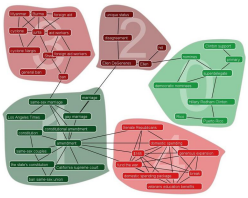
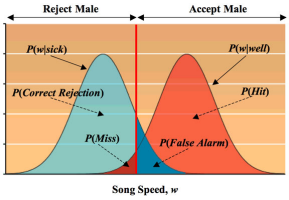
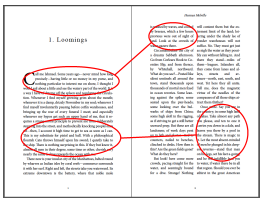
Excel@FIT 2015
http://excel.fit.vutbr.cz

Automatic Link Detection in Parts of Audiovisual Documents

Marek Sychra*







Abstract
This paper deals with the topic of finding similarities amongst a group of short documents according to their topic. It solves finding borders between two topically different parts in a large document. The goal is achieved by text and word analysis, which contains learning the meaning and importance of each word. Both problems use the same analysis, only different application. The solution of the first problem (*link detection*) gives great results despite using a simple analysis method. The second problem (*story segmentation*) is harder to rate precisely, but also gives good results. Both tasks were tested against short documents - world news reports. The main motivation for implementation and research was practical application with the use of presentation materials from lectures at FIT BUT (linking parts of different lectures and courses).

Keywords: Topic detection — Link detection — Story segmentation — Term frequency - Inverse document frequency

Supplementary Material: *N/A*

*xsychr05@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

This work has two main goals - the first is to find and test a method for finding links amongst parts of text documents. The second is to release this feature upon the recorded courses at FIT BUT. The first itself can be divided into two parts. Firstly, it is important to study methods working with *topic detection*, design and implement text document comparing algorithm according to their topic. Then, with the knowledge of appropriate methods we have to implement a story segmentation algorithm capable of dividing a single document into a group of smaller ones, each with its own topic. Both subproblems should be also evaluated properly. In case of good results, it could be introduced into common use (the second goal) by linking parts of different courses (at FIT BUT), which are about the same topic. This feature contained in online video streaming could be helpful for students to understand the lectures and their connections properly.

As was mentioned before, the whole implementation problem is in fact two smaller programs. The outcome of the first should be able to tell whether two text documents share the topic. For example, after feeding the program with 100 news reports, it should give us information which of the news discuss the same event. The result of comparison is a number between 0-1. The goal is to choose the right *threshold*, which makes the border between the same topic and a different one. The second task (which in fact comes in first) is to divide the text into smaller parts of which none can be divided again into two or more separate

documents. After the large text has been segmented, we compare the parts against each other and find out which parts discuss the same topic.

There are many approaches to get the meaning of a text. According to Ch. Wartena and R. Brussee [1] it is possible to extract few keywords describing a group of documents. The advantage of this approach is that it doesn't need training data, the algorithm is trying to cluster keywords to get topics; each document generates a group of keywords that describe the document. Similar solution to the one discussed here is brought by T. Hazen [2], who works with recorded phone calls. He states that every document can be represented by a set of counted occurrences of each word in the document. He also filters the most common words and for each topic he chooses only 10% of words that appear in documents with this topic. However our approaches differ when topic detection is concerned. We compare two documents to decide whether they have same topic. In his case, he creates a classifier for each topic. Then he trains each classifier the features of one topic (common words, etc.) and for each document he determines (using the classifiers) the probabilities of the document belonging to each topic. He mentions using Naive Bayes classifier and Support vector machines. But since our task is more about *link detection* than *topic detection* it is not necessary to use classifiers. When segmentation is concerned, M. Riedl and Ch. Biemann [3] present methods *TextTiling* and *TopicTiling* in their paper. At first, they split the document into small basic units (sentences) and calculate *cosine similarity* between each two adjoining sentences. Afterwards they plot a curve with the values and for each found minimum they calculate a *depth score* which determines the probability of the breaking point between two topics.

Our solution of the problem is quite unique - because of the connection of two problems (link detection + story segmentation). When describing documents we use a method *term frequency - inverse document frequency* (same as [2]) and compare by *cosine similarity*. The solution of the story segmentation problem is based on [3], but differ in boundary weighting and many smaller details. The story segmentation part could be labeled as a whole new approach.

From comparisons we get quite good success rate (85-90%). This means that for every document we are positive to find a few documents that are sure to be of the same topic. When we talk about segmentation, it is relative, what toleration of a topic boundary we use or what is our exact goal. After preparing the algorithm for Czech language, it could well be a good studying support.

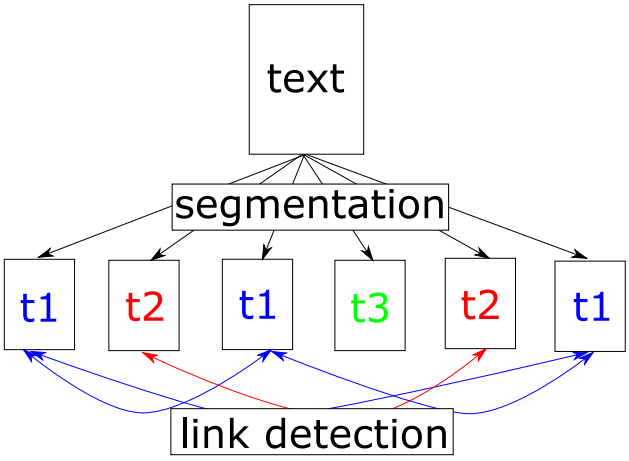


Figure 1. First, the text is segmented into parts that have the same topic. Then we search for links (same topic) amongst all newly created parts; t1, t2 and t3 stand for different possible topics.

2. Approaches in topic detection

The solution and approach *topic detection and tracking* (TDT) originated from the growing amount of information that appear all around people. Methods to filter and cluster information according to its similarity so that it would ease searching for important and required information were needed. Research of TDT and similar problems began to expand around the year 1997, when it gain many new benefits with the financial help from DARPA and its program *Translingual Information Detection, Extraction, and Summarization* (TIDES). The initiative divided the TDT problem into five main branches: [4]

- Story Segmentation - Detect boundaries between topically cohesive sections
- Topic Tracking - Keep track of documents similar to a set of example documents
- Topic Detection - Build clusters of documents that discuss the same topic
- First Story Detection - Detect if a document is the first document of a new, unknown topic
- Link Detection - Detect whether or not two documents are topically linked

It is possible to choose many sides from which to look upon the complex problem of TDT. Therefore it is important to know what is the input and what should be the output. When we want to find the topic of a document, classifiers are applicable. Each classifier is trained with the features and traits of each one of known topics. After running a document through a classifier, it produces a probability of the document

belonging to the topic. Another possible task might be to generate a group of keywords to summarize the document; that might be handy when abstracts and summaries are concerned. Last but not least is to find links between two texts.

3. Used text analysis and segmentation

Analysis and preprocessing of the text comes before the whole process of creating a vector. We want to get to know the text and what it's about using machine learning the best. That's why we've got to prepare the data for upcoming methods using the following:

Stemmer Human can understand that a word in singular or plural is still the same word, but for machine we've got to unite all these words into one - using a stemmer, which cuts off prefixes and suffixes and leaves only the word core.

Stoplist In all languages there are words that carry no important information, only connect more important words and make the sentence whole. E.g. prepositions, conjunctions, pronouns, ... Especially in the spoken language. All these words must be eliminated from analysis.

Now that we've got a text stripped of unimportant words and all of the prefixes and suffixes have been cut off, we start with the analysis. We chose a method *term frequency - inverse document frequency*, which creates a vector for each document for further comparisons.

$$tf(t,d) = \begin{cases} 0, & f(t,d) = 0 \\ 1 + \log_{10} f(t,d), & f(t,d) \in \mathbb{N} - \{0\} \end{cases}$$

$$idf(t,D) = \log_{10} \frac{|D|}{|\{d \in D : t \in d\}|}$$

$$tf\text{-}idf(t,d,D) = tf(t,d) * idf(t,D)$$

t	word
d	document
D	document set
$f(t,d)$	frequency of word t in document d
$tf(t,d)$	TF value of word t in document d
$idf(t,D)$	IDF value for word t in document set D
$tf\text{-}idf(t,d,D)$	TFIDF value for word t in document d and document set D

The method *term frequency* calculates relative frequency of each word in a single document (can be calculated separately). On the other hand the method *inverse document frequency* can be calculated only in a group of documents, due its main power - to find out the weights of each word according to its appearance over all documents. Therefore, when a word appears in every document, we evaluate it with a small weight - it gives us no further information for determination whether two documents are similar.

So we take a list of all words over the document set and for each document we: 1) take each word from the list 2) compute TFIDF value 3) append value to the vector.

After creating vectors it is necessary to somehow compare them. And because laws of mathematics apply even here, vectors' proximity can be determined by computing cosine of the angle a pair of vectors have between them; this is called *cosine similarity*. From mathematics we know the cosine between two vectors can be computed by equation 1.

$$\cos\theta = \frac{A \cdot B}{|A| \cdot |B|} \tag{1}$$

We get a cosine value using vectors of two documents which we want to compare and from the fact that $\cos 0^\circ = 0$ we know that the closer the value is to zero, the more similar the vectors (and documents) are.

3.1 Segmentation

The approach for story segmentation contains the same methods as link detection, it's just supplemented by a few details to be able to find the actual boundaries. The goal is to find the boundaries between parts differing by their topic as accurately as possible. First of all we split the text into many small units/sentences (50-80 words per unit). Then we calculate the similarities between each two adjoining units (and up to five more units to either side) in order to determine whether they share a topic. But if we took the bigger range (>80 words), the boundary could very well fall in the middle, so it could not be found. Therefore we used a sliding window technique. We calculate cosine values in all dividing and then we shift the start of each part by a number of words and calculate cosines again. Thus we get more precise results without having to reduce the context. We apply it N times, where N is the least common multiple of the shift and the length of the basic block. The result will be a chart with plotted cosine values (figure 5). In the graph all the minimums are possible boundaries; some, however, have much smaller values and therefore bear greater possibility to be a more accurate boundary.

4. Experiments and results

The dataset *TDT5 Multilingual News Text* which contains short (400 words) reports from world news, was used for all testing and experimenting. It includes 250 distinct events in total. For each event there are many short and long reports.

When experimenting with link detection we chose 1100 reports from 53 events (topics) and split them in ratio 3:1 (it was 3:1 for each event) to train set and test set. From the training set we get 1) the weight of each word (IDF value), 2) the most valuable words in each event and 3) the most important thing - a dividing threshold. Cosine values which are between zero and threshold are marked as topic mismatch and values above the threshold are marked as topically cohesive. When experimenting with the test set we don't need to go all over through all the documents, we just take two and using the learned knowledge we determine whether they share the topic or not.

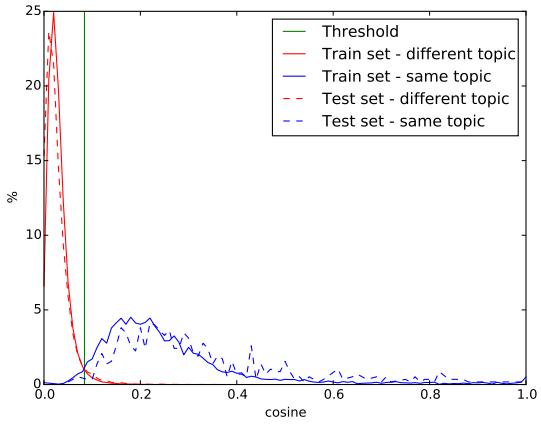


Figure 2. This chart shows the frequency of cosine values according to the result of the comparison (red - different / blue - same topic in a pair). At the intersection of the two curves there is a place with the smallest error (the smallest % of *false alarms* and *missed detection*) - desired threshold (green line). Dashed line shows values obtained with a testing set.

There were close to 350 thousand comparisons when training with the train set and 31 thousand comparisons when applying the information with the test set (the results can be seen in figure 2). There are two possible mistakes: *false alarm* meaning marking two documents with different topic as topically similar; *missed detection* is the opposite: marking two documents with the same topic as a mismatch. The balance of both types and pure success rate can be seen in figure 3.

The error was calculated by the following equation [5]:

$$C_{det} = (C_{Miss} * P_{Miss} * P_{Target} + C_{Fa} * P_{Fa} * (1 - P_{Target})) \quad (2)$$

C_{miss}, C_{fa}	weights of errors of the two types
P_{miss}, P_{fa}	probability of occurrence (obtained from results)
P_{target}	mean probability for a document to find another of the same topic

The resulting value is normalized by the smaller of two values: a) saying "the same" or b) saying "different" to every comparison. We got the success rate between 83-90% depending on the input factors (train/test set ratio, total amount of topics, ...).

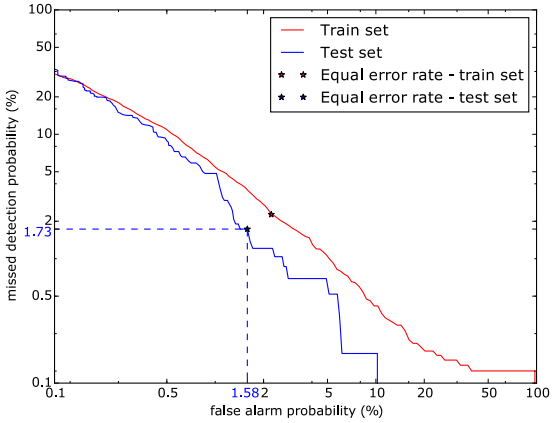


Figure 3. DET curve showing dependency of *missed detection* and *false alarm* on the sliding threshold. With decreasing amount of MD, the amount of FA increases. The point, where the two values are the most alike is called Equal error rate. The closer it is to 0, the better the result values from comparisons are separated (same / different topic). This place is basically where the threshold should be.

Story segmentation results (figure 4) were harder to interpret. Our test set consisted of 16 news reports (approximately one hour of speech) put together, one after another. Then we searched for boundaries (figure 5). We raised the threshold from 0 to 1 by 0.01 step and took only boundaries with cosine value below the threshold. For each threshold we noted amount of missed boundaries, extra boundaries and weighted sum of these numbers. The fact that it's weighted gives us the possibility to choose what should be the output (could be finding all predefined boundaries, but for the price of having some extra). We achieved missing 20% (3/15) boundaries at the price of 2 (3% of potential places) extra.

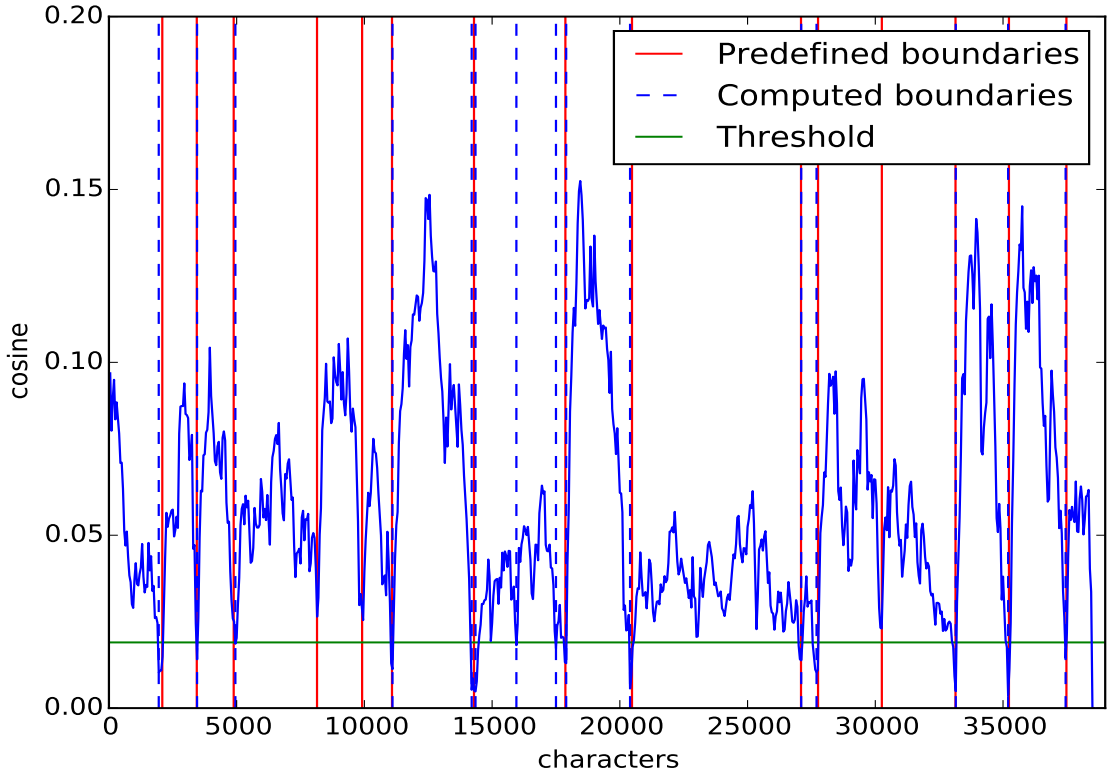


Figure 5. Chart showing cosine values between sentences. All minimums are potential boundaries, but only those below the vertical line are taken. There can also be seen some distinctly separated columns (by a noticeably lower minimums) representing single topics.

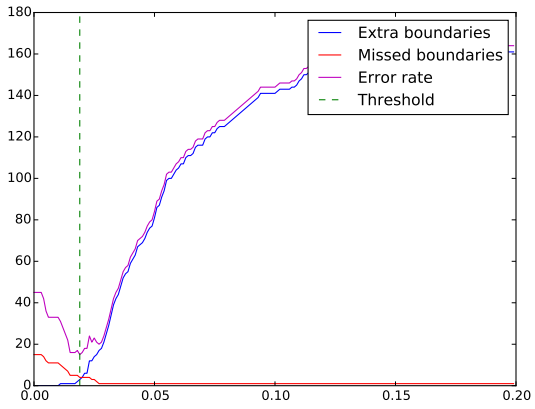


Figure 4. This chart shows the relationship of decreasing number of missed boundaries, increasing number of extra boundaries and final error curve. We chose missed boundary to have three times greater weight that an extra boundary.

5. Conclusions

In this paper we introduced the problem *topic detection and tracking* and our approach with the use of TF-IDF. We also showed our solution of the *story segmentation* problem.

Experimenting with comparisons on TDT5 gave us mean success rate 87%. The story segmentation implementation 1) was able to find 80% of predefined boundaries with only a few extra ones or 2) it found all of them, but with 14% (11 of 78 potential places) extra. However, some of the extra could be connected together (run a second round of boundary finding - but with already connected parts).

One of the goals was to show that natural language processing might really ease people's lives. It could be by filtering email or sorting news on the internet; many of which could be automatized and therefore save time and energy. The methods described also access more sophisticated searching, not only by several words.

One of the main initiatives for the origin of this work was its practical application on recorded courses at FIT BUT. There are many things to prepare for, like processing of the Czech language or an imperfect transcript of spoken language. Still, the base works well and during the next months the final application should be ready.

Acknowledgements

I would like to thank my thesis supervisor Ing. Igor Szöke, PhD. for many supportive consultations and inspirational ideas.

References

[1] Christian Wartena and Rogier Brussee. Topic detection by clustering keywords. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 54–58. IEEE, 2008.

[2] Timothy J Hazen. Mce training techniques for topic identification of spoken audio documents. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(8):2451–2460, 2011.

[3] Martin Riedl and Chris Biemann. Text segmentation with topic models. *Journal for Language Technology and Computational Linguistics*, 27(1):47–69, 2012.

[4] Nist speech group website. http://www.itl.nist.gov/iad/mig//tests/tdt/. Accessed: 2015-03-15.

[5] Jonathan G Fiscus and George R Doddington. Topic detection and tracking evaluation overview. In *Topic detection and tracking*, pages 17–31. Springer, 2002.