UNIVERSITY OF TARTU

Faculty of Mathematics and Computer Science

Institute of Computer Science

Oskar Gross

Finding Non-Trivially Similar Documents from a Large Document Corpus

Master's Thesis (30 EAP)

Supervisors: Sven Laur, D.Sc. (Tech)
Prof. Hannu Toivonen, PhD

Author: ..... "....." May 2011
Supervisor: ..... "....." May 2011
Supervisor: ..... "....." May 2011

Approved for defense:

Professor Jaak Vilo, PhD: ..... "....." May 2011

TARTU 2011

Contents

1 Introduction

1.1 Background

This thesis introduces the methods which are used for measuring the similarity between documents. The document similarity measures are an important topic in information retrieval and in document classification systems. Finding similar documents from a document corpus is applicable in many different fields - web search engines, news aggregation services, advertising systems et cetera. An important aspect for a document similarity measure is, that the human opinion of the similarity should concur with the score of similarity. The problem of semantic similarity arises. The standard way to find similarity between documents is to compare the co-occurrence of words in them. Thus it is possible, that two documents which are contextually very similar, but to dot contain the same words, are marked dissimilar by the standard document similarity measures. The goal of the semantic similarity measures is to take into account the context of the documents and use this information for measuring the similarity.
The goal of this thesis is to first give an overview of different methods which are used for standard and for semantic document similarity. The second goal is to experiment with the document similarity measures on a news portal dataset and to explore whether we can find some interesting properties of those measures.
The motivation for the topic originates from an idea to create a new advertising network which is able to target advertisements better than the networks currently in the market. The goal was to analyse whether we could find a simple, intuitive, yet effective method for finding the non-trivial similarity between documents.

1.2 Structure

We can divide the thesis roughly into two parts. In the first part we are dealing with document similarity measures and theoretical backgrounds for analysing user behaviour on website. We give an overview of different document similarity measures and analyse their performance. We also propose a new method for finding similar documents efficiently from a large document corpus by analysing the associations between concepts in the documents. Although our main focus is to

find non-trivially similar documents from a large document corpus, we also cover the generally used methods for detecting document similarity. We also give an overview of the methods which we are going to use in the case study to analyse user behaviour.
In the second part of our work, we analyse the performance of different similarity measures and user behaviour using the dataset of an Estonian daily news provider Postimees. In the case study one of our goals is to find the best performing measure or combinations of measures for finding similarity between two news stories. Another goal is to understand the user behaviour and to see, whether we can find interesting navigation patterns from the web access logs.
The motivation for analysing user behaviour is that it enables making better news recommendations for user given the browsing history. Combining these results enables the website to offer user behavioural- and content-based advertising on the website. The general approach is to analyse how users interest changes during the session of news browsing.

1.3 Problem statement

First of all it is important to make the distinction between the standard similarity and non-trivial similarity between documents. If we are finding documents similarity by using words which they contain, then it might be that we will have a small similarity score between two documents, though they may actually be contextually very similar. This might be the case when two documents contain different words, but their contextual meaning is more or less the same.
A quite straightforward solution is to find how many frequent words are similar in respective documents. For the non-trivial similarity we are trying to find documents which are similar on more abstract level. Let us illustrate this with a small toy example. Consider a query document A and document corpus B, C and D. Consider a similarity score $s(x, y)$ which represents the similarity between these two documents. Suppose that the contents of the documents is as given in the following table.

A	*Coffee is a brewed drink prepared from roasted seeds, called coffee beans, of the coffee plant.*
B	*Tea is the agricultural product of the leaves, leaf buds, and internodes of various cultivars and sub-varieties of the Camellia sinensis plant, processed and cured using various methods.*
C	*Jack and John love to drink a cup of coffee. Jack knows everything about coffee - from growing the coffee plant to roasting the seeds and brewing the coffee.*
D	*Peter and Mary love to drink a cup of tea every morning. They are even growing their own Camellia sinensis plants in their backyard.*

For a standard document similarity matching the most similar document to A would definitely be C, as they have many words in common ('coffee', 'seed', 'brew', 'drink' et cetera). We see that all the documents are similar, but when using standard similarity scores, it is probable that $s(A, B)$ would score less than we would intuitively think, as these two documents do not contain many words in common, though their context is similar. Our goal is to give a significantly high score for $s(A, B)$ and $s(A, C)$, as these documents are contextually very similar.
The second problem which we are dealing with, is user behaviour on news website. Now, again, let us make a toy example to illustrate what knowledge we are trying to mine from the sequential document reading. The content of the documents is taken from New York Times headlines. Consider, that user visits documents in the following sequence $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$. Suppose that the documents contain the information:

$\mathbf{A_1}$	*Libya Wages Counterattack on Rebels, and Battles Rage.*
$\mathbf{A_2}$	*U.S. Freezes a Record $30 Billion in Libyan Assets.*
$\mathbf{A_3}$	*In Libya Capital, Long Bread Lines and Barricades.*
$\mathbf{A_4}$	*Egypt Reopens Museums and Historical Sites.*

It is not hard to find the similarity between the first three documents $A_1 \rightarrow A_3$. On the other hand, for traditional term based similarity measure we would give a very low score for similarity between A_4 and other documents. For the non-trivial document similarity, we try to see the associations between these documents, which

may have lead the reader from $A_3 \rightarrow A_4$. Intuitively we can think, that the user was reading document A_3 and that reminded him or her about the riot in Egypt and lead to the article A_4. Our goal here would be to get a significant score for A_1 and A_4).

2 Life-Cycle of Finding Document Similarity

In the following we will give an overview of the steps needed for finding the similar documents. In brief, the steps are:

1. Defining similarity measure.
2. Extracting important content of the document.
3. Document preprocessing.
4. Storing the features of documents.
5. Computing document similarity matrix.
6. Clustering/Analysing performance of different measures.

As follows, we are going to give a quick overview of these steps putting more emphasis on Step 1.

2.1 Extracting Content

Let us consider a system where we need to aggregate information from various internet web pages. When we are dealing with just a couple of different websites, it is not hard to create custom content extraction parsers, but extracting content dynamically from differently structured HTML documents is not a trivial task. A good example of such setting might be a news surveillance system, where we get news stories from very many different news providers in HTML format. In this case we need to somehow dynamically filter out the noise (i.e advertisements, comments, embedded videos, social networking boxes etc) and extract the news story.

There are various approaches for content extraction: statistical, information theoretical, using structural analysis [LH02, CYWM03] et cetera. One tool for content extraction is Readability - developed by a software company Arc90. There are implementations of Readability's algorithm in different programming languages available in the World Wide Web [ABC+09].

2.2 Document Preprocessing

In the following our goal is not to go into technical details of specific methods but to give an overview. We will cover different ways for preprocessing documents.

Named Entity Recognition. Named entity recognition (NER) is a technique for finding different names (e.g. locations, persons, dates et cetera) from natural text. In information retrieval the concept of using named entity recognition and its effectiveness in queries is for instance discussed in [GXCL09]. While preprocessing we are concatenating the named entities with an underscore. For the sake of clarity, let us consider a sentence

Steve Jobs is the co-founder and chief executive officer of Apple Inc

then after preprocessing we get

Steve_Jobs is the co-founder and chief_executive_officer of Apple_Inc.

Named entity recognition is a very good way for reducing noise and extracting useful features from document data. The problem with NER is that usually it is computationally expensive and thus the preprocessing of documents takes longer.

Filtering Stop Words. The term stop words which describes the most frequently used words was proposed by Luhn in 1958 [Luh58]. Stop words are the most commonly used words and they are frequently filtered out from text in information retrieval tasks. When removing the stop words we get rid of noise and we also save space when storing documents. Although, filtering out stop words may in some cases cause information loss, e.g. the band name *The Who*, a citation from Hamlet *To be or not to be* et cetera. One way to reduce the information loss is by removing the stopwords *after* identifying the named entities.

Lemmatization and Stemming. The difference between stemming and lemmatization is that stemming is a heuristic process of cutting off the ends of words in the hope of getting correct lemma for most of the cases, whereas lemmatization

uses the vocabulary and morphological analysis of words to get the base or dictionary form of the word. There are various word stemmers [Por80] available and for English text one of the most used is the Porter stemmer.
In information retrieval, stemming and also lemmatization both lower the precision and increase the recall of a query. Lemmatization and stemming can be used to make the level of the word meaning more abstract.

Normalization. The main point of normalization is to convert all terms which mean the same thing, but can be written in different forms (e.g. *USA* and *U.S.A*) into the same form. In the normalization we are using the following techniques:

- Remove all punctuations in the named entities.
- Lowercase all words in the beginning of the sentence.
- Remove all special characters in the text.

These methods are good because of the following aspects:

- They are easy to apply.
- They handle most of the cases.
- They are language independent.

More technical details about text normalization are given in the handbook [MRS08].

2.3 Document similarity

There are different ways for finding similarity between documents. We will give an overview of some of the popular methods used for finding the document similarity score.

2.3.1 Notation

Let $D\{d_1, d_2, ..., d_n\}$ be a set of documents (*document corpus*). Then we use $S_i \subset d_i$ as a set of sentences of the document d_i. We treat each sentence $s_j \in S_i$ as a sequence of terms $s_j = (t_1, ..., t_m)$. By the length of document d we denote $|d|$ as

the number of words in this document.
For a sentence S_i a term pair $T_p = \{t_1, t_2\} \subset S_i$ is a subset of the sentence S_i. Let ws denote the size of a window. Let us denote the term pair of a window

$$T_p(ws) = \{t_j, t_k\} \subset S_i : |k - j| \leq ws \ ,$$

as all pairwise combinations of words, which belong to sentence S_i and are at most ws words apart from each other.

2.3.2 Similarity Measures

In the following we will present measures which are commonly used for calculating the similarity between documents.

Document-Term Vector. Before going into details of similarity measures, let us denote the document-term vector as:

$$dt(d) = (w_1, w_2, ..., w_n)$$

where w_i is the weight of a term t_i in document d. If the term does not exists in this document, its weight is 0. There are a total of n weights where n is the total number of different terms in the document corpus. Already for a small corpora these vectors are sparse, so in terms of memory consumption, it is reasonable to store only the weights which are greater than zero.

Tf-Idf Measure. Tf-Idf measure is a popular method for extracting relevant terms from documents. Let $n_t(d)$ denote the number of occurrences of given term t in document d. Then the *term frequency* for term t and document d is given as:

$$\mathsf{tf}(t, d) = \frac{n_t(d)}{|d|} \ . \tag{1}$$

A natural method for extracting important terms from a document is to reverse order the terms by their frequency. The drawback of simply measuring the importance by frequency is the fact that, there are words which appear frequently in

many documents (e.g. 'many', 'going', 'meeting', 'yourself' etc). To overcome this issue, we can measure the inverse document frequency idf:

$$\mathsf{idf}(t) = \log \frac{|D|}{|\{D_j : D_j \in D \wedge t \in D_j\}|} \ , \tag{2}$$

where the numerator is the number of documents in the corpus and the denominator is the number of documents which contain the term t. The inverse document frequency measures the "rareness" of the term with regards to the corpora. If the term occurs in every document, then $\mathsf{idf}(t) = 0$ and if the document appears in only 1 document, then $\mathsf{idf}(t) = \log(D)$, which is maximal.
We get the tf-idf value by computing the product between tf and idf:

$$\mathsf{tf\text{-}idf}(t, d) = \mathsf{tf}(t, d) \cdot \mathsf{idf}(t) \ . \tag{3}$$

It gives the highest value, when the term is frequent in the document and not frequent in the rest of the corpus. Thus, the higher the tf-idf, the more document specific the terms are. For more information and explanations about tf-idf can be found in the handbook [MRS08].

2.3.3 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors by measuring the angle between them. For documents the cosine similarity measure is used to find the angle between two document-term vectors. The angle between these vectors shows how similar these documents are - the range of values for this measure is between 0 and 1, where 0 means that the two vectors are orthogonal and 1 means that the two vectors point to the same direction. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the cosine similarity between them is defined as:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \ ,$$

where $\cdot$ denotes the dot product between vectors x and $\mathbf{y}$, and $\|\mathbf{x}\|$ is the Euclidean norm of the vector $\mathbf{x}$.

2.3.4 Latent Semantic Indexing

Latent semantic indexing (LSI) is a technique which uses singular value decomposition (SVD) to find patterns in the relationships between terms or concepts in unstructured text. The goal of latent semantic indexing is to incorporate more semantic structure into query of document as individual terms provide unreliable evidence about the context and meaning of the document. In general the idea is to use SVD for generating "semantic" space, where these terms and documents which are closely associated are near to each other [DDL$^+$90]. LSI overcomes some of the biggest problems of Boolean keyword queries - the issue with synonymy and polysemy. The synonymy are different words which mean the same thing and plolysemy are the words which have several different meanings. It is fundamentally important, as people use surprisingly great variety of words to refer to the same thing [FLGD87].
For any matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ it is possible to express $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4}$$

where $\mathbf{U}$ is $m \times m$ real or complex unitary matrix, $\mathbf{V}^T$ is $n \times n$ real or complex unitary matrix and $\mathbf{\Sigma}$ is $m \times n$ diagonal matrix with non-negative real numbers. The values on the diagonal of $\mathbf{\Sigma}$ are known as singular values of $\mathbf{M}$ and by convention are constructed to be ordered in decreasing magnitude.
Let us consider a table A which has m rows, n columns and contains information of the occurrences of m different terms in n documents. Initially, the constructed matrix holds the term frequencies with respect to certain term and document. It is also possible to apply different global and local weighting functions to the matrix. The singular value matrix $\mathbf{\Sigma}$ can be seen as mapping between $\mathbf{U}$ and $\mathbf{V^T}$.
The main idea of LSI is the dimension reduction

$$\mathbf{M} \approx \hat{\mathbf{M}} = \mathbf{U}'\mathbf{\Sigma}'\left(\mathbf{V}'\right)^T \quad , \tag{5}$$

where $k \ll n$, matrices $\mathbf{U}'$, $\mathbf{\Sigma}'$ and $\mathbf{V}'^T$ are $m \times k, k \times k$ and $k \times n$ dimensional, respectively. The dimensionality reduction of the singular matrix $\mathbf{\Sigma}$ is the key for mapping together the words which are associated with the same concepts. The

exact number of dimensions to choose is still a research problem, but values from 200-300 has shown good results on datasets [Bra08], so we will use 250 as the golden average. By the reduction we are creating a new space, where $\mathbf{U}'$ and $\mathbf{V}'^T$ describe the terms and documents, respectively, in this space. The singular values $\mathbf{\Sigma}'$ describe the amount of variation along the axis in this space. In matrix $\mathbf{U}'$ the rows define a vector for every term which describes the terms relation to the reduced concepts space and the columns represent the strongest concepts which were extracted by singular value decomposition. In the matrix $\mathbf{V}'^T$ the columns represent the documents in the reduced semantic space and each row shows the weights of the documents in the reduced space. Term similarity scores produced by LSI are high for words which have similar meaning, so the higher the score, the more synonymous words are.
For queries the resulting matrices are used as follows. Consider a query $\mathbf{q}$ which is a m dimensional vector, containing the weights of terms of the query. Then we can represent the vector in the reduced space as follows:

$$\hat{\mathbf{q}} = \mathbf{q}^T \mathbf{U}' \mathbf{\Sigma}'^{-1} \quad .$$

Now the query vector can be compared to all the other document vectors given in $\mathbf{V}'$ by using cosine or some other similarity measure.
In depth analysis and examples of using LSI in terms of intelligent information retrieval is given in [BDO+95].

3 Similarity and Concept Associations

Finding similar documents using the concept associations consists of the following steps:

1. Preprocessing of the documents.
2. Extracting the background graph.
3. Extracting specific features of a document.
4. Calculating pairwise distances.
5. Retrieving the most similar documents.

There are different possibilities for finding features from document. The features of a specific document may be the words, combinations of words or any other data which is derived from the specific document. Usually the extracted features of a document are words and a popular method is to store them in a vector. Then the whole document corpus can be held in a document-term matrix, where each row corresponds to a term and each column corresponds to a document. The limitations of this method is, that some specific co-occurrence information is lost, as the document is represented as a vector $\mathbf{v}$

$$\mathbf{v} = (v_1, v_2, ..., v_n) \ ,$$

where v_i is the weight (e.g. `tf-idf`, frequency) of respective term in the document. As described before, there are methods which make an assumption that words which appear often together are most probably semantically similar. Our goal is to represent these connections explicitly and generate them using the document corpora. In general, our assumption is that words in one sentence are more strongly related than words in different sentences.
More formally we propose a method for describing the document corpora as a background graph $G = (V, E)$, where V is a set of vertices belonging to G and E is the set of edges which connect two vertices by a certain weight. The vertices in the graph represent concepts given in the corpora. The weighted edge

between two vertices describes how strongly these two concepts are generally connected. Such a graph gives us opportunity to apply different graph operations. For instance we can find shortest path from arbitrary vertex v_x to vertex v_y, which describes the most commonly made logical association path between these concepts. As an illustrative example, consider two concepts *'synthetic rubber'* and *'car'*. A logical connection between these concepts would be for instance *'synthetic rubber'*→*'tire'*→*'car'*, meaning that they are connected to each other via concept *'tire'*.

The goal of creating such graph is to model the different domains of the document corpus. For more certain example, consider a document corpora of news stories. Our goal is to model both (a) the associations between different news stories in certain category; and (b) the associations between different categories.

The *concept graph* gives us a possibility to add domain specific information to document. In information retrieval, this might be useful for query expansion, as the query may contain too specific information. Using the background graph gives the opportunity to involve more general information to query-document matching.

The following method we are proposing is a generalisation of finding document similarity by using a phrase indexing graph model proposed by Hammouda and Kamel [HK04]. The idea of phrase indexing graph model is to store phrases found in documents into a graph and then find document similarity by analysing the paths shared by documents. In proposed method, the weighted combination of phrase similarity and single-term similarity was used.

3.1 Similarity with Background Information

In the following we are proposing a method for finding similar documents by using similar concept associations in the documents. We will describe how to find the similarity between documents by using background information of the domain. The background information is represented as graph and our goal is to use general associations between words to detect the non-trivial similarity between the documents.

Document preprocessing. Preprocessing of documents leads to less noise and computational time during the next steps. This step takes unprocessed HTML files as input and produces output which consists of sentences, separated by specific character pattern (e.g. *". "*). For removing some highly probable noise we will have some constraints for terms (a) terms may contain only characters $A - Z$; and (b) terms must be at least 3 characters long. The more in-depth discussion of document preprocessing was in Section 2.2.

Extract background graph of the documents. The method of background graph extraction uses the preprocessed documents and list of term pairs with scores to produce a graph which incorporates background information over the whole document corpora. As we will see later in more detail, we can create the pairs between the terms which are in a certain sized window or between terms which appear in the same sentence. For storing the graph, we used the trivial graph format, where the first two columns defined the nodes and the third column defined the edge weight.

Extract features specific to a document. Our goal is to extract the features which are specific to a certain document. We present two approaches: extracting terms by traditional tf-idf measure and extracting terms by using document specific concept associations [SM83].

Calculate pairwise distances of documents. The goal of this method is to calculate the distances between all the documents. For each of the document pairs, this method takes the document-term vectors obtained in the previous step as an input. Depending on the similarity measures, these vectors are used to calculate the document similarity value between two documents. Whether the background graph is incorporated in the similarity calculations depends on the specific measure.

Retrieve the most similar documents. Consider that we have a document corpus D as defined before. Then the document similarity vector obtained from

previous step for a document d is

$$S(d) = (s(d, d_1), ..., s(d, d_n)) \ ,$$

where $s(x, y)$ is a document similarity measure. Different methods for retrieving similar documents are available: (a) taking top-N most similar documents; (b) retrieving documents for which similarity is above some pre-defined threshold; and (c) clustering documents and treating a cluster of documents as related documents.

Error sensitivity of all steps. One of the most significant part of the methods is document preprocessing. This is a good way for removing noise in the beginning of the cycle. Good preprocessing of documents leads to less-noisy background graph and reduces the computational time.
For the background graph extraction, it is reasonable to test different window size parameters - the idea here can be related to the generalization of the model. If the window size is very small, this leads to overfitting and if the window size is too big, the model becomes too general.
The feature extraction step of specific document generally depends on the preprocessing and these methods are not very sensitive.
There are different parameters which we can tune when calculating the similarity between two documents. Definitely this step is error prone to unreasonable parameter selection.
There might arise a question, what is the need for the background information graph. Due to the fact that we are extracting document specific associations, the probability that two documents share a specific association is not very high. On the other hand, if the documents are similar, they are close to each other in the background graph with high probability. The idea is illustrated in Figure 1.

3.2 Extracting Background Information

The idea of the background graph is to model the associations between concepts. Moreover, our goal is to emphasize connections between these concepts which are common (e.g. association between 'car'-'tachometer' is more likely than 'airplane'-

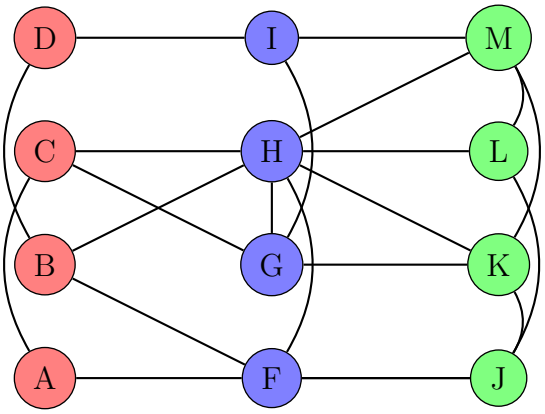


Figure 1: The illustration of two documents mapped to the background graph. The blue nodes represent the nodes on the background graph. Green and red nodes describe the important terms given in two different documents. The documents can be also visually distinguish, as the left "column" represents one document, the central "column" represents the background graph and the right "column" the second document. Though in this extreme case the documents do not share any common terms, they may still might be contextually very similar.

'tachometer' et cetera) and thus represent the background information.

The *de-facto* standard in information retrieval community for analysing connections between terms in documents is log-likelihood ratio test, which is a parametric statistical test [Dun93]. In terms of the background graph, we want associations which depend on each other to have higher weights than the associations which describe weaker dependence between the concepts.

There are two commonly used alternatives to test the independence between two events - the χ^2 and the log-likelihood ratio test. The idea of log-likelihood ratio test is to express how many times more likely the data is from one model than from the alternative model. In our setting, we are using the multinomial distribution. In the ratio, for the *null* model we expect that the two events are independent and for the alternative model is the two events to be dependent. We can create a parallel with p-value, which measures the probability that we would get results at least that *extreme* as observed assuming that the *null* model holds. In the log-likelihood ratio, we do something similar - we take the ratio between the concrete model and null model, basically testing how much more likely is the parametrized

model than the null model.

The χ^2-test is a test which is used to analyse the co-occurrence of two events. The problem with the χ^2-test is that it is based on several assumptions, which frequently do not hold in textual analysis. It is explained in more depth in the article [Dun93].

Let us consider the two potentially related terms or events A and B. Then we can observe the contingency table K

	A	$\neg A$
B	k_{11}	k_{12}
$\neg B$	k_{21}	k_{22}

where k_{11} is the count of events where A and B occur together, k_{12} is the count of events where B occurs, but not A, k_{21} is the count of events where A occurs, but not B, k_{22} is the count of events where neither one of the events A or B occur.

Assume that we know the $K = (k_{11}, k_{12}, k_{21}, k_{22})$ for two terms t_i and t_j. Let us denote a likelihood function $L(\omega; k)$, where ω is a configuration of concrete model from parameter space Ω and k denotes the observations. The likelihood function describes the probability of experimental outcome of k from a model with parameters ω. Then we can define ratio

$$\lambda = \frac{\max_{\omega \in \Omega} L(\omega; k)}{\max_{\omega \in \Omega_0} L(\omega; k)},$$

where Ω_0 denotes the null model parameter space and Ω denotes the concrete parameters of hypothesis which are being tested. For our case the null hypothesis is that events A and B occur independently and the alternative model is that they are statistically dependent. So we can state our null hypothesis Hyp_0 that A and B are independent, so $p(A|B) = p(A|\neg B) = p(A)$ and thus

	A	$\neg A$
B	$q_1 q_2$	$q_2(1 - q_1)$
$\neg B$	$q_1(1 - q_2)$	$(1 - q_1)(1 - q_2)$

where

$$\mathbf{q} = (q_1, q_2).$$

Hyp_1 is that two events A and B are not independent thus we have

	A	$\neg A$
B	p_{11}	p_{12}
$\neg B$	p_{21}	p_{22}

where

$$P_i = p_{1i}, p_{2i} \ ,$$

are the parameters of the model which describe the probabilities for each of the k_{ij} events. Let us denote

$$K_i = k_{1i}, k_{2i} \ ,$$

which contains the elements of the ith row of the contingency table. Now we can give the likelihood ratio

$$\lambda = \frac{\max_Q L\left(Q, Q; K_1, K_2\right)}{\max_{P_1,P_2} L\left(P_1, P_2; K_1, K_2\right)} \ , \tag{6}$$

where the function L is the parametrized multinomial distribution

$$L(P_1, P_2; K_1, K_2) = \binom{k_{11} + k_{12} + k_{21} + k_{22}}{k_{11}, k_{12}, k_{21}, k_{22}} p_{11}^{k_{11}} p_{12}^{k_{12}} p_{21}^{k_{21}} p_{22}^{k_{22}} \ . \tag{7}$$

Note that

$$L(Q, Q; K_1, K_2) = \binom{k_{11} + k_{12} + k_{21} + k_{22}}{k_{11}, k_{12}, k_{21}, k_{22}} (q_1 q_2)^{k_{11}} [q_2(1-q_1)]^{k_{12}} [q_1(1-q_2)]^{k_{21}} [(1-q_1)(1-q_2)]^{k_{22}} \ .$$

Observe that the multinomial coefficients cancel out and after optimization we get

$$\lambda = 2 \cdot \sum_{k \in K} k \cdot (H(K) - H(\{k_{11} + k_{12}, k_{21} + k_{22}\}) - H(\{k_{11} + k_{21}, k_{12} + k_{22}\})) \ , \tag{8}$$

where $H(X)$ is the sum of Shannon entropies

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)) \ . \tag{9}$$

Using the measures and previously defined document corpus D, we can generate the background graph. The generation of the background graph contains two important steps:

- We pair the words which fit in a certain window or occur in the same sentence.
- We calculate the log-likelihood measure for any extracted pair.

When creating term pairs, we need to define a window or some other constraint, in which we combine the words pairwise. Assuming that words co-occurring in the same sentence are related makes it reasonable to pair words within sentences. A property of the log-likelihood measure is, that when storing the data wisely, we can calculate the edges of the weights only when we need them.
When calculating log-likelihood values we need to decide, whether we want to calculate the term co-occurrence in terms of sentences, documents or some other units. This means, that we have to create new contingency table K with respect to chosen unit. The case study shows that in practice using the sentence level measure the graph is more similar to our goal than when using document level measure. In more detail this can be read in Section 5.5.
Given the document corpus $D = \{d_1, ..., d_n\}$. Let us define the two neighbour sets

$$N_S(t) = \{t_i : \{t_i, t\} \in s_j\},$$
$$N_D(t) = \{d_i : t \in d_j\},$$

where the set N_S contains all terms with the parent sentence which contains term t and the set N_D contains all the parent documents of term t. Using these sets give us convenient way to calculate the values $k_{11}, k_{12}, k_{21}, k_{22}$ for the log-likelihood measure. Let us consider that we want to calculate the log-likelihood for term pairs t_i and t_j. We can express the values of k_{ij} as follows:

$$k_{11} = |N_D(t_i) \cap N_D(t_j)|$$
$$k_{12} = |N_D(t_i) \setminus N_D(t_j)|$$
$$k_{21} = |N_D(t_j) \setminus N_D(t_i)|$$
$$k_{22} = |N_D \setminus (N_D(t_i) \cup N_D(t_j))|.$$

3.3 Tpf-Idf-Tpu Measure

When generating background information we weigh the co-occurring term pairs by taking into account the global occurrences of terms. For finding document specific associations we propose using the tpf-idf-tpu measure, where tpf stands for *term pair frequency*, tpu stands for *term pair uncorrelation* [Hyn10] and idf is *inverse document frequency*. We want to use the measure for extracting associations between concepts which are specific to a certain document. The value of the measure should represent the novelty of the association with respect to the document corpus. When using these term pairs as document features when calculating similarity between two documents, we try to pair documents which create associations between similar concepts.
As follows, we will give an overview of the different components which give us an opportunity to score these defined term pairs.

3.3.1 Term Pair Frequency and Inverse Document Frequency

Let us consider document d, then tpf is defined as the relative frequency of sentences which contain term pair T_p:

$$\mathsf{tpf}\,(T_p, d) = \frac{|\{s \in d : T_p \subset s\}|}{|\{s \in d\}|} \ .$$

The inverse document frequency idf of term pair T_p is the logarithm of the inverse of the relative number of documents in the given collection C that contain both terms in the same sentence:

$$\mathsf{idf}\,(T_p) = \log \frac{|C|}{|\{d \in C : \exists s \in d : T_p \subset s\}|} \ .$$

By finding the product of tpf and idf we can define the tpf-idf measure:

$$\mathsf{tpf\text{-}idf}(T_p, d) = \mathsf{tpf}(T_p, d) \cdot \mathsf{idf}(T_p) \ ,$$

which scores high these term pairs which are frequent in the specific document and not frequent in the whole document collection.

3.3.2 Term Pair Uncorrelation

Considering a term pair T_p it is probable that the term pair is *not novel* and *not interesting* if it satisfies one of the following conditions:

1. Term t_1 occurs almost always with term t_2.
2. Terms t_1 and t_2 occur in the same set of documents.
3. Occurrence of term t_1 in different documents is very high.
4. Occurrence of term t_2 in different documents is extremely low.

The goal of the tpu measure is to give lower score of these term pairs, for which the elements satisfy these conditions. Let $r(v|u), r(u|v)$ to denote the relative amounts of a term pair $T_p = (u, v)$, such that:

$$r(v|u) = \frac{|\{d \in D : \exists s \in d : u, v \subset s\}|}{|\{d \in D | v \in d\}|} \ ,$$
$$r(u|v) = \frac{|\{d \in D : \exists s \in d : u, v \subset s\}|}{|\{d \in D | u \in d\}|} \ .$$

Now we can define tpu:

$$\mathsf{tpu}\,(T_p) = \gamma - \max(r(u|v), r(v|u)),$$

where $\gamma \geq 1$ is used to weight the importance of the tpu component.
To illustrate how the measure works, in Table 1 we can see term pairs which had a high or low scores on the *Postimees* corpora. The first row represents the condition where one of the terms almost always co-occurs with the other. As *Marju Länik* is Estonian singer and the term a is *laulja (singer)*, due to this, almost always when the term b occurs also term a occurs. On the second row we see the case where the number of documents the term b occurs is very small. On the third row we see an interesting example of two words *hitt_pakkett (hit package)* and *lisa_au_hind (extra award)* which are words which occur roughly in the same set of documents. For the fourth example, the term a occurs almost always with the term b.
As follows the terms in the table, for which the condition is marked as `N/A` are

Term a	Term b	Tpu Score	r(a|b)	r(b|a)	Condition
laulja	länikult	1.0	0.0	1.0	1
eeter	pabermärkmed	1.0	0.001	1.0	4
hitt_pakett	lisa_au_hind	1.0	1.0	0.5	2
vahipataljonis	üksik	1.0	1.0	0.01	1
kultuuri_maja	tipp_hetk	1.99	0.01	0.01	N/A
muusika	prantslane	2.0	0.0	0.0	N/A
erakonna_kaaslane	kontsert	2.0	0.0	0.0	N/A
järjekord	kostüümide	1.98	0.0	0.02	N/A

Table 1: Examples of term pairs with high and low tpu scores. The first four examples are term pairs with low tpu scores and the last four are examples with high tpu score.

term pairs which have scored high on tpu. Most of these words are frequently used, but rarely occur together. And interesting example is the third positive example where term a is *erakonna_kaaslane (political party companion)* and term b is *kontsert (concert)*, which rarely occur in the same documents.

3.3.3 Tpf-Idf-Tpu

Using the results we can now define the tpf-idf-tpu measure:

$$\text{tpf-idf-tpu}\left(T_p\right) = \text{tpf}\left(T_p\right)^{w_1} \cdot \text{idf}\left(T_p\right)^{w_2} \cdot \text{tpu}\left(T_p\right)^{w_3} \quad , \tag{10}$$

where $w = \{w_1, w_2, w_3\}$ is a weight vector initialized at $\mathbf{1}$. The goal of this measure is to describe the novelty and interestingness of given term pair.

3.4 Finding Document Similarity

Consider we have the background information graph which is generated as described in 3.2. In the following we will give different measures for finding the similarity between two documents. The measures are later validated in the case study, where we benchmark the methods in finding similar documents. Our general idea is to see how far the two documents are from each other in terms of the background information graph.

3.4.1 Formulation

Let us give a mathematical formulation of the problem. Consider a document corpus C and two documents $D_1 \in C$ and $D_2 \in C$. We are given three graphs $B = (V, E, W)$, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where V, V_i are the set of vertices, E, E_i is the set of edges, $W = \{w_i | w_i \in \mathbb{R}, i \in \mathbb{N}\}$ is the set of association weights, B is the background information graph and G_1, G_2 are the important associations graphs for documents D_1 and D_2, respectively.
Our goal is to determine the distance between these two documents by using the document graphs G_1 and G_2 and the background graph.

3.4.2 Average Distance

One way for calculating the distance between two sub-graphs is to find the average shortest path for each node in the background graph. So we can define the distance between the documents D_1 and D_2

$$d(D_1, D_2) = \frac{1}{|V_1||V_2|} \sum_{u \in V_1,\ v \in V_2} \mathsf{shortest\text{-}path}(u, v) \ ,$$

where shortest path is found from every vertex in V_1 to *any* vertex in V_2.
The problem with the proposed measure is, that it does not take into account the weights of the edges. It would be reasonable to take into account the association strength between terms. By now we have used edge weights for which larger weight refers to a stronger connection than lower weight. For applying common shortest

path algorithms, we reverse the edge weights by

$$W = \left\{ \frac{1}{w_i}, i \in \{1...n\} \right\} \ .$$

3.4.3 Neighbourhood Similarity

Let us consider a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. The complexity of finding the shortest path between two vertices in G is $\Theta(|V|^3)$. So it makes it reasonable to analyse the similarity of the expansion of G_1 and G_2 in terms of the background graph. By expansion we mean, that we get the set of vertices $N_1(G_1)$ by *walking* away n steps from all the nodes of sub-graph G_1. In other terms we expand from the sub-graph and include extra vertices from the background B. Mathematically we can formulate this as:

$$N_1(W) = \{v \in V : \exists u \in W : (u, v) \in E\} \ , \tag{11}$$

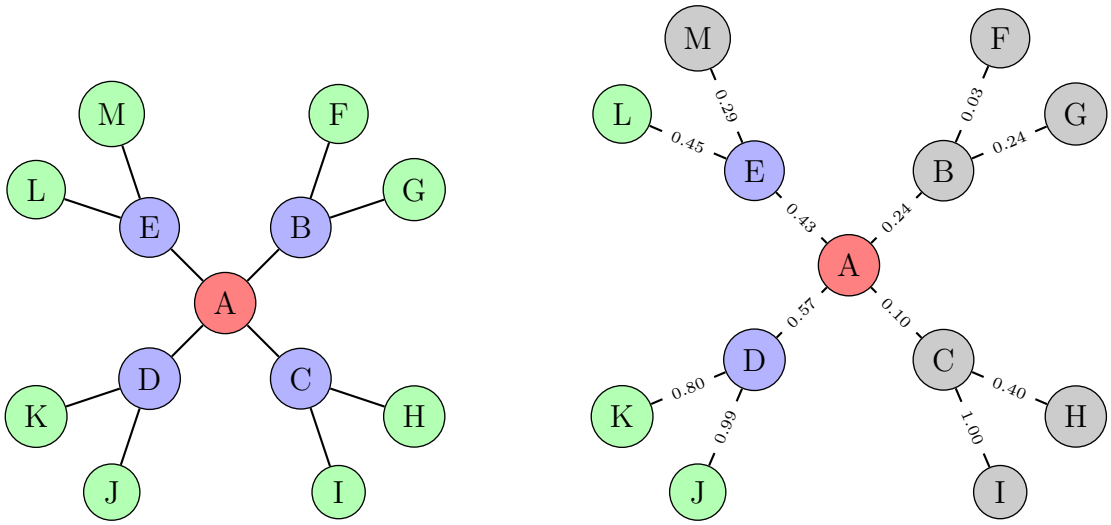
$$N_i(W) = \{v \in V : \exists u \in N_{i-1}(W) : (u, v) \in E\} \ .$$

We can iterate in breath-first manner by first finding the union and then applying the formula again. As an example, let us consider we want the two step neighbourhood of documents D_1 and D_2. First we apply (11) getting $N_1(D_1)$ and $N_1(D_2)$. Then by combining $N_1(D_1) \cup V_1$ and $N_1(D_2) \cup V_2$ and applying the respective formulas again, we get the two step neighbourhood et cetera. The illustration of the neighbour expansion can be seen on Figure 2a. Now the similarity score between document D_1 and D_2 can be calculated as the Jaccard coefficient

$$J(D_1, D_2) = \frac{|N_1(D_1) \cap N_1(D_2)|}{|N_1(D_1) \cup N_1(D_2)|} \ , \tag{12}$$

which is the relative overlap of the neighbourhoods of the two documents.
Observe, that in the measure we are not using the originally overlapping terms between documents. Let us denote the neighbourhood with features similarity measure

$$J_F(D_1, D_2) = \frac{|(N_1(D_1) \cap N_1(D_2)) \cup (D_1 \cap D_2)|}{|(N_1(D_1) \cup N_1(D_2)) \cup (D_1 \cup D_2)|} \ , \tag{13}$$



(a) Not weighted expansion

(b) Weighted expansion

Figure 2: Consider document D contains only term A. Then the blue nodes are $N_1(D)$ and the green nodes are $N_2(D)$. The gray nodes denote the nodes which are not included to the $N_i(D)$ due to the fact that the edge weight is less than $\varepsilon = 0.30$.

which also takes into account the overlap of the features which were originally in the documents D_1 and D_2.

Proposed similarity measures suffer under the problem that we actually expand in all *directions* of the graph, which means, that two documents may be close to each other, but if we have nodes with very high degree, the similarity score is tampered. This is due to the fact, that we may have a word which occurs together with very many words, but has very low log-likelihood ratio with most of the terms. When setting threshold for the minimum edge weight, we expect stronger connections between words, and thus eliminate a large part of the noise.

We can overcome the problem by defining the weighted neighbourhood similarity score for which the expansion rule is

$$N_1(W, \varepsilon) = \{v \in V : \exists u \in W : (u, v) \in E, W_{(u,v)} \geq \varepsilon\} \ ,$$
$$N_i(W, \varepsilon) = \{v \in V : \exists u \in N_i(W) : (u, v) \in E, N_i(W)_{(u,v)} \geq \varepsilon\} \ .$$

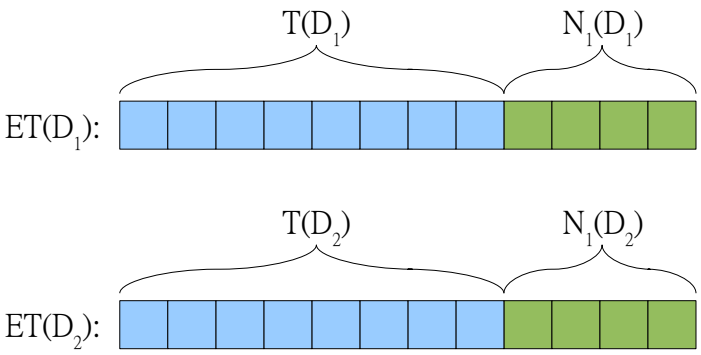


Figure 3: Extending the document feature vectors from background, where $T(D_i)$ are the original features of the document and $N_1(D_i)$ is the corresponding one step neighbourhood.

The illustration for this expansion rule can be found in Figure 2b. Notice that in this case we are using the original edge weights, not the reversed ones.

3.4.4 Extended Cosine Measure

Another way for comparing documents using the background graph and cosine measure, is to first append context information to documents from the background graph G for which we have normalized weights to the range $[0, 1]$. Consider that we want to find similarity between document D_1 and D_2. First we extract *top-n* tf-idf terms with their weights $T(D_1), T(D_2)$ from D_1 and D_2. Now our goal is to add some contextual information to documents D_1 and D_2 from the background graph G. Thus we take all the first level neighbours of $N_1(D_1)$ and $N_1(D_2)$ from graph G and add as features to the documents D_1 and D_2, where the weights of the added terms are the weights of the connecting edges. The idea is illustrated on Figure 3. Then we calculate the cosine similarity measure between these two extended document feature vectors as given before in subsection 2.3.3.

3.4.5 Voltage Distance

The motivation for the voltage distance measure is to give weight to the neighbouring nodes which are shared by two documents. The idea is a simplified version of graph interpretation as electrical networks and center-piece sub-graphs [FMT04, TF06]. We will represent a graph $G = (V, E)$ as an electrical network, where the weight of an edge e is given as $C(e)$. In the electrical network interpretation edge e represents a resistor with conductance $C(e)$.
Consider that we apply voltage of $+x$ to a node s, and ground (0 volts) on the node t. Let $I(u, v)$ denote the current flow from u to v and let $V(u)$ denote the voltage at node u. As follows, we have two laws, Ohm's law:

$$\forall u, v : I(u, v) = C(u, v)(V(u) - V(v)) = C(u, v)V(u) - C(u, v)V(v) \ ,$$

and Kirchhoff's current law:

$$\forall \neq s, t : \sum_u I(u, v) = 0 \ .$$

It is easy to see, that by combining these laws, we will get a linear system which solution determines all the voltages and currents:

$$V(u) = \sum_v \frac{C(u, v)V(v)}{C(u)} \ ,$$

where $C(u) = \sum_v C(u, v)$ is the total conductance of the edges which are adjacent to node u. The only exceptions are $V(s) = x$ and $V(t) = 0$. It is also proposed [PF03] to use the universal sink node z, which is grounded $V(z) = 0$ and is connected to every node u in the graph G, such that it's conductance is given as

$$C(u, z) = \alpha \sum_{w \neq z} C(u, w) \ ,$$

where $\alpha > 0$. We follow [FMT04] and use $\alpha = 1$. The idea is that the universal sink penalizes the high degree nodes, by absorbing current, which flows through them and through their neighbours. Now we can solve the system of linear equations

using the least squares solver which gives us the voltages for all the nodes.
The given method can be also applied for many source and ground nodes, than we just have to take s and t as sets of nodes and do the calculations based on that information. Using these voltages we can calculate the sum over the voltages of all the shared neighbours of documents D_1 and D_2

$$\sum V(v_i), v_i \in N_1(D_1) \cap N_1(D_2) \ ,$$

which we will use as the similarity measure.

3.4.6 Fail Distance

Fail distance measure got his name due to the authors oversimplification of the concept of voltage distance measure. The approach which we are going to introduce intuitively handles the high or low degree node problems - we give less weight to neighbour which comes from a very high degree node and on the other hand more weight to the neighbour which comes from a low degree node. In addition to the node degree we could also take into account the edge weights between the nodes. For achieving this we treat the graph as a network, where we can give weights to the nodes and then edges behave as resistors when transferring the weight from one node to another. Before going into details, assume that for the background graph $G = (V, E, W)$ we have the edge weights as their inverse $W = \{\frac{1}{w_i} : w_i \in W_1\}$. As given before, the vertices which belong to two documents D_1 and D_2 are the sets V_1 and V_2, respectively. Let us consider the set of shared neighbours $N_s = N_1 \cap N_2$. Consider the weight of a vertex $v_{ij} \in V_i$ as ϑ_{ij} and the weights of the neighbours N_s to be defined as

$$\Gamma = \{\gamma_1, ..., \gamma_n\} \ ,$$

where $n = |N_s|$. For normalizing the weight of a node by its degree, let us denote

$$\vartheta'_{ij} = \frac{\vartheta_{ij}}{|v_i j \in E_i|} \ ,$$

which is the weight given to any neighbour of v_{ij}. As we treat the edge weights as resistors, the transferred weight over the edge from one node to its neighbour γ_k

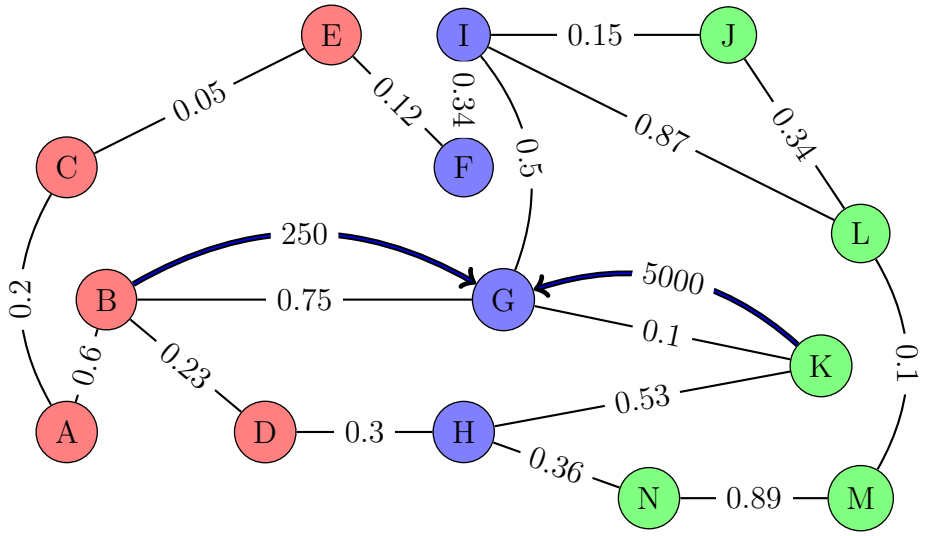


Figure 4: The illustration of the fail distance measure. The weights given to node G are illustrated by the blue edges. The incoming weights are summed giving us the total weight.

is given as:

$$\gamma_k = \sum_{\vartheta'_{ij}:(v_{ij},n_k)\in e_{ij}} \frac{\vartheta'_{ij}}{w_l} \ , \tag{14}$$

where n_k is the neighbouring node and w_l is the weight of the edge between the nodes v_{ij} and n_k. For an example consider weighted graph as illustrated on Figure 4. Let us calculate the final weight of node G, when we set the weight of 1000 to any neighbouring node. Node G is connected to nodes B and K and I. Giving 1000 units to K, we distribute the weight evenly between all the adjacent edges, which means that each edge gets 500 units. The edge between K and G has resistance of 0.1, thus the amount of weight coming from K is $I_K = \frac{500}{0.1} = 5000$ units. By the same logic we get weight from B, which is then $I_B = \frac{1000}{3} \cdot \frac{3}{4} = 250$ units. Totalling, we get that the weight on G is $I = I_K + I_B = 5250$ units.

4 User Behaviour Analysis

Analysing user behaviour on websites is a good way for obtaining implicit information of user preferences. Due to the information overload, user behaviour analysis for recommending web content which might be useful seems reasonable idea. Some machine learning approaches for user behaviour analysis for news services can be found in [LL02, SK97, BP00]. Though, large part of these approaches is focused on which information to extract from user behaviour - scrolling, reading time and other actions which user is performing on the website.
We are mainly interested in sequential browsing of news stories and our goal is to see whether there is correlation between the news similarity scores and the sequential browsing of news stories.

4.1 Scores

As we are looking for sequential patterns from data, we will define the problem in terms of sequence mining. Let us consider a set of transactions $T = \{t_1, t_2, ..., t_n\}$ where every transaction is a sequence of events $t_i = \langle e_1 e_2 ... e_m \rangle$. In our case two-event sequence $s' = \langle xy \rangle$ is a subsequence of transaction t_i in case $\langle xy \rangle \in t_i$ or $\langle yx \rangle \in t_i$, which are transactions where x and y occur consecutively.
Let us denote the function cover:

$$\mathsf{cover}(\langle xy \rangle) = \{s : \langle xy \rangle \in s \vee \langle yx \rangle \in s\} \ ,$$

which is the set of sentences which contain elements x and y side by side. Then we can define the support count

$$\mathsf{support}(\langle xy \rangle) = |\mathsf{cover}(\langle xy \rangle)| \ ,$$

which is the number of elements that $\langle xy \rangle$ covers. Let us denote a frequency which is relative to the number of transactions which contain two events x and y

$$\mathsf{relative\text{-}support}(\langle xy \rangle) = \frac{\mathsf{support}(\langle xy \rangle)}{|\{t \in T : x \subset T \vee y \subset T\}|} \ . \tag{15}$$

Let us denote the relative-support for one element sequence with regards to the sequence $\langle xy \rangle$

$$\mathsf{relative\text{-}support}(\langle x \rangle, \langle xy \rangle) = \frac{\mathsf{support}(\langle x \rangle)}{|T|} \ , \tag{16}$$

which is the support of event x normalized by the total number of transactions where sequence $\langle xy \rangle$ occurs. The score (15) can be used to measure the connection strength between two news stories. In some sense we can think of it as the Jaccard measure for two events – how often two events occur together divided by the number of all occurrences of either event x or y.
Another score which we can calculate is the interest factor:

$$\mathsf{interest\text{-}factor}(\langle xy \rangle) = \frac{\mathsf{relative\text{-}support}(\langle xy \rangle)}{\mathsf{relative\text{-}support}(\langle x \rangle) \cdot \mathsf{relative\text{-}support}(\langle y \rangle)},$$

which compares the frequency of the co-occurrence of the events against frequency which is computed under the statistical independence assumption. A nice overview of the interest factor and many other measures is given in the handbook [Han05].
It is important to note here that we can use these measures for two sequentially occurring news stories, but we can also define a `maxgap` constraint, which allows gaps between the occurrences of news stories x and y. Let us define the cover function with `maxgap` constraint:

$$\mathsf{cover}(\langle xy \rangle, \mathsf{maxgap}) = \{t_k : e_i = x \in t_k, e_j = y \in t_k \wedge |i - j| < \mathsf{maxgap}\} \ ,$$

which we can use in the support and interest-factor calculations.
The initial setting is the special case for `maxgap` being equal to 2. Setting `maxgap` equal to the length of transaction, we count x and y to be a subsequence of t_i if x and y appear together in transaction t_i.
In the case study, we will use these measures to find news stories which are related to each other considering users browsing behaviour.

5 Case Study

In the case study, we will give an overview of the practical part of our work. As follows we first give an insight into the problems which we are solving, then we introduce the dataset and discuss various aspects in data preprocessing. We will also give a methodology for benchmarking different similarity measures and a way to analyse the correspondence between the behaviour of users and the similarity measures.

5.1 Problem Statement

Our goal is to analyse the methods for finding similar documents which were proposed in the theoretical part. In addition to accuracy we also analyse whether it is possible to use proposed methods in real-world applications. Our interest in the measures is two-sided - on one hand we want the measures to give high results in accordance with human decision, on the other hand we would also like the measures to detect interesting underlying connections between documents where the similarity is not trivial. The motivation for the first aspect is not hard to see - methods can be used in information retrieval systems, news recommender engines, topic detection et cetera.

The second part of our case study analyses user behaviour on a website. Our goal is to see whether there are frequent patterns which occur in the browsing sessions and how do they relate to the similarity scores. This is valuable information for the news provider in many ways. For instance it enables the website to make news recommendations for a user in order to extend the browsing session or it gives a possibility to avoid publishing news stories which may bring along shorter sessions. We will also analyse how the terms connected in the background graph model are similar to human opinion. We will use two approaches for analysing this: (1) we cluster the graph and see whether the clusters contain words which appear in similar contexts; (2) we perform analysis on human-selected keywords.

5.2 Preliminary Operations

First, we give the preliminary operations which we have to do before calculating the similarity between documents. The goal is not to go over the theoretical part, but to give step-by step overview what decisions we did and why.

Data. For obtaining data we scraped the popular Estonian news website *Postimees.ee*. In total we scraped 71279 news stories. The scraping process itself was trivial - we saved the whole web page HTML on the hard disk, extracted all the links from the web page which referred to another news story and moved around on the web page in the breadth first manner. We implemented the crawler by ourself, using the *htmlunit* [htm] library

Data Preprocessing. From the HTML documents we extracted the title and contents of the news stories by using regular expression parser. As the morphological parsers use the structure of the document, the next step was to detect the word types in the document. Our text corpora was in Estonian, so we used the Estonian Morphological Analyser (ESTMA) [Kaa97]. This is an important step of data preprocessing as it has strong influence on the quality of the features we are going to extract later.
Given the text with morphological tags, we need to decide which words we will keep and which we will remove from the text. We used two different settings: (1) we leave only nouns, foreign words and names; (2) we leave nouns, verbs, foreign words and names. We decided to use these features, as intuitively these word forms explain a large part of the variation of the textual content. After extracting the words, we merged them together into original sentences.

5.2.1 Extracting Document Features

We used two document feature extraction methods- tf-idf keywords and tpf-idf-tpu associations. We generated two files, where for every document we stored the tf-idf keywords into one file and the tpf-idf-tpu concept associations to the other file. As

ROBIN JUHKENTAL

NIMI JA VANUS:
Robin Juhkental (19)
ELUKOHT:
Harjumaa, Tallinn
PRAEGUNE KOOL VÕI AMET:
Hetkel olen muiduleivasööja.
VARASEM OSALEMINE KONKURSSIDEL JA TULEMUS:
Kunagi (vist kolmandas klassis) sai osaletud "Laulukarusellil". Televooru sain, aga tõenäoliselt maksis rohkema saavutamise asjaolu, et olin seal veel mitu korda rohkem närvis kui viimase "Kahe takti" saate salvestusel.
KAS OLED LAULMIST ÕPPINUD, KUS JA KELLE JUHENDAMISEL:
Ei ole kunagi laulmist õppinud
MIKS OSALED SAATES "KAKS TAKTI ETTE":
Konkreetset põhjust polegi, aga tagantjärgi ütleks, et pigem hea kogemuse pärast kui mingi meeletu võidusoovi või telepurki saamise nimel.
LEMMIKLAULJA/EESKUJU EESTIST:
Hetkel ei näe ma Eestis kedagi Vaiko Eplikule võrdset, eriti just kõiges selles, mis puudutab heliloomet. Aga väga hästi on laulnud veel Urmas Alender, Tõnis Mägi, Jaak Joala ja Ivo linna. Naislauljatest on minu jaoks kõige nauditavam Liisi Koiksoni laulmine.
LEMMIKLAULJA/EESKUJU VÄLISMAALT:
Neid on palju. Suurimad eeskujud on Paul McCartney, John Lennon, Robert Plant, Paul Rodgers, Joe Cocker, Sam Cooke, Brian Wilson.
MILLIST MUUSIKAŽANRIT EELISTAD:
On juhtunud nii, et hetkel laulan peamiselt vanakooli rocki, aga üldiselt meeldivad ka blues, funk, jazz ja ka hästitehtud popmuusika.
MIS ON SINU ELUS HETKEL KÕIGE TÄHTSAM:
Siia peaks vastama, et muusika on kõige tähtsam? Muusikata elada oleks raske, aga selline vastus kõlaks minu arvates klišeelikult.
MILLEST UNISTAD, EESMÄRGID:
Minu jaoks on muusik see, kes elatub muusika tegemisest. Tore oleks kui saaks kunagi ka ennast muusikuks pidada.
SINU PARIM OMADUS:
Võib-olla võime kohaneda eri stiilides, aga tegelikult saaks objektiivsema vastuse, kui seda küsida kelleltki teiselt.

Figure 5: An interview with Estonian singer Robin Juhkental. A sample news story from the Postimees corpus.

the bag of words model is easy to extract from the original document we did not create specific bag of words file. As an example consider a short interview article with an Estonian singer Robin Juhkental which can be found in Figure 5. The top 5 most highly scored features of the tf-idf and tpf-idf-tpu can be seen in Table 2. This is quite an interesting example and conceptually the tf-idf keywords give the important aspects of the news story as the terms are important with regards to the context. It is a bit harder to interpret the term pairs of the document, though in general these associations may be considered important in the document.

Tf-idf	Score	Tpf-idf-tpu	Score
lemmik_laulja	0.17	(muusika, tähtsam)	4.67
laulmine	0.16	(laulmine, õppinud)	4.00
ees_kuju	0.13	(kõige, muusika)	3.54
muusika	0.11	(ees_kuju, paul)	2.79
muiduleivasööja	0.11	(rask, tähtsam)	2.79

Table 2: Frequent keywords and term pairs for article about an Estonian singer Robin Juhkental.

5.2.2 Background Graph Generation

Different constraints can be used when we generate the background graph. There are some properties of the graph which come due to the method of generation. In the following we will show some results of the background graph which we have found to be present also for other document corpora.

In general, the background graph is usually fully connected or has very few connected components. By graph filtering we mean, that we remove some edges lower than some fixed threshold. On the Figure 7, we can see four different components which have been extracted after filtering the background graph with *threshold* of 200. All but one of the connected components represent different contexts, one of them (in the bottom right corner) is a false positive which connects different contexts by last name 'jaanson'.

One interesting aspect to see, what the background graph looks like around a certain concept. First we removed all the edges from the graph which have weight lower than 40 (the threshold was chosen by traditional method of trial and error). Then we selected a term `kesk_erakond` (Estonian Centre Party), which is the centrist social liberal party in Estonia and extracted the neighbourhood such that we kept only neighbours, which have edges connected to other neighbours. The results can be seen on Figure 6. Intuitively we see, that these concepts are really very closely related to the Center Party - `savi_saar` is the leader of the party, Savisaar is also the mayor (node `linna_pea`) of Tallinn et cetera. As we can see, this gives a good motivation for trying graph clustering. To see, whether the whole graph is similar to the Centre Party example, we clustered the large background graph

Clusters			
1	**2**	**3**	**4**
sool	vene	meri	paha_pill
maitsesta	väeüksused	sõjalaevade	meelehärmi
keedu_sink	dmitir	ületamiseks	mikk
köömnetega	senakist	naftatööstusesse	kümnevõistlejate
väherasvast	phothist	kaspia	kõrgushüppes
pipar	tõmbumist	energia_ressurss	tõkkeid
muskaatpähklit	soomuki_kolonn	nafta_juhe	olümpianormi
oliivõli	sõjaväebaas	semjonov	täitmise
riivitud	territooriumilt	sõjalaevastiku	tõukab
laimimahla	väeüksuseid	tshinvalisse	meistrivõistluste

Table 3: Table which shows the words which are clustered together by MCL clustering algorithm.

using MCL [Don00, mcl] clustering algorithm with *inflation* parameter of 4.5. The nodes which were put into one cluster can be seen in Table 3. We can observe very strong relations between the words which belong to the same cluster. The first cluster gives words for cooking, the second contains Russian military operation in Georgia, the third cluster contains information about oil and energy (most probably related to Nord Stream) and the third cluster contains concepts from sports. Our goal was to model the most common connections between concepts, and as we we see the background graph does that quite well.

5.2.3 Related News Stories

In addition to the news stories, we also extracted additional information from most of the scraped websites. Hand curated databases are valuable if we want to benchmark methods which should behave similar to humans. For *Postimees.ee*, there are an average 3.2 related news stories per one news story. Thus, for each of the downloaded news stories we also extracted the related news stories which were hand-picked by editors. By this we got pairs of related news stories.
An important thing to see here is that there is a temporal aspect in the related

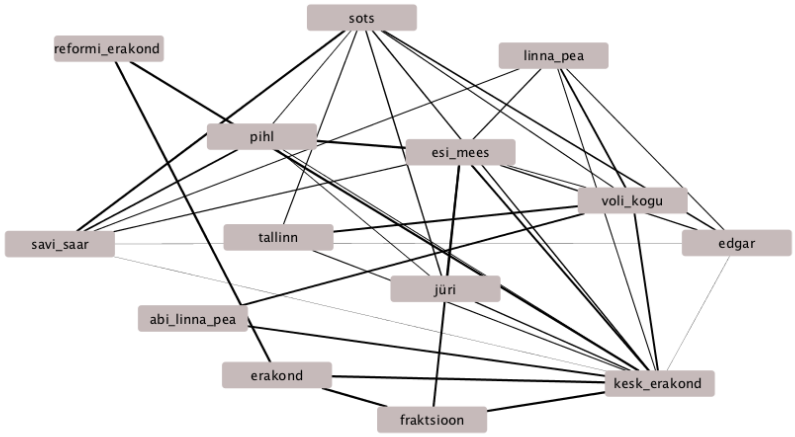


Figure 6: A neighbourhood of the term kesk_erakond. The boldness of the edge refers to the strength of the connections. Interesting is that connection between `savi_saar` and `kesk_erakond` is not too strong.

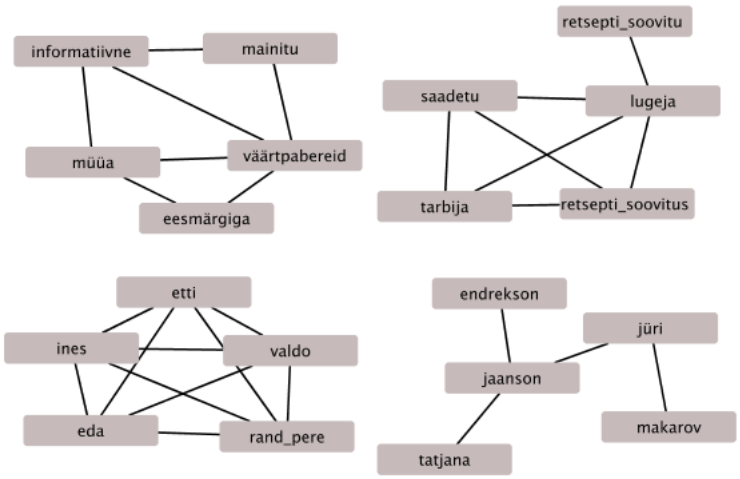


Figure 7: A sample of the background graph, where all the nodes with a weight smaller than 200 are deleted. This illustrates how words which appear in different context form different strongly connected components.

news stories - on the web page the older news stories are not connected with the later published stories. This is one aspect which has to be taken into account, when dealing with the news associations.

5.2.4 Sampling News Stories

For comparing the similarity measures of the news stories, we used the following methodology. We created pools of the related and unrelated news stories. In each of these pools we have 30000 news story pairs, where the paired news stories are related or unrelated, respectively. Our main experiment model is, that we take random samples from both pools and then make comparisons of the methods based on these subsets. As an example, consider we take n samples from the news story pools. Notice that this means that we have randomly selected n news story pairs not n news stories.
There are some experiments for which we use another sampling strategy. We selected a random news story d from the whole news story dataset. Then we extracted all the related to n unrelated news stories with regards to d. After this sampling we calculated the similarity measures, stored and repeated it m times.

5.2.5 Method Parameters

There are parameters which influence the methods when calculating the similarity between documents. The main tunable parameters are the `topN` keywords or term pairs features we are using for each document and the other parameter `expEdges` defines how many high scored edges we include for each expanding node. The `expEdges` parameter can be thought of as a dynamic expansion parameter, which measures the edges' importance with regards to a node, not by an absolute value. Another tunable parameter `bgThresh` is the threshold for the edge weights in the background graph. By using this parameter, it gives a way to remove nodes and edges which are too weakly connected to the rest of the graph, i.e., removes noise. In the following we will give a method which we used to get some approximation which of the parameters would be reasonable to use.

Receiver Operating Characteristic Curve. A common methods for visualizing the goodness of classifiers are the Receiver Operating Characteristic curves which are also known as ROC curves. The ROC curve illustrates the ratio between the false positive and true positive (hit) rates. For a random classifier these values are both equal. For a good classifier we expect it to have high true positive and low false positive rate.
Another measure of goodness for classifiers is the Area Under the (ROC) Curve (AUC). As the AUC is is a portion of an area of unit square it's value is always between 0 and 1. The AUC value for a diagonal line, which represents the random classifier is 0.5. If classifier has AUC less than 0.5, then it classifies instances in the opposite way, which means that the classifications can be reversed. The interesting statistical property of the AUC value is, that it is equal to the probability that the classifier will rank a random positive instance higher than randomly chosen negative instance [Faw06].
The R package *verification* gives a convenient framework for drawing ROC curves [Pro10]. It gives a possibility to plot false alarm rate against the true positive rate for a probabilistic forecast for a range of thresholds.

Finding Optimal Background Graph. First we need to find the optimal threshold for the background graph. It seems reasonable to use the measure which measures the document similarity by the overlap of the neighbourhoods as by this we see which of the tuned background graphs contains the least noise. Considering document features and expansion to the background we are also interested in the `topN` and `expEdges` parameters. For achieving this we conducted the following experiment:

1. Initialize parameters `topN`, `expEdges` and `bgThresh`.
2. Select 100 news stories.
3. For every news story extract related stories and 100 unrelated news stories.
4. Calculate neighbourhood overlap similarity for all configurations of parameters.
5. Find the AUC scores for the measure under configuration.

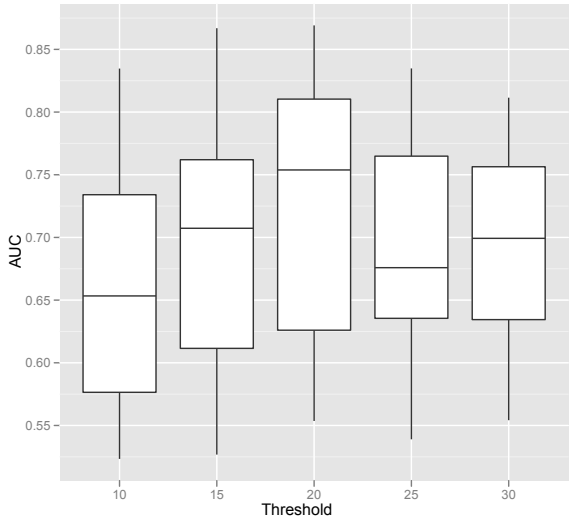
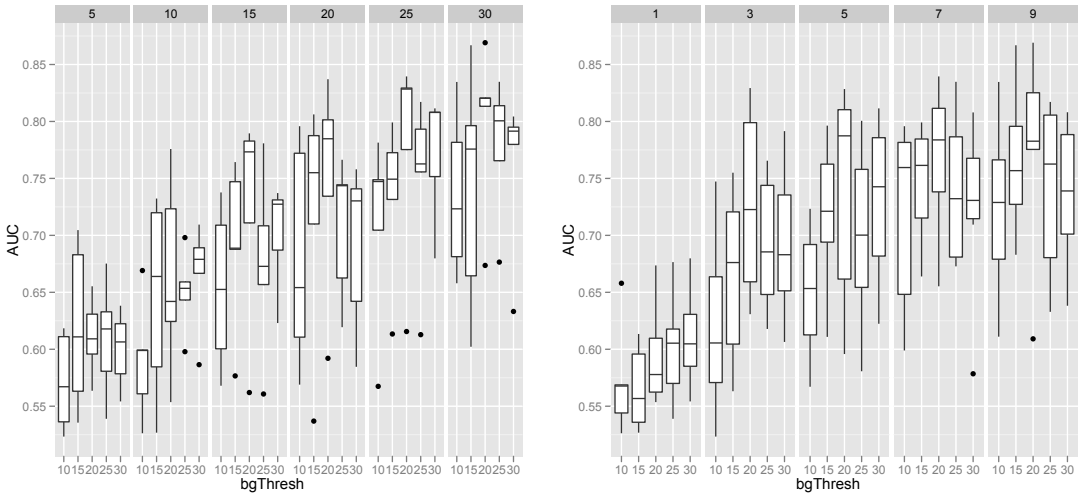


Figure 8: The AUC values for different `bgThresh` thresholds.

6. Select the best configuration.

At first let us find the best configuration of threshold of the background graph. As follows we are going to analyse the AUC scores for different parameters. The idea is to fix the parameters `bgThresh`, `expEdges` and `topN` by analysing the performance of the neighbourhood similarity measure under different parameter combinations. On Figure 8 we can see the AUC distributions of experiments for which the `bgThresh` was $10, 15, 20, 25, 30$, respectively. We can see, that the threshold of 10 stands out with quite low AUC mean score, thus is reasonable to discard it from the further analysis. Significantly low AUC scores by Wilcoxon test we saw for $topN < 15$ (p-value $= 9.9 \cdot 10^{-6}$) and `expEdges` < 3 (p-value $= 4.6 \cdot 10^{-6}$), which means that we can also discard these parameters from further analysis. The next step is to analyse the AUC values given `bgThresh` and `topN`. The distribution of the AUC scores can be seen on the Figure 9a. As we can see the highest score is achieved by the configuration where `bgThresh` $= 20$ and `topN` $= 25$. On Figure 9b we can see the distributions of AUC scores between `bgThresh` and `expEdges`. The best scored configuration from the plot would be where `bgThresh` $= 20$ and `expEdges` $= 7$ or `expEdges` $= 9$. The Figure 10 shows the relationship between the `expEdges` and `topN` parameters wrt. `bgThresh`. Again we see, that good scores



(a) `bgThresh` and `topN`.

(b) `bgThresh` and `expEdges`.

Figure 9: Figures which represent the distribution of AUC values with respect to constraints given under the subfigures.

are achieved with the parameters we pointed out. Thus, though a robust way for finding the optimal configuration later we decided to use the following constraints: `bgThresh` = 20, `topN` = 25 and `expEdges` = 7. We could have also chosen `topN` = 30 and `expEdges` = 9, but due to the very small difference in AUC scores we decided to use 20 and 7, respectively, as it is computationally less expensive.

5.3 Similarity Score Experiments

In the following, we will analyse the behaviour of the different similarity measures. The purpose of these experiments is to see whether there exists a correlation between human opinion and the similarity measure. Moreover, we are interested if the similarity measure is higher for these news stories which are annotated by an editor as related news. The scores we are going to analyse are as follows: `cosine`, `extended cosine`, `neighbourhood`, `shortest path`, `neighbourhood with features`, `fail distance` and `voltage based` similarity measure. In the `cosine` calculation we take the terms with the tf-idf weights and calculate the angle between two documents. The `extended cosine` means, that we have included neighbourhood information into the `cosine` calculation. The `neighbourhood` measure is the ratio

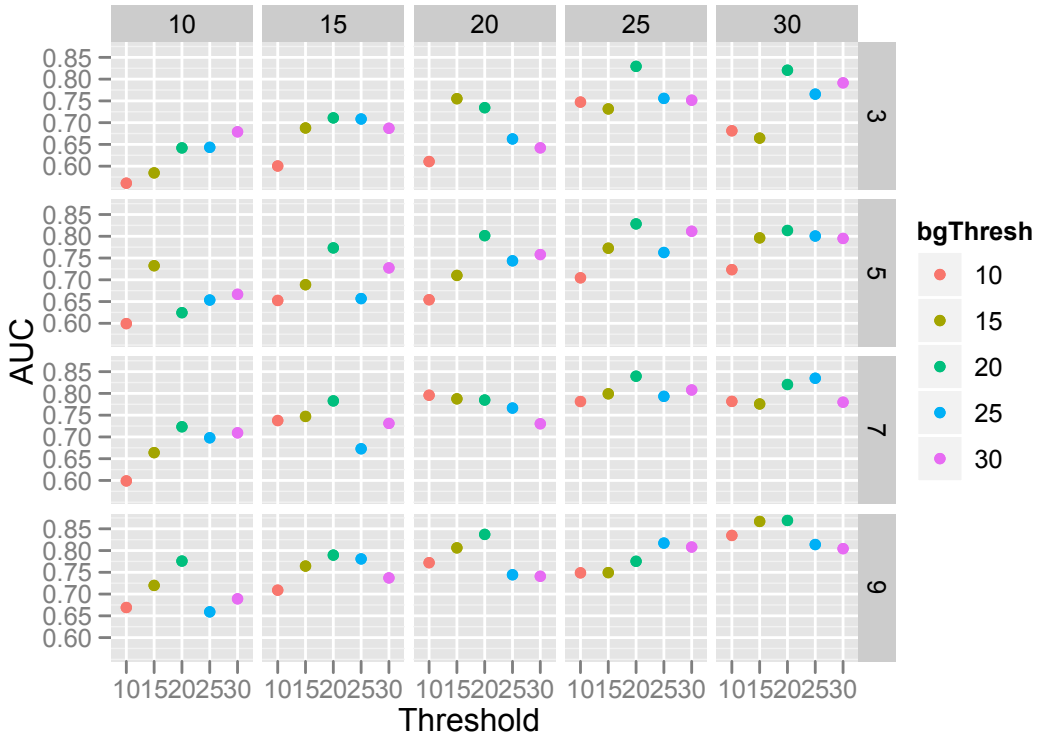


Figure 10: Average AUC values for different `expEdges` thresholds grouped by the `topN` parameter. The group columns refer to different `topN` parameters and horizontal groups to `expEdges` parameter.

between the the first level neighbours which are shared and all the neighbours of the two documents, `shortest path` calculates the average shortest path between two documents, `neighbourhood with features` means that in addition to neighbourhood we also take into account the original terms, `fail distance` tries intuitive approach to handle high-degree node problem by incorporating weights and the `voltage` based similarity which treats the background graph as a electrical network.

Score Sanity Check. First of all, let us check which of the scores are generally higher for related news stories and smaller for unrelated news. For this we took all

the similarity scores for related news stories and for unrelated news stories. For testing whether the population means differ significantly we used the Wilcoxon signed-rank test for all the methods. For all measures except the `shortest-path`, we got the p-value $< 2.2 \cdot 10^{16}$, which means that the methods give significantly different (presumably higher) scores for related news stories than for unrelated news stories. For the `shortest-path` measure we got the p.value = 0.12. This might mean that the `shortest-path` score is not working as well as other measures. The reason for this might be, that the background graph is very dense, thus the shortest path between two terms do not describe the similarity between them very well.

Verifying Scores. The method for verifying the scores is the following. We took a sample of 1000 related and 1000 unrelated news stories. Then we merged these two samples together and calculated the AUC scores and ROC plots for each of the scores. Also, for seeing the variance of the performance, we conducted such experiment 100 times.
Let us first take a look at the general predictive performance of the measures. On the Figure 11 we can see the variability and performance of different methods. We can clearly see, that least variability and the highest AUC is achieved by the `cosine` measure. The second place is achieved by the `neighbourhood with features` measure. In this case it seems, that the background methods do not work very well and thus we get lower prediction scores than expected. The summaries of the different methods are given in Table 4. It is interesting to see, that as we suspected in the scores sanity check phase, the shortest path distance measure does not give even reasonable results on our dataset.
It is also important to note here, that the related news stories are not actually "hand-picked" but they are recommended to the editor by the cosine recommender, thus it is quite logical that the `cosine` measure works very well on the dataset.

Short News. As some news stories are shorter than other, we had an idea to try the performance of the methods on those news stories. The idea behind this is, that we can see how stable the measures are - how much information does the method need, to make correct predictions. For testing this, we created two new news story

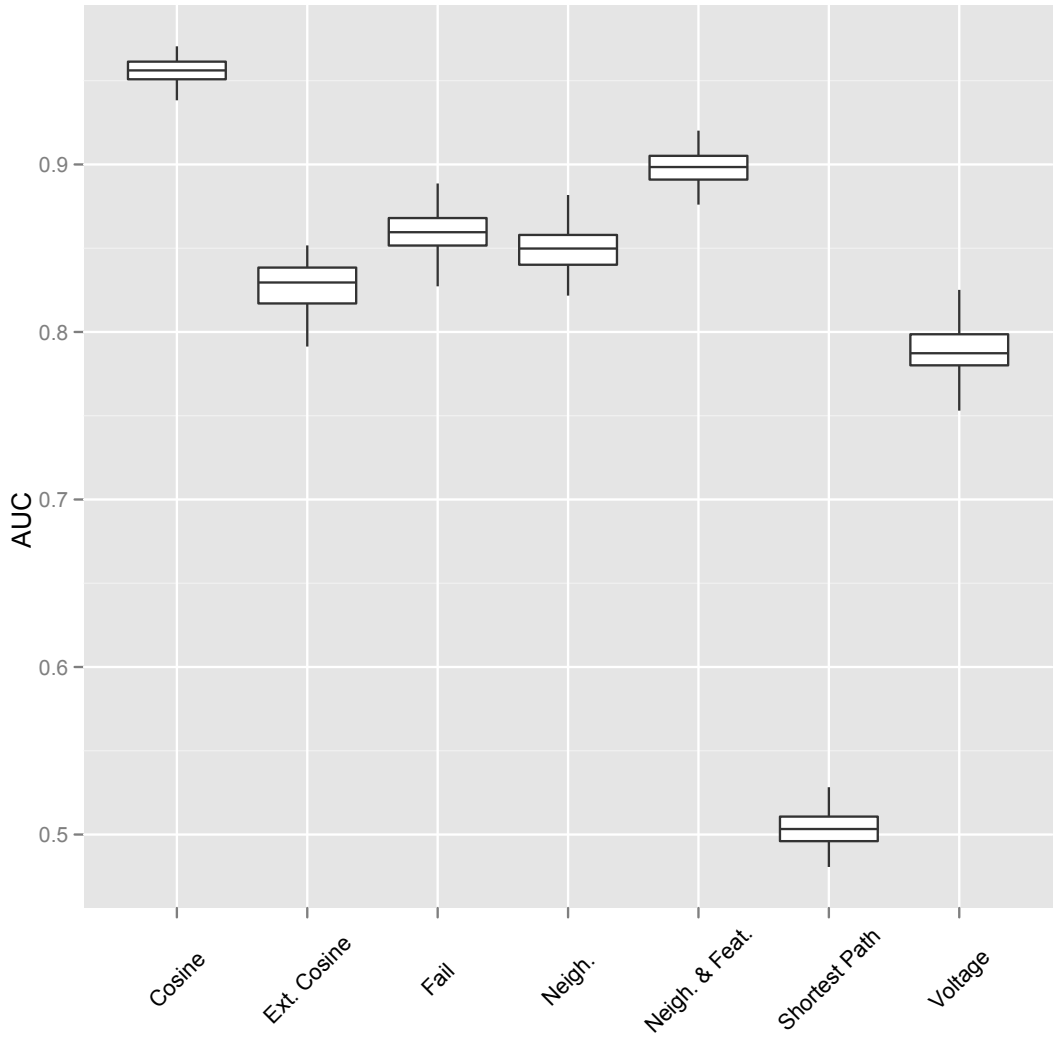


Figure 11: AUC values calculated over all the experiments.

pools, one for related news stories and the other for unrelated news stories. These temporary pools contained news story pairs such, that both of the news stories contained less than 50 words. The reasoning for this is, as we normally use top 50 `tf-idf` terms for the cosine measure, then we are interested in these news stories, which we have less information. Then we divided the document pairs into two parts depending whether they have more than 20 words or 20 or less. Using our iteration technique, we took many samples from the pools, calculated the AUC

Method	Mean	Median
Cosine	**0.95**	**0.96**
Extended Cosine	0.83	0.83
Neighbourhood	0.85	0.85
Shortest Path	0.51	0.51
Voltage	0.79	0.79
Fail Distance	0.86	0.86
Neighbourhood with Features	**0.90**	**0.90**

Table 4: The AUC distribution descriptive values for each of the methods.

values and then analysed the variance of the performance. The results can be seen on Figure 12. Notice, that the `cosine` similarity measure is quite strongly affected when the number of features decreases. On the same time the measures which use the background information seem to be more stable and their accuracy even increases. The reason for this is, that a large number of words "confuse" the background graph measures by generating too much noise. This observation made us wonder, whether the methods which use contextual information, are more stable when the number of available features is small.

Query Expansion. We are going to study the performance of the methods with regards to the query expansion in different settings. By query expansion we mean, that by using the initial document features, we incorporate more relevant information to the similarity calculation.
In the first approach we want to test how small number of query terms affects the retrieval of relevant documents. Consider we want to calculate similarity between two documents d_1 and d_2 and we want to see how well the measures work with n query terms. We extract n terms from document d_1 and before defined `topN` terms from d_2 and perform the similarity calculation. We performed this kind of experiment for random 1000 related and 1000 unrelated news stories taken from the pools for $n = (2, 4, 6, 8, 10, 12, 15)$. This experiment was conducted 5 times. The performance of each of the measures can be seen on Figure 13, where the performance lines are smoothed such, that the variance of the experiments is taken into account. We can observe some interesting aspects. First we see, that the term

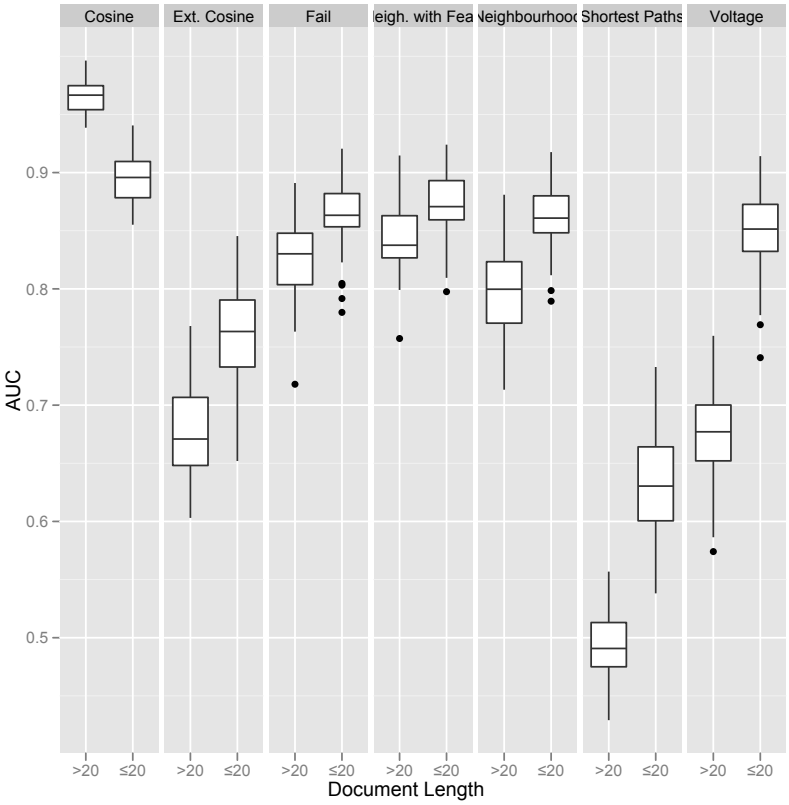


Figure 12: The AUC value difference between news stories with length under or equal to 20 words and for more than 20 words.

reduction has somewhat stronger influence on the `cosine` measure than for the `neighbourhood with features` measure and for ≈ 2 terms their performance is more or less equal. The effect on other measures is quite similar to `cosines`', but we can say that the `neighbourhood with features` measure seems to be the most stable. The numerical values for the experiment can be found from Table 5. When analysing the means of the performance we see, that on average the graph based measures are a bit more stable than just the standard `cosine` measure.

News Stories by Titles. As we are interested in the query expansion then comparing news stories by only using their titles is an interesting experiment. News story titles usually contain the most important information about the actual content. One approach to compare how well the measures work with limited

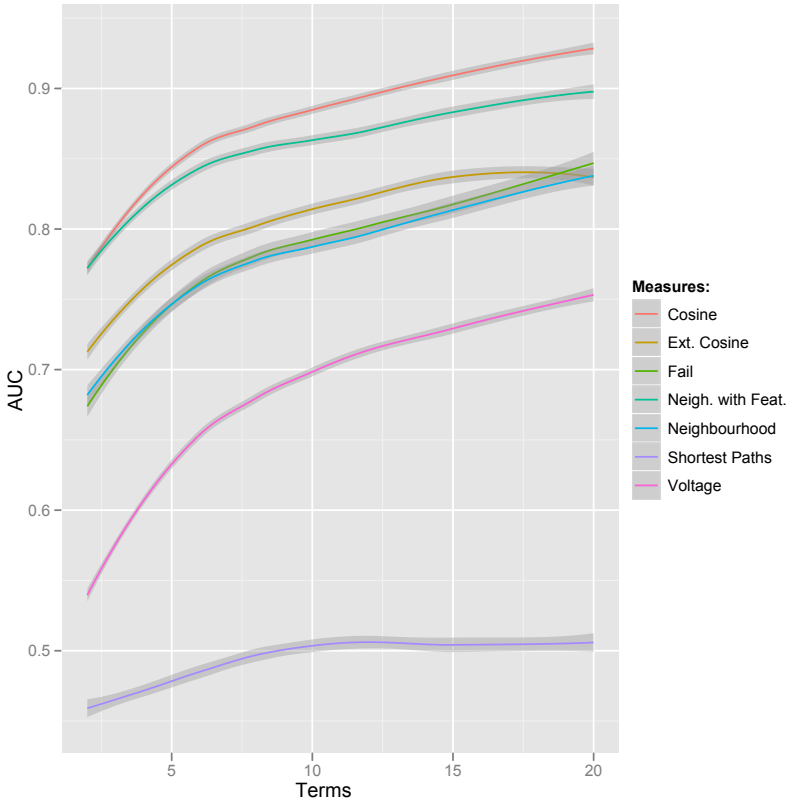


Figure 13: AUC values of the methods when using different number of terms.

	Number of Terms						
Method	**2**	**4**	**6**	**8**	**12**	**15**	**20**
Cosine	0.77	0.83	0.86	0.87	0.88	0.89	0.91
Ext. Cosine	0.71	0.77	0.78	0.81	0.81	0.82	0.84
Neighbourhood	0.68	0.73	0.76	0.78	0.79	0.79	0.82
Shortest Path	0.46	0.47	0.48	0.50	0.50	0.51	0.50
Fail Distance	0.67	0.73	0.76	0.78	0.80	0.80	0.82
Neigh. with Feat.	0.77	0.82	0.84	0.86	0.87	0.86	0.89
Voltage Distance	0.54	0.61	0.65	0.68	0.7	0.72	0.73

Table 5: The mean AUC values for each of the methods for different number of terms.

Method	Mean	Median
Cosine	0.71	0.71
Extended Cosine	0.68	0.68
Neighbourhood	0.64	0.64
Shortest Path	0.76	0.76
Voltage	0.53	0.53
Fail Distance	0.61	0.61
Neigh. with Feat.	0.73	0.73

Table 6: The AUC distribution descriptive values for each of the methods in case we are using only document titles as features of a document.

amount of information, is to use only the words in the titles as features. For this we extracted only the titles of the news stories and gave all the words an equal weight. Then we calculated the similarity for 3000 related and 3000 unrelated news stories. Then we used the scheme by sampling 500 documents from both datasets for 100 times and plotted the distributions of the AUC scores on Figure 14. Surprisingly we see, that the `shortest-path` measure works very well on this dataset. Also the Wilcoxon test gives p-value $< 2.2 \cdot 10^{16}$ for all the measures including the `shortest-path` measure. This may be due to the fact, that we have a small number of very specific terms and if there exists a path between two documents, then the documents are most probably related. On the other hand, if the titles are very different, they are most probably far from each other. This might not be the case we tested in the short news experiments as there are usually greater number of features and the probability for a path from one document to another is much greater. Another interesting fact is that the `neighbourhood with features` measure is quite stable. In this experiment the cosine measure works remarkably poorly. We see, that the main issue with the cosine measure is the stability. The mean and median values of the methods performances can be found in Table 6.

Similarity Using Tpf-Idf-Tpu. As in our ongoing research tpf-idf-tpu shows potential for creating mind maps of documents, we will also make an experiment to analyse how the measures work when the initial keywords are extracted by the tpf-idf-tpu measure i.e. word pairs. Intuitively, as these concept associations are

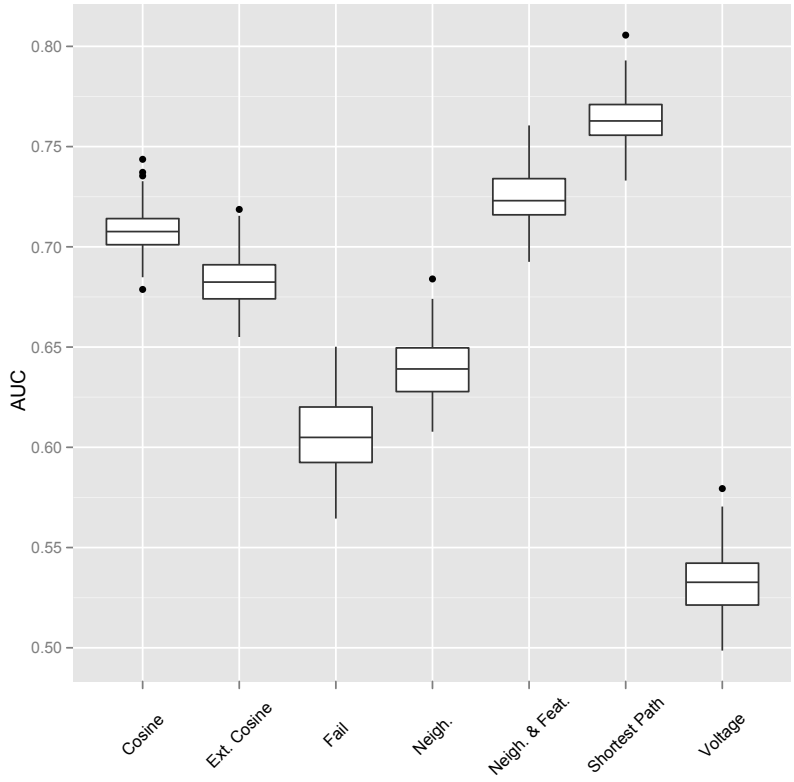


Figure 14: The AUC scores for news stories where the news story titles are used as the features of a document.

important given specific document, we can reason, that these terms are also important in the context of the document.
The AUC distribution using these features can be see on Figure 15. We see, that `cosine` measure is still working better than other measures, though the performance is more than 20% worse than for tf-idf terms. Surprisingly this feature extraction method does not give good results on document similarity.

Ensemble. As we have seen from the experiments the methods work well in many different settings. This gives a reasonable assumption that if we could use these measures effectively together, we could get better results on predicting the related news stories. Though, we achieved very good performance with the cosine measure on the news stories in general, we also tried an ensemble model. We used

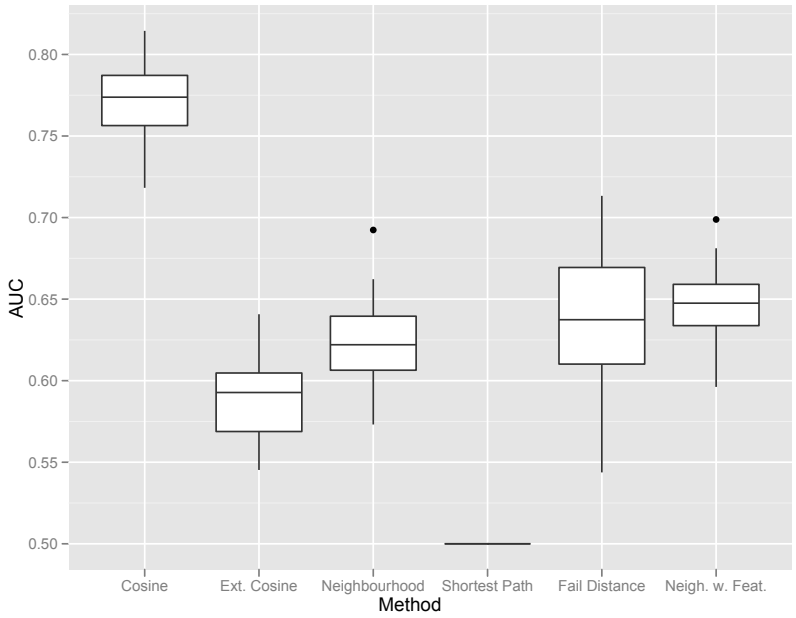


Figure 15: AUC distributions for different methods using the terms of top `tpf-idf-tpu` term pairs as features.

the logistic regression model and saw that the performance gain was minimal (from AUC ≈ 0.957 to AUC ≈ 0.963). This was also quite well explained by principal component analysis which showed that most of the variance in data is explained by the cosine measure.

Another setting where we tried the ensemble method was predicting news stories by their titles. The best mean AUC score 0.76 was achieved by the shortest path method. For creating the model, we divided the data into train and test sets. Then we trained the model and calculated the AUC scores on the predictions. We iterated through this process 20 times. By using the logistic regression model we were able to achieve mean AUC score of 0.90. The best model summary can be seen in Table 7. Though the p-value for the `neigh_inc` feature is larger than 0.05, removing it from the model makes the overall predictions worse.

Performance of LSI. The last experiment regarding the document similarity is as follows. We sampled 500 document pairs from the related news story pool and 500 document pairs from the unrelated news story pool. Then we merged all these

	Estimate	Std. Error	z value	Pr($>\|z\|$)
(Intercept)	-2.6820	0.0562	-47.75	0.0000
short	13.2287	0.5399	24.50	0.0000
cos	27.4236	1.7984	15.25	0.0000
neigh_inc	-0.7644	2.2623	-0.34	0.7355
volt	-0.0013	0.0008	-1.72	0.0860

Table 7: The best found model for predicting news story relatedness by using only titles. The `short` is for shortest path measure, `cos` is for cosine, `neigh_inc` is for the neighbourhood with features and `volt` is for the voltage measure.

news stories together and calculated the LSI matrices and reduced the matrices to 250 dimensions. After calculations we took top $n = (2, 3, 4, 5)$ terms of the documents and calculated the similarity with LSI methodology between the documents which originated from the pools. Then for every number of different initial terms we calculated the AUC values. We took the same news story samples which were used during LSI and calculated the similarities with other methods, respectively with $n = (2, 3, 4, 5)$ initial terms. The comparison of proposed measures and LSI is given in Figure 16.

A quite big downside of the method is that computing the singular value decomposition needs a lot of memory and computational power, thus it is only reasonable to use standard LSI on small scale datasets. For large scale databases there are available methods which deal with feature selection before computing the singular value decomposition, for instance [YYLC09, STS04], but the in-depth analysis of them is not in the scope of current thesis.

5.4 Behaviour Experiments

Usually every browsing action of a specific web page visitor is stored into a database. The information incorporated in the log is the time, user id, the source and the destination. Usually the user browsing behaviour is traced by session or cookie id. For the session id case we can trace the behaviour of user during one session (which is ended after x minutes) and for cookie id's we can do this over longer periods of time. In our case, we can use the access logs to extract the

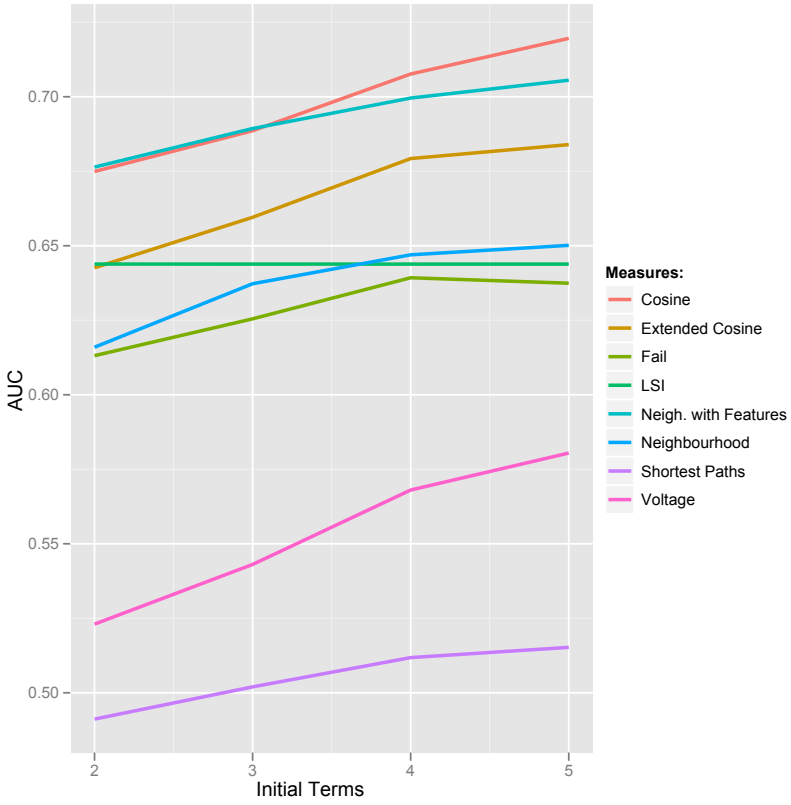


Figure 16: The AUC values for LSI and other previously used document similarity measures.

information how users browse news stories.

A quite interesting way to analyse how well the scores work, is to find correlations between the browsing behaviour of users and the similarity scores. For this we extracted the user sessions from the access database and we decided to split user session if there is more than 15 minutes between two clicks. This is reasonable, as there might be longer articles which take more time to read, but 15 minutes strongly indicates that users has been away for a while. This is also quite important, as users interest may change between sessions, thus is is important to analyse them separately.

Methods. Consider a transaction $T = \langle t_1, t_2, ..., t_n \rangle$. For analysing the behaviour, we have two different approaches. The first assumes that user chooses the next news story based on the last one he/she read - thus the average similarity score might be higher between news stories which are browsed sequentially. Thus, we calculate the similarity scores only between the items $t_i, t_{i+1} \in T$.
The other approach is under the assumption that the user tends to read similar news stories during one session and the sequentiality is not that important. For the second option we calculated all the pairwise similarities between two news stories which occur in one session. So given the same transaction T, this means that we calculate distances between any $t_i, t_j \in T, i < j$.

Frequent News. Some of the news stories are read sequentially more often than others. First we assume, that the similarity scores between frequently together read news are higher than the scores which are read together non-frequently. We tested the hypothesis for both cases when the news stories are visited sequentially and when they are visited together in one session. For testing the hypothesis we ordered the scores by frequency and took the top 5 and 10 percent of the most frequent news stories and for 200 random samples used the Wilcoxon Rank Sum Test to analyse whether the score means differ significantly with regards to randomly selected scores from the set. It is important to note here, that in the sequential news pairs in the top 5 and 10% there were 18.83% and 15.83% of hand-picked related news stories, respectively. For the session news pairs for 12.55 and 10.76% only were 0.26% and 0.57% marked as hand-related, respectively.

The mean p-values of the tests can be seen in Table 8. We can observe some interesting things:

- For the top 5% news stories we can observe that the measures using the background graph measure may give more reasonable predictions than the traditional cosine measure, as the p-values show significant difference between the similarity scores. For the 10% sequentially browsed news stories we observe that this does not apply any more.

- For both, top 5% and 10% of the news stories which are frequently browsed

Sequentiality	Cosine	Ext. Cosine	Neighbourhood	Fail	Neigh. with Feat.
Sequential (5%)	0.21	0.021	0.032	0.027	0.028
Sequential (10%)	0.22	0.039	0.079	0.076	0.079
Combination (5%)	$1.14 \cdot 10^{-9}$	$1.3 \cdot 10^{-12}$	$< 2.2 \cdot 10^{-16}$	$< 2.2 \cdot 10^{-16}$	$< 2.2 \cdot 10^{-16}$
Combination (10%)	$2.6 \cdot 10^{-11}$	$5.2 \cdot 10^{-13}$	$< 2.2 \cdot 10^{-16}$	$< 2.2 \cdot 10^{-16}$	$< 2.2 \cdot 10^{-16}$

Table 8: The p-values for Wilcoxon Rank Sum Tests. The news stories were ordered by frequency and then the similarity scores of top 5 or 10% of the stories were compared to randomly selected pairs of news stories.

Sequentiality	Cosine	Ext. Cosine	Neighbourhood	Fail	Neigh. with Feat.
Sequential (5%)	0.54	0.60	0.60	**0.61**	0.60
Sequential (10%)	0.53	**0.57**	0.56	0.56	0.56
Combination (5%)	0.58	0.62	0.66	**0.68**	**0.68**
Combination (10%)	0.57	0.59	0.63	**0.65**	0.63

Table 9: The AUC values when the top 5 and 10% were set as related and all other news stories as unrelated.

within one session, tend to have different similarity scores.

- In the pairwise sequential behaviour the cosine measure does not seem to work well.

For seeing the actual performance of the measures, let us make an assumption that these news stories which are frequently visited together should be marked as related. Thus we mark all the top 5 or 10% of the news stories as related and the rest as unrelated. Then we calculate the classifiers AUC which shows the predictive performance. We provide an analogous table as before, but with AUC values seen in Table 9. We can observe that the similarity measures which use background information always work better than the standard cosine measure. Though, the AUC values are not very high, in the best case our proposed measures get 8.5% better performance.

Dependent News. In the previous paragraph, we assumed that interesting news stories are those which are frequently looked at together. In the following analysis we assume, that interesting are these news stories which are dependent on each other. As previously we sorted the news stories by their frequency, then now let

Sequentiality	Cosine	Ext. Cosine	Neighbourhood	Fail	Neigh. with Feat.
Sequential (5%)	**0.75**	0.65	0.68	0.69	0.70
Sequential (10%)	**0.65**	0.58	0.62	0.62	0.63
Combination (5%)	**0.70**	0.64	0.69	**0.70**	**0.70**
Combination (10%)	0.66	0.62	0.67	**0.69**	0.67

Table 10: The AUC values when the top 5 and 10% of the news story pairs, ordered by interest factor, were set as related and all other news stories as unrelated.

us analyse the same properties of news story similarities by sorting the news pairs by their *interest factor* score. When the sequentially browsed news are sorted by their interest factor the number of hand-related news stories in top 5 and 10% are 38.51 and 26.96%, respectively. For the within session created news pairs the hand-related news stories in the top 5 and 10% are 28.13 and 17.93%, respectively. After sorting the news story pairs by their lift we assigned the top n% of the pairs to be related and the rest as unrelated. The AUC scores for such classifiers is summarized in Table 10. As we can see in this setting the performance differences are quite small. There seems to be correlation between the percent of related news stories and the performance of cosine measure - in this sense other methods may be a bit more stable, but not significantly. All in all it seems, that the quality of predictions depends how many hand-related news stories happen to be in the sorted list. What we can see is, that the lift measure seems to order the news quite well according to their actual relatedness.

In the current experiments we do not present results with the voltage distance. The reason for this is, that these experiments were conducted before we started using the voltage distance. The goal of this experiment was to see, whether the methods which use background graph work significantly better than the ones which use just the standard similarity. As the results indicate that this is not so, then we did not find it necessary to conduct the experiments with the voltage distance.

5.5 Experiments with Keywords

As for every news story editor has to pick a minimum of two keywords for the news story, it leads us to the final experiments, where our goal is to first analyse the

correlation between the human picked keywords for a news story and the calculated tf-idf and tpf-idf-tpu keywords and the second experiment is to see, how the hand-picked keywords and the background graph correlate. We will also see, how the document based and sentence based log-likelihood measures differ.
Before going into details let us briefly define two important measures used in the evaluation of pattern recognition - *precision* and *recall*. In our case, let us consider that we have the hand-picked keyword set K_h and retrieved keyword set K_r. Then precision is given as:

$$precision = \frac{|K_h \cap K_r|}{|K_r|},$$

which describes the relative overlap between the matched keywords with regards to the retrieved documents. Another measure is recall, which is given as:

$$recall = \frac{|K_h \cap K_r|}{|K_h|},$$

which is the relative overlap between the matched keywords with regards to the hand-picked keywords. As both of these measures have their weak points, a popular method is to combine them into F_1 measure:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall},$$

which is basically the weighted average of the precision and recall. More information of these measures can be found in [TSK05].
For tf-idf keywords it is seems reasonable to select top 5 keywords for news story and then we can calculate the precision and recall for these retrieved keywords. For tpf-idf-tpu keywords this is less trivial, as the retrieved *keywords* are actually term pairs. We can approach the problem in two different ways: a) we create pairs between all the keywords and calculate the precision and recall between create pairs and retrieved pairs b) we take top 5 pairs, transform them into 10 keywords and calculate the precision and recall between the retrieved keywords and actual keywords.

Background Graph. The methodology for validating the background graph is as follows. For every hand-picked keywords selected for a news story, we will create pairs between all of them. For instance if the keywords for news story are 'war','iraq' and 'usa', we get the pairs ('iraq', 'war'), ('iraq','usa') and ('usa', 'war'). Similarly when creating the background graph, we store the values $k_{11}, k_{12}, k_{21}, k_{22}$, where they represent the co-occurrences of terms in the keyword sets, and calculate the log-likelihood ratios for each of these pairs.
For validating the background graph we take all such hand-picked keyword pairs which have edges in the graph. Then we will get two ordered sets, one where the keyword pairs are ordered by their co-occurrences in the keyword sets and one where they are ordered by their co-occurrences in the news stories. This enables us to calculate the *Kendall tau distance* between these two lists τ_1, τ_2:

$$K(\tau_1, \tau_2) = \sum_{\{i,j\} \in P} K'(\tau_1, \tau_2),$$

where P is the set of pairwise elements in τ_1 and τ_2 and the function $K'(\tau_1, \tau_2)$ is given as:

$$K'(\tau_1, \tau_2) = \begin{cases} 0, & \text{if i and j are in the same order in } \tau_1 \text{ and } \tau_2 \\ 1, & \text{otherwise} \end{cases}$$

As the maximum value of the distance is $m = \frac{n(n-1)}{2}$ and minimum distance is 0, we will use m as normalizing constant by dividing $\frac{K(\tau_1,\tau_2)}{m}$, which gives us a distance measure in the range $[0, 1]$.

Keyword LLR vs. Other Log-Likelihoods. For seeing how the keyword log-likelihoods correlate with the log-likelihood ratios calculated from the document corpus, we conducted an experiment, where we chose such random 100 keyword pairs which had edge in the background graph. Then we created to ordered term pair lists - one which was reverse ordered by the keyword log-likelihood measure and the others which were ordered by the sentence and document based log-likelihood measure. Then we calculated the Kendall tau distances between

these ordered list. On the Figure 17 we can see the distributions of the Kendall tau distances of lists which are ordered by different log-likelihood score. As we can see, the methods usually do not order the term pairs similarly. The most similar orderings are between the keyword and sentence based log-likelihood ratios, but even a Kendall tau's distance of 0.4 means that the orderings are not very similar. The document based calculation orders the pairs completely differently from the keyword based measure. Its distance is even greater than for the randomly ordered pairs, which means, that it tends to order the term pairs in opposite manner wrt keyword likelihood measure. Also we can see, that sentence and document based log-likelihood ratios work in quite a different way. As the sentence level measure works more similarly to our goal, we chose this as our measure for the background graph.

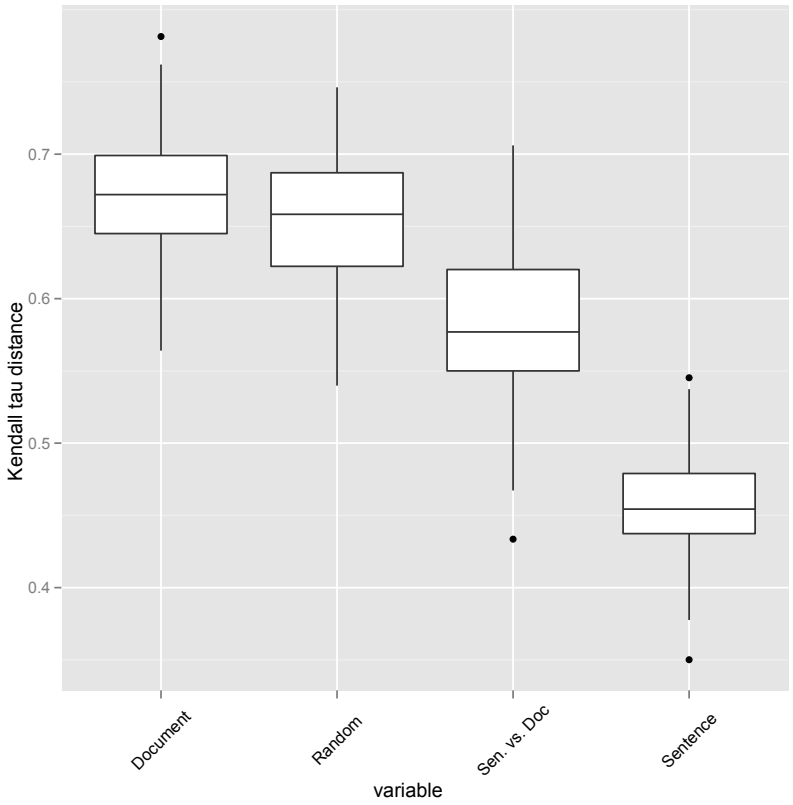


Figure 17: The Kendall tau distance distributions between lists which are ordered by differently calculated log-likelihood measures. The labels *Document* and *Sentence* state that the list of term pairs is ordered by the log-likelihood ratio calculated over the corresponding unit. The *Random* states that the term pairs are ordered randomly. These three sets are compared to the keyword lists ordered by the log-likelihood values of the keywords co-occurrence. The *Sen. vs. Doc* shows the distance between the sentence level and document level orderings.

6 Conclusion

In this thesis we address the problem of finding non-trivially similar documents from a large document corpus. We have given a theoretical overview of some of the traditionally used document similarity methods, which are the cosine similarity and latent semantic indexing using the tf-idf weights. We introduced a concept of a background graph and defined document similarity measures, which used this graph for measuring contextual similarity. We conducted several experiments on Estonian news story provider *Postimees.ee* dataset. For general case our proposed methods did not work as well as the traditional cosine measure. Though, we observed that the proposed measures are quite effective in those cases where the documents are represented by a few high-quality features. On the *Postimees.ee* dataset we also observed that some of our proposed measures work better than latent semantic indexing method. We saw that the proposed similarity measures, which use the background graph, have potential for predicting which news stories would be interesting for the news website user.

The conducted experiments in this thesis give a strong indication that there might be several reasonable applications for the background graph. In the future we want to put more emphasis on how different contexts are represented in the background graph and how it would be possible to map documents into different contexts. There is an ongoing research, which studies the ways of using the proposed background graph and the tpf-idf-tpu measure for creating mind-maps and summaries of documents.

7 Mittetriviaalselt sarnaste dokumentide otsimine suurest dokumentide korpusest

Magistritöö (30 EAP)

Oskar Gross

Sisukokkuvõte

Käesoleva magistritöö eesmärgiks on uurida, kuidas leida mittetriviaalselt sarnaseid dokumente suurest dokumentide hulgast. Antud töös kirjeldatakse nii traditsioonilisi meetodeid dokumentide sarnasuse uurimiseks kui ka tutvustatakse uusi. Lisaks viiakse läbi eksperimendid, et uurida väljapakutud mõõtude käitumist andmetel.

Traditsioonilised dokumentide sarnasusmeetodid mõõdavad sarnaste sõnade esinemist kahes dokumendis. Antud töös käsitleme, mis probleemid kaasnevad kui me kasutame dokumentide sarnasusmõõdu arvutamisel vaid viimastes leiduvaid sõnu, tutvustame olemasolevaid kui ka pakume välja uusi mõõte nende probleemide ületamiseks. Dokumendid on mittetriviaalselt sarnased, kui nad sisaldavad vähe ühiseid sõnu, kuid on kontekstuaalselt sarnased.

Selleks, et tuvastada dokumentide konteksti pakume töös välja taustgraafi kontseptsiooni. Taustgraafi eesmärk on modelleerida sõnade ehk kontseptsioonidevahelist seost, andes rohkem kaalu nendele sõnadele, mis esinevad tihti koos. Saadud taustgraafi kasutame erinevate dokumentidevaheliste sarnasusmõõtude arvutamiseks.

Käesolevas töös käsitletakse ka kasutaja käitumise ja sarnasusmõõtude vahelist seost. Töös antakse lühiülevaade järjestuste kaevandamise põhimõistetest ning kasutakse neid, et uurida, kuidas erinevad sarnasusmõõdud modelleerivad kasutaja käitumist.

Töös viiakse läbi erinevaid eksperimente uudisportaali Postimees.ee andmetel. Taustgraafi uurimisel näeme, et loodud graaf kirjeldab kontekstisiseseid kontseptsioonide vahelisi seoseid väga hästi. Uurides sarnasusmõõte näeme, et üleüldisel uudiste soovitamisel töötab meie väljapakutud meetoditest paremini traditsiooniline meetod. Mõõdud, mis kasutavad taustgraafi informatsiooni, annavad pare-

maid tulemusi traditsioonilistest meetoditest, juhul kui me kasutame väheseid, kuid kvaliteetseid andmeid dokumendi kohta.

Käesolev magistritöö pakub välja uue metoodi dokumentide sarnasuse leidmiseks ning näeme, et antud meetodid töötavad kindlatel juhtudel paremini kui varem kasutusel olnud mõõdud. Väärtuslikuks avastuseks töös võib tuua välja taustgraafi loomise, mille edasine uurimine võib anda huvitavaid tulemusi teksti kontekstituvastuses ja ka mõttekaartide koostamisel.

Autor soovib avaldada südamest tänu oma juhendajatele, kelle ideed, põhjalikkus ja järjekindlus ei ole olnud mitte abiks ainult käesoleva töö kirjutamisel vaid seda kogu minu viimaste aastate, ja loodetavasti ka tulevaste, õpingute jooksul.

References

[ABC+09] Marco Arment, Roger Black, Jay Chakrapani, Sarah Chubb, Anil Dash, Paul Ford, Jeffrey MacIntyre, Karen McGrane, and Jeffrey Zeldman. A web & mobile app for reading comfortability: Readability. https://www.readability.com/ (Last accessed 21.03.2011), 2009.

[BDO+95] M. W. Berry, S.T. Dumais, G.W. O'Brien, Michael W. Berry, Susan T. Dumais, and Gavin. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[BP00] Daniel Billsus and Michael J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10:147–180, February 2000.

[Bra08] Roger B. Bradford. An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 153–162, New York, NY, USA, 2008. ACM.

[CYWM03] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications*, APWeb'03, pages 406–417, Berlin, Heidelberg, 2003. Springer-Verlag.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[Don00] Stijn Dongen. A cluster algorithm for graphs. Technical report, Amsterdam, The Netherlands, The Netherlands, 2000.

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19:61–74, March 1993.

[Faw06] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27:861–874, June 2006.

[FLGD87] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30:964–971, November 1987.

[FMT04] Christos Faloutsos, Kevin S. Mccurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *Knowledge Discovery and Data Mining*, pages 118–127, 2004.

[GXCL09] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 267–274, New York, NY, USA, 2009. ACM.

[Han05] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[HK04] Khaled M. Hammouda and Mohamed S. Kamel. Document similarity using a phrase indexing graph model. *Knowl. Inf. Syst.*, 6:710–727, November 2004.

[htm] HtmlUnit - Welcome to HtmlUnit. `http://htmlunit.sourceforge.net/`. [Online; accessed 22-May-2011].

[Hyn10] Teemu Hynönen. Mining unobvious connections between terms from unstructured text. Master's thesis, University of Helsinki, 2010.

[Kaa97] Heiki-Jaan Kaalep. An estonian morphological analyser and the impact of a corpus on its development. *Computers and the Humanities*, 31:115–133, 1997. 10.1023/A:1000668108369.

[LH02] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from web documents. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 588–593, New York, NY, USA, 2002. ACM.

[LL02] T.-P. Liang and H.-J. Lai. Discovering user interests from web browsing behavior: An application to internet news services. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7 - Volume 7*, HICSS '02, pages 203–, Washington, DC, USA, 2002. IEEE Computer Society.

[Luh58] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research Development*, 2(2):159–165, 1958.

[mcl] Mcl - a cluster algorithm for graphs. `http://micans.org/mcl/`. [Online; accessed 16-May-2011].

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[PF03] Christopher R. Palmer and Christos Faloutsos. Electricity based external similarity of categorical attributes. In *In PAKDD 2003*, pages 486–500. Springer, 2003.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[Pro10] NCAR Research Application Program. *verification: Forecast verification utilities.*, 2010. R package version 1.31.

[SK97] Hidekazu Sakagami and Tomonari Kamba. Learning personal preferences on online newspaper articles from user behaviors. *Comput. Netw. ISDN Syst.*, 29:1447–1455, September 1997.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1983.

[STS04] K. Shima, M. Todoriki, and A. Suzuki. Svm-based feature selection of latent semantic features. *Pattern Recognition Letters*, 25(9):1051 – 1057, 2004.

[TF06] Hanghang Tong and Christos Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–413, New York, NY, USA, 2006. ACM.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[YYLC09] Jun Yan, Shuicheng Yan, Ning Liu, and Zheng Chen. Straightforward feature selection for scalable latent semantic indexing. In *SIAM International Conference on Data Mining*, pages 1159–1170, 2009.