_{d_n \in c_n, d_m \in c_m} sim(d_n, d_m), \qquad (1)$$

where similarity $sim(d_n, d_m)$ is the cosine similarity of the documents. We use a similar approach but have made a small modification: the most similar clusters are joined if the minimum similarity between documents in the two clusters is above a given threshold t_c. That is, we do not use a predefined number of clusters k but let the cluster count vary among datasets. The algorithm stops when there are no more clusters to be joined, i.e., if there are no clusters with similarity above t_c. We found that this approach performs better in our case. In addition, we use the Inverse Document Frequency to measure the weight of the words before the similarity is calculated as this will make documents that have matching rare words more similar than documents with matching common words. We feel that this does not affect the overall result of the process as this only benefits the clustering by decreasing the impact of more frequent words.

3.3 Word Informativeness Evaluation

We consider that short documents contain two different types of keywords: ones that are more abstract and are therefore more common, and the ones that are more expressive and therefore more rare. The more common keywords usually describe the text as a whole; for example, terms like 'Rock and Roll' and 'Action Movie' define the content of the document in a more abstract level. Words like 'Aerosmith' and 'Rambo' give a more detailed description. Both of these levels are important as without them important information would be missing. This is true especially when the keywords are used by a computer in an automated system instead of presenting them directly to humans.

We address this issue by evaluating the words in two different levels: corpus level and the cluster level. Document level analysis uses the results from the other levels. In corpus level, the informative words are the ones that have an optimal frequency, and in the cluster level the ones that appear often in a single cluster and rarely in other clusters.

3.3.1 Corpus Level Word Evaluation

The aim of the corpus level evaluation is to find words that define the document in a 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' instead of just 'event' or 'music'. Our hypothesis is that the most informative words in the corpus level are those 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 rely on 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. Therefore, document frequency (df) is $df = tf_c$ and, for example, with Residual IDF, observed IDF equals expected IDF. However, even if Residual IDF is not a good option with short documents we use its basic idea: we want to find words that have an IDF close to the assumed optimal value. The greater the difference between the observed IDF and the assumed optimal IDF is, the less informative the word is in the corpus level. This is an inverse assumption that is used in Residual IDF. In addition, we substitute the expected IDF with the expected optimal IDF.

To get the corpus level score $s_{corpus}(t)$ we use an approach we call *Frequency Weighted IDF* (IDF_{FW}). It is based on the idea of updating the observed IDF using *Frequency Weight* (FW):

$$IDF_{FW}(t) = IDF(t) - FW(t), \tag{2}$$

where *FW(t)* is the assumed optimal IDF described below.

The intuition behind FW is to penalize words when the corpus level term frequency does not equal the estimated optimal frequency n_o. Equation 3 shows how FW is calculated: the penalty is calculated as IDF but we use n_o as the document count $|D|$ and tf_c as df. This affects the IDF so that all the term frequencies below n_o will get a positive value, if $tf = n_o$ no penalty will be given, and when $tf > n_o$ the value will be negative. To give penalty on both cases, we need to take the absolute value of the penalty.

$$FW(t) = \alpha \times |\log_2 \frac{tf_c}{n_o}|. \tag{3}$$

Even though FW will be larger with small term frequencies, IDF will be also larger. In fact, when $tf_c = df$ and $tf_c < n_o$, the result $IDF_{FW}(t_1) = IDF_{FW}(t_2)$ even if $tf_{t_1} < tf_{t_2}$ for all $tf_c < n_o$. We use α to overcome this issue and give a small penalty when $tf_c < n_o$; $\alpha = 1.1$ is used in our experiments.

An important part of the equation is the selection of n_o. We use a predefined fraction of the corpus size: $n_o = 0.03 \times |D|$. That is, we consider that a word is optimally important in the corpus level when it occurs in 3 % of documents. This number was decided after empirical evaluation and experimentation with the event dataset, and it has produced good results in all of the experiments. It may be beneficial to change this value when using different datasets, however this number was good in all of our studies.

This approach has two useful features: first, it also considers df in the evaluation. That is, in the rare occasions when $tf_c < n_o$ and $df < tf_c$, these words are emphasized. Second, the IDF_{FW} is considerably smaller when $tf_c > n_o$ than when $tf_c < n_o$, which is preferred as we consider less frequent words more informative than more frequent words. That is, this emphasizes rare words over common words. As this approach has the functionality we require, and as we did not find any existing approaches that fulfill the given requirements, we consider this approach the most suitable option for the corpus level assessment.

3.3.2 Cluster Level Word Evaluation

Next, the word's informativeness is assessed in the cluster level. The idea is to group similar documents together and find words that are important for the group: if the word w appears often in the cluster c and not at all in other clusters ($C \setminus c$), the word is informative in this cluster. Assessing the cluster level informativeness is done by using similar

word informativeness assessment approach as used by Timonen et al. (2011a) where the idea is to assess word's *Term − Corpus Relevance* (*TCoR*) and *Term − Category Relevance* (*TCaR*) as defined below.

In order to assess *TCoR* and *TCaR*, each document is broken into smaller pieces called fragments. The text fragments are extracted from sentences using breaks such as question mark, comma, semicolon and other similar characters. In addition, words such as 'and', 'or', and 'both' are also used as breaks. For example, sentence "Photo display and contemporary art exhibition at the central museum" consists of two fragments, "photo display" and "contemporary art exhibition at the central library".

There are two features used for assessing *TCoR*: *inverse average fragment length (fl)* and *inverse category count (ic)*. Average fragment length is based on the idea that if a word occurs in short text fragments it is more likely to be informative. The average fragment length is calculated simply by taking the average of the word counts within the fragments where the word appears in. Inverse category count is used to give more emphasis on words that appear in a single category. It is calculated by taking the count of the categories (cluster in our case) where the word appears in. For example, if the word occurs in two clusters, *ic* is 0.5. Term-Corpus Relevance is the average of these two values:

$$TCoR(t) = (\frac{1}{avg(l_f(t))} + \frac{1}{c_t}), \quad (4)$$

where $avg(l_f(t))$ is the average length of the text fragment where the word t appears in, and c_t is the count of clusters where the word t appears in.

Term-Category Relevance, which in this case should be called Term-Cluster Relevance, evaluates the word's informativeness among clusters. The idea is to identify words that occur often within the cluster and rarely in other cluster. More often the word occurs within the cluster, and the less clusters the word appears in, higher the *TCaR* score. The score consists of two probabilities: $P(c|t)$, probability that the word t occurs within the category c, and $P(d_t|c)$, probability for the word t within the category c. Former, presented in Equation 5, takes the distribution of word's occurrences among all the categories c:

$$P(c|t) = \frac{|d_{t,c} \in D_c|}{|D_t|}. \quad (5)$$

The probability is calculated simply by taking the number of documents $|d_{t,c}|$ in the cluster's document set D_c ($D_c \in c$) that contain the word t and dividing it with the total number of documents $|D_t|$ where the word t appears in (D_t is the set of documents containing the word t).

The probability for the word within the cluster, shown in Equation 6, takes the distribution of the word t within the cluster:

$$P(d_t|c) = \frac{|d_{t,c} \in D_c|}{|D_c|}, \quad (6)$$

where $|d_{t,c}|$ is the number of documents with the word t within the cluster c and $|D_c|$ is the total number of documents within c. *TCaR*(t,c) for the word t in the cluster c is calculated as follows:

$$TCaR(t,c) = (P(c|t) + P(d_t|c)). \quad (7)$$

The two scores *TCoR* and *TCaR* are combined when the cluster level score is calculated:

$$s_{cluster}(t,c) = TCoR(t) \times TCaR(t,c). \quad (8)$$

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 would appear in two or more clusters. Even though the cluster level word evaluation does some corpus level evaluation as well, we found that the results are often better when using both *TCoR* and *TCaR* instead of only *TCaR*. This is due to the fact that the approaches used here are complementary. More information and the intuition behind the metrics can be found from the paper by Timonen et al. (2011a).

3.3.3 Document Level Word Evaluation

The final step of the process is to extract the keywords from the documents. For finding the most important words of the document, we use the word scores from the previous analysis. The idea is to extract the words that are found informative on either the corpus level or the cluster level; or preferably on both.

Before calculating the document level scores, the corpus level scores are normalized to vary between [0,1]. This makes them comparable with the cluster level scores. The normalization is done by taking the maximum corpus level word score in the document and dividing each score with the maximum value. Equation 9 shows the normalization of s_{corpus}.

$$s_{n,corpus}(t) = \frac{IDF_{FW}(t)}{\max_{t_d \in d} IDF_{FW}(t_d)}. \quad (9)$$

After normalization, the word with the highest $IDF_{FW}(t)$ has the score 1.

The document level score s_{doc} for word t in document d, which belongs to cluster c, is calculated by taking the weighted average of the cluster level score

$s_{n,cluster}(t,c)$ and the normalized corpus level score $s_{n,corpus}(t)$. This is shown in Equation 10:

$$s_{doc}(t,d) = \frac{\beta \times s_{cluster}(t,c) + (1-\beta) \times s_{n,corpus}(t)}{2}, \quad (10)$$

where c is the cluster of the document d, and β indicates the weight that is used for giving more emphasis to either cluster or the corpus level score. To give more emphasis for the cluster level scores, we should use $\beta > 0.5$, and vice versa. We use weighted average for two reasons: we get scores that vary between [0,1], and the effect of a low cluster level or corpus level score is not as drastic as it would be, for example, when the two scores are multiplied. The latter is important as we want also words that score highly on either level and not just on both levels.

As we noticed that keywords occur often in the beginning of the document, we included a *distance* factor $d(t)$ that is based on the same idea used in Kea (Frank et al., 1999). The distance indicates the location of the word from the beginning of the document and it 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.

$$d(t) = 1 - \frac{i(t)}{|d|}, \quad (11)$$

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.

As Part-of-Speech tags are often useful in keyword extraction we use them in the document level scoring as an option. Due to the fact that POS-taggers are not freely available for all the languages we include the POS-tags only as an option, i.e., our approach can easily be used without a POS-tagger. We use the POS-tags as another weighting option for words: different tags get a different POS-weight (w_{POS}) in the final score calculation.

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, $w_{POS}(tag_1) > w_{POS}(tag_2)$ 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 $d(t)$:

$$s(t,d) = s_{doc}(t,d) \times d(t) \times w_{POS}(t), \quad (12)$$

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 most informative words are then assigned as keywords for the document. As some times the number of informative words per document varies, a threshold t_d can be used 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: $t_d = r \times \max s(t,d)$. For example, if $r = 0.5$, the keywords that have a score at least 50 % of the highest score are accepted. However, if there are more than k keywords that fulfill this condition, the top k are selected. We have used $k = 9$ and $r = 0.5$ in our experiments.

4 EVALUATION

We evaluate our approach using three different datasets that we describe in Section 4.1. The test sets were built by manually selecting the keywords that were considered most relevant. We use the datasets in two different types of performance comparisons. We compare the results between IKE and the following methods: CollabRank, KeyGraph, Matsuo's χ^2 measure, TF-IDF and Chi-squared feature weighting. In the first experiment we use the manually picked keywords to see which approach performs the best. In the second experiment we use the extracted keywords to create user models and see which model can produce the best recommendations. We consider the latter experiment the most indicative of performance as it is the most objective.

4.1 Data

To make the experimentation more versatile we use datasets of three completely different languages: Finnish, English and Chinese. Finnish is a complex language with lots of suffixes, Chinese is a simpler language without prefixes and suffixes but a complex language due to its different character set and writing system. English is the standard language in most of the systems.

For Finnish, we use a dataset that consists of approximately 5,000 events from the Helsinki Metropolitan area. The events were collected between 2007 and 2010 from several different data sources. The descriptions hold information about the type of the event and the performers in a concise form. After preprocessing, 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 per document.

For Chinese, we use Velo[3] dataset that 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; even though longer than most of our data this was short enough to be used in our experiments. One of the challenges with Chinese is to tokenize the text into words; for this, paodingjieniu, a Chinese Word segmentation tool was used to divide the descriptions into words separated by blank space.

For English, we use movie abstracts from Wikipedia[4]. This was selected due to its free and easy access. We downloaded approximately 7,000 Wikipedia pages that contain information about different movies. We use MovieLens dataset[5] when selecting the movies: if a movie is found from MovieLens dataset, we download 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.

For the first experiment we created the test set by randomly selecting 300 documents and manually tagging them for keywords. Event and Wikipedia data was tagged by two research scientist from VTT Technical Research Centre of Finland, and Velo data was tagged by two students from East China Normal University. At most nine keywords were chosen per document. The agreement rate among annotators was 69 % for the Event data, 64 % for the Wikipedia data, and 70 % for the Velo data. The test set was updated after disagreements were resolved.

For POS-tagging in English and Chinese we use the Stanford's Log-Linear Part-of-Speech tagger[6]. For POS-tagging in Finnish we use LingSoft's commercial FinTWOL tagger.

4.2 Evaluation of Keyword Precision

We evaluate the feasibility of the extracted keywords using a set of manually annotated keywords. We use all three datasets for evaluation.

4.2.1 Evaluation Setup

We implemented CollabRank algorithm as described by Wan and Xiao (2008). There were some parts that were not clearly described and we therefore made the following assumptions: first, we used window size of 10, as described in the paper. However, the window was not extended over sentence breaks such as full stops and question marks. The candidate words were selected after getting word co-occurrences. That is, the words without appropriate POS-tag were removed after the affinity weights were calculated. This is important as it affects the weights. We used these settings as they produced the best results for CollabRank.

We experimented using both clustering approaches (predefined cluster count and threshold similarity) and found that they produced similar results for CollabRank. However, when using predefined cluster count with IKE, the results were poorer. Therefore, we use the score threshold clustering in all experiments that require clustering.

For Term Frequency - Inverse Document Frequency (TF-IDF) and Chi-Squared we used the standard implementation of the approach. For Ohsawa's KeyGraph (Ohsawa et al., 1998) and Matsuo's χ^2 keyword extraction approach (Matsuo and Ishizuka, 2003) we used Knime[7] and its implementation of the two algorithms. We ran them using the default parameters that were described in the articles. To improve the results we used stop word lists and N char filter ($N = 3$) to remove uninformative words and characters from the documents.

After empirical evaluation, we selected the following parameters for IKE: $\beta = 0.3$, and POS-tag weights $w_{POS}(N) = 1.0$, $w_{POS}(JJ) = 1.0$, $w_{POS}(V) = 0$, $w_{POS}(Others) = 0$. However, when we use event data, we use $w_{POS}(N) = 3.0$ and $\beta = 0.6$. For all the approaches, at most nine keywords per document were extracted.

4.2.2 Results

The baseline result is the F-score received when all nouns and adjectives are extracted. We included adjectives as they are relevant in some domains; for example, adjective *explosive* can be considered relevant in the description "explosive action movie". Therefore, the words with the following POS-tags are extracted: N, A, ADJ, AD, AD-A, -, JJ, NN, NNS, NNP, and NNPS. Some of these tags are used in FinTWOL and some in Stanford POS-tagger. The tag "-" means that also words without a tag, which are usually names not recognized by the tagger, are also extracted. Therefore, the tag "-" is treated as NP in our experiments. This produced the following baselines:

[3]Velo is a company based in Shanghai China that owns and maintains coupon machines.

[4]http://www.wikipedia.org/

[5]http://www.grouplens.org/node/12

[6]http://nlp.stanford.edu/software/tagger.shtml

[7]Konstanz Information Miner: http://www.knime.org/

Table 1: 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 and Ishizuka (2003). Due to the Chinese character set, we were unable to evaluate KeyGraph and Matsuo's χ^2 keyword extraction approach on Velo data using Knime.

	IKE	CollabRank	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

Event data 0.36, Wikipedia data 0.22, and Velo data 0.20.

Table 1 shows the results of our experiments. The best results for both Event and Wikipedia data were received using IKE. For the Chinese Velo data the difference between IKE and CollabRank, and the baseline is great. However, we can see that TF-IDF and χ^2 perform almost as good in this case. After careful review of the results, we conclude that most of the keywords extracted by IKE are feasible even though not originally picked by humans. This is true with all of the datasets. The keywords picked by both χ^2 and TF-IDF were in most cases uncommon, such as the name of the movie. Keywords extracted by IKE were both uncommon and common; for example, name of the movie, actors, and the genre were all extracted.

KeyGraph did not produce good results which was expected. The F-score with Wikipedia data was only 0.22 with precision 0.19 and recall 0.26. Event data produced F-score of 0.36 with precision of 0.31 and recall of 0.42. The reason for poor performance in both cases is the same as with several other approaches: most words occur only once in the document. Due to the fact that KeyGraph uses only a single document, there is not enough information within a short document to make this approach feasible.

Matsuo's χ^2 keyword extraction approach also performed poorly. F-score for Wikipedia data was 0.21 with precision of 0.18 and recall of 0.24. For Events the F-score was 0.35 with the precision of 0.30 and recall of 0.41. The reason for poor performance is the same as KeyGraph: it is impossible to assess the important words from a short document without using the corpus.

4.3 Evaluation of Keyword Utilization for User Modeling

To overcome the subjectivity of the first test set we did a simple experiment where we compare the recommendation precision for each approach. The idea was to see which approach extracts the most useful words from the text, i.e., words that produce the best recommendations. The recommendation precision is assessed by recommending a top-*n* list of movies and comparing how many of them the user has liked.

4.3.1 Evaluation Setup

We use Wikipedia data for keyword extraction and the user ratings from MovieLens data for user modeling and recommendation.

To test the recommendation precision we created a simple user model: first, we randomly selected 10,000 users from the MovieLens dataset. Then for each user, all the movies they rated were retrieved. This set was divided into a training set and a test set with 75 % - 25 % ratio. However, only movies with a positive rating (rating 4.5 or 5) were added to the test set.

For each of the movies in the training set, the keywords were extracted from the Wikipedia page. Each of the keywords were then used as a tag in the user model. The tags were weighted using the user's rating for the movie: for example, if the rating was 3, the tag was assigned a weight of 0, if the rating was 0, the weight was -1.0, and if the rating was 5, the weight was 1.0. If the same keyword is found from several movies, we use the user's average rating among the movies.

To evaluate the model's precision we take the test set and add randomly $k \times 5$ movies from the set of all movies to the test set, where k is the initial size of the test set. That is, if we have 5 movies in the test set, we take randomly 25 movies among all movies the user hasn't seen to make the total size of the test set 30 movies. The recommendation is done by scoring each of the movies: take the keywords of the movie and match them to the user's tags. The score of the movie is the summed weights of the matching tags; for each keyword found from the user model the weight of the tag is added to the score. The top n scoring movies are then put into an descending order and selected as the top-n list of movies. The precision of the user model is the number of user-rated movies in top-n. That is, if the top-n lists consists solely on movies the user has seen and rated highly, the precision is 1.

4.3.2 Results

The baseline used here is the same as before, i.e., all nouns and adjectives are selected and used as keywords. The user model was created as described above. The recommendation precision was calculated

Table 2: Comparison of user models for recommendation when extracted keywords are used for user modeling.

	IKE	CollabRank	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**	**0.30**

by taking the ratio of correct movies in the top-*n* list. To assess the precision of the model, we skipped the movies that did not have any matching tags in the user model. This was done to simulate an actual recommendation system: when assessing the precision, we are only interested in movies that can be linked to the user model.

In some cases, such as with KeyGraph, there was a problem of overspecialization as the approach produced too specific models. In these cases the model was able to do only a very limited number of recommendations. An example of this was James Bond movies: the model consisted solely of keywords like James Bond. Using this model recommendation precision of James Bond movies was high but it could not recommend any other movies. We wanted to emphasize broader models so in addition to precision we include *coverage* to the assessment of performance. Coverage in this case measures the percentage of users which can receive recommendations. The score for the approach is then calculated as *recommendation precision* $\times$ *recommendation coverage*.

Table 2 shows the results of our experiment. When all the nouns and adjectives are used in the user model, the average precision was 0.39, i.e., approximately 2 movies out of 5 were found from the top-5 list. The recommendation coverage for the baseline, i.e., the percentage that shows how many users get recommendations in the test set, was 89 %. This produced the score of 0.30. When using IKE, the precision was 0.55 with the recommendation coverage of 75 %, making the score of 0.41. We consider this result better as the precision is considerably higher and the coverage is good. Finally, CollabRank produced the score of 0.24, TF-IDF 0.25, and χ^2 0.23.

Even though the precision is excellent with Chi-squared and TF-IDF, the poor coverage would make them unusable in a real world setting. However, combining IKE with Chi-squared and/or TF-IDF could be beneficial for user modeling.

These results show that by extracting only the informative words instead of all of them, the results are notably better. In addition, we can see that IKE can extract more useful words for recommendation than CollabRank, TF-IDF and χ^2. The difference in the final score can be credited to the fact that IKE extracts both common and uncommon keywords where as the others focus only on one of them.

5 CONCLUSIONS

In this paper, we have described the challenge of keyword extraction from short documents. We consider a document short when it contains at most 100 words, which is equal to a short abstract. We proposed Informativeness-based Keyword Extraction (IKE) approach for extracting keywords from the short documents. It is based on word evaluation that is done in three levels: corpus level, cluster level and document level. In order to do the evaluation on the cluster level, text clustering is used.

We compared the results against several other keyword extraction approaches. In all of the experiments our approach produced the best results. In addition, we compared effectiveness of the extracted keywords for user modeling and recommendations. In this experiment, the user models created with the keywords using IKE produced considerably better results than any other approach. This is encouraging as it shows the feasibility of Informativeness-based Keyword Extraction for user modeling and recommendation.

In the future, more focus should be given on noun phrase identification as we feel it would benefit summarization and user modeling by extracting more detailed entities from the text. Even though our approach has performed well we believe that there is still room for improvement. We hope that our work can benefit the future research in the field of keyword extraction and text mining from short documents.

ACKNOWLEDGEMENTS

Authors wish to thank the Finnish Funding Agency for Technology and Innovation (TEKES) for funding a part of this research. In addition, the authors wish to thank Prof. Hannu Toivonen and the anonymous reviews for their valuable comments.

REFERENCES

Andrade, M. and Valencia, A. (1998). Automatic extraction of keywords from scientific text: Application to the

knowledge domain of protein families. *Bioinformatics*, 14:600–607.

Clark, K. and Gale, W. (1995). Inverse document frequency (idf): A measure of deviation from poisson. In *Third Workshop on Very Large Corpora*, pages 121–130.

Ercan, G. and Cicekli, I. (2007). Using lexical chains for keyword extraction. *Inf. Process. Manage.*, 43(6):1705–1714.

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction. In Dean, T., editor, *IJCAI'99*, pages 668–673. Morgan Kaufmann.

HaCohen-Kerner, Y. (2003). Automatic extraction of keywords from abstracts. In Palade, V., Howlett, R. J., and Jain, L. C., editors, *KES 2003*, volume 2773 of *Lecture Notes in Computer Science*, pages 843–849. Springer.

HaCohen-Kerner, Y., Gross, Z., and Masa, A. (2005). Automatic extraction and learning of keyphrases from scientific articles. In Gelbukh, A. F., editor, *CICLing 2005*, volume 3406 of *Lecture Notes in Computer Science*, pages 657–669. Springer.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Hulth, A. (2004). Enhancing linguistically oriented automatic keyword extraction. In *North American Human language technology conference*.

Hulth, A., Karlgren, J., Jonsson, A., Boström, H., and Asker, L. (2001). Automatic keyword extraction using domain knowledge. In Gelbukh, A. F., editor, *CICLing'01*, volume 2004 of *Lecture Notes in Computer Science*, pages 472–482. Springer.

Kim, S., Medelyan, O., Kan, M., and Baldwin, T. (2010). Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010*, pages 21–26.

Matsuo, Y. and Ishizuka, M. (2003). Keyword extraction from a single document using word co-occurrence statistical information. In Russell, I. and Haller, S. M., editors, *FLAIRS Conference*, pages 392–396. AAAI Press.

Nguyen, T. D. and Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In Goh, D. H.-L., Cao, T. H., Sølvberg, I., and Rasmussen, E. M., editors, *ICADL*, volume 4822 of *Lecture Notes in Computer Science*, pages 317–326. Springer.

Ohsawa, Y., Benson, N. E., and Yachida, M. (1998). Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *ADL'98*, pages 12–18. IEEE Computer Society.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford.

Rennie, J. D. M. and Jaakkola, T. (2005). Using term informativeness for named entity detection. In Baeza-Yates, R. A., Ziviani, N., Marchionini, G., Moffat, A., and Tait, J., editors, *SIGIR'05*, pages 353–360. ACM.

Timonen, M. (2012). Categorization of very short documents. In *In-press KDIR'12*. SciTePress Digital Library.

Timonen, M., Silvonen, P., and Kasari, M. (2011a). Classification of short documents to categorize consumer opinions. In *ADMA'11*. Online proceedings.

Timonen, M., Silvonen, P., and Kasari, M. (2011b). Modelling a query space using associations. *Frontiers in Artificial Intelligence and Applications*, 255:77–96.

Tomokiyo, T. and Hurst, M. (2003). A language model approach to keyphrase extraction. In *Proceedings of ACL Workshop on Multiword Expressions*.

Turney, P. D. (2000). Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336.

Turney, P. D. (2003). Coherent keyphrase extraction via web mining. In Gottlob, G. and Walsh, T., editors, *IJCAI'03*, pages 434–442. Morgan Kaufmann.

Wan, X. and Xiao, J. (2008). Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In Scott, D. and Uszkoreit, H., editors, *COLING'08*, pages 969–976.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., and Nevill-Manning, C. G. (1999). Kea: Practical automatic keyphrase extraction. *CoRR*, cs.DL/9902007.

Yih, W., Goodman, J., and Carvalho, V. R. (2006). Finding advertising keywords on web pages. In Carr, L., Roure, D. D., Iyengar, A., Goble, C. A., and Dahlin, M., editors, *WWW'06*, pages 213–222. ACM.