Filtering Relevant Text Passages Based on Lexical Cohesion

Mathias Priebe, Clemens H. Cap

mathias.priebe@volkswagen.de, clemens.cap@uni-rostock.de

Abstract: Monitoring news and blogs has become a promising application for global operating groups, who are interested in recognizing topic developments in a fragmented topic landscape. News articles especially long ones may consist of several topics or different aspects of the same topic. In terms of Topic Detection and Tracking (TDT) it is hard to figure out the topic development in a stream of news or blog articles with the scope of a certain information need since articles often contain only a limited amount of the relevant information. In this paper we address the problem of filtering relevant portions of text, commonly known as passage retrieval, by using linear text segmentation methods based on lexical cohesion. We present two strategies for passage retrieval and compare their performance with cohesion based approaches – *TextTiling* (cf. [Hea97]) and *TSF* (cf. [KG09]) – developed in the context of linear text segmentation.

1 Introduction

For many people the Internet became the most important information and news source. In addition to the classic media and since Web 2.0, almost everyone can actively participate in public topics and discussions. This leads to the consequence that we have to deal with a continuous growing stream of news and blog articles which results in the emergence of a fragmented topic landscape. For organizations, especially global operating groups, who are under permanent observation, it is very important to know which topics in a fragmented topic landscape are currently discussed. Further more they are interested in the consequences of the topics. Topic detection and tracking (TDT, cf. [ACD+98]) offers retrieval approaches to organize an incoming stream of news articles. Nevertheless it is hard to figure out the topic development in a stream of news with the scope of a certain information need since articles often contain only a limited amount of the relevant information (e.g. stock market reports, short news).

In this paper we address the problem of filtering relevant portions of text, commonly known as passage retrieval, by using linear text segmentation methods based on lexical cohesion. The purpose of passage retrieval is filtering only those portions of text in a document that correspond to a certain topic of interest. Based on the observation that especially long documents may consist of several topics it is hard to figure out the relevant information associated with a particular information need.

Identifying only the topical relevant text passages in documents a passage retrieval algorithm has to locate the boundaries between the relevant and the irrelevant units of text. This problem is strongly related to the task of linear text segmentation (topic segmenta-

tion). Linear text segmentation can be described as the process of splitting a long text into lexically cohesive fragments of consecutive text fulfilling the following requirements: *(a)* each segment deals with a particular subject or topic and *(b)* adjacent segments deal with different subjects or topics. More precisely it discovers topic boundaries between contiguous segments of text in order to highlight local semantical coherences [LALS08].

2 Passage Retrieval

2.1 Segmentation Strategy

In the following, we present two strategies for passage retrieval using linear text segmentation methods based on lexical cohesion. The first strategy (A) is an application of the traditional linear text segmentation problem that has been widely discussed in the past (cf. [Hea97], [KG09]). It returns the text segment that best matches an information need after a text segmentation algorithm was applied. The closeness of agreement between a segment and the information need is simply calculated by counting the number of keywords the current text passage contains.
The second strategy (B) initially tries to find the approximate closeness to a region in the document that best matches the information need and subsequently identifies the adjacent boundaries around the current position. Estimating that region, we apply a simple heuristic: For every sentence in the document a score is computed given by the number of keywords the current sentence contains and weighted by its total number of terms. Afterwards, for every sentence a region score is calculated by summing up the individual scores of each sentence in the close proximity weighted by their distance to the current sentence. The region size is equal to the minimum segment size M which is provided by the user. The sentence with the highest region score marks the center of the passage that probably corresponds to the users interest. Starting from this position, the algorithm tries to localize the next probable preceding and succeeding boundary.

2.2 Segment Representation

For sentence and segment representation we use the common vector space model (VSM, cf. [SWY75]). A sentence is defined as a *bag-of-words*, whereby each term (word, feature), which occurs in that sentence, is weighted according to a weighting scheme (e.g. $TFIDF$). Formally let N be the number of preselected features, which span the feature space $F = (f_1, ..., f_N)$. Each sentence s is transformed into a feature vector $\vec{s} = (w_1, ..., w_N)$ where w_i is the corresponding weight of feature f_i in $\vec{s}$. Depending on the level of abstraction, a text segment S can either be interpreted as a set of sentence vectors $S = \{\vec{s_1}, \vec{s_2}, ..., \vec{s_k}\}$ or as a single segment vector $\vec{S} = (w_1, ..., w_N)$ containing the averaged term weights of its sentences. $TFIDF$ is used as the prevailing technique for term weighting. The weight w_i for a term f_i within a sentence s_j is the combination of its normalized frequency ($TF_{i,j}$) and its inverse document frequency (IDF_i) given by

the logarithm of the quotient of the number of documents $|D|$ divided by the number of documents containing f_i (cf. Eq. 1).

$$TFIDF_{i,j} = \quad TF_{i,j} \cdot IDF_i = \quad \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|D|}{|d : f_i \in D| + 1} \tag{1}$$

2.3 Measuring Lexical Cohesion

A text is basically made up of a group of sentences that commonly form blocks of cohesive units. A text block can be considered as lexically cohesive if all sentences in that block concern the same topic indicated by word repetition and semantic connectedness. Former research has shown that this observation is useful for detecting boundaries between text segments (cf. [Hea97], [KG09]). In terms of linear text segmentation this means that an area of low lexical cohesion between fragments of high lexical cohesion indicates a semantic break or boundary.
Hearst [Hea97] and Kern & Granitzer [KG09] use different concepts to locate these areas but share the same intuition that sentences within a cohesive segment tend to be similar. For that reason they calculate a lexical cohesion score for each boundary candidate (generally the end of a sentence) based on the similarity between the preceding and succeeding block around the current position. According to the vector space model, the segment or block similarity can be computed in two different ways, depending on their representation. If a block is represented as a set of sentence vectors, the segment similarity can be calculated as the mean pairwise sentence similarity $sim^{MPS}(S_a, S_b)$ between the sentences of both segments using the cosine measure $sim^{COS}(\vec{s_{a,i}}, \vec{s_{b,j}})$ (cf. Eq. 2). If a block is represented as one segment vector, the similarity is given by just calculating the cosine similarity between both segment vectors $sim^{COS}(\vec{S_a}, \vec{S_b})$ (cf. Eq. 3).

$$sim^{MPS}(S_a, S_b) = \quad \sum_{i=1}^{|S_a|} \sum_{j=1}^{|S_b|} \frac{sim^{COS}(\vec{s_{a,i}}, \vec{s_{b,j}})}{|S_a| \cdot |S_b|} \tag{2}$$

$$sim^{COS}(\vec{S_a}, \vec{S_b}) = \quad \frac{\sum_{i=1}^{N} w_{a,i} \times w_{b,i}}{\sqrt{\sum_{i=1}^{N} w_{a,i}^2} \times \sqrt{\sum_{i=1}^{N} w_{b,i}^2}} \tag{3}$$

While Hearst's [Hea97] *TextTiling (TT)* algorithm simple computes the cosine similarity between both segment vectors, Kern & Granitzer's [KG09] *TSF* algorithm puts the mean inner similarity within both blocks into relation with their mean outer similarity to extrapolate the cohesion between them (cf. Eq. 4-7). The consequent score of the *TSF* algorithm can be interpreted as the dissimilarity of the two blocks around the current position (cf. [KG09]). A dissimilarity score greater than zero indicates a potential boundary.

$$score^{TT}(S_a, S_b) = \quad sim^{COS}(\vec{S_a}, \vec{S_b}) \tag{4}$$

$$score^{TSF}(S_a, S_b) = \quad \frac{sim^{in} - sim^{out}}{sim^{in}} \tag{5}$$

$$sim^{in}(S_a, S_b) = \quad \frac{sim^{MPS}(S_a, S_a) + sim^{MPS}(S_b, S_b)}{2} \tag{6}$$

$$sim^{out}(S_a, S_b) = \quad sim^{MPS}(S_a, S_b) \tag{7}$$

2.4 Boundary Candidate Selection

For each boundary candidate we calculate its cohesion score by using one of the scoring functions of Hearst [Hea97] and Kern & Granitzer [KG09] described in the previous section. We separate the text at each boundary candidate into two adjacent blocks of sentences: One block that precede the current position and one block that succeed the current position. The block size is equal to the minimum segment size M introduced in Section 2.1. Both methods applied in the context of our retrieval task identify a boundary by plotting the cohesion score between the two adjacent blocks.
Hearst's *TextTiling* algorithm determines the strength of the decrease (depth score) of the cosine similarity scores by summing up the distance from the peaks around the current boundary candidate. The candidate is selected as a boundary if there is no higher decrease that exceeds a threshold value Θ within a window equal to the minimum segment size M. The threshold is given by the difference of the mean μ and the standard deviation σ of their depth scores. A more conservative measure resulting in a higher precision but lower recall can be chosen by setting the threshold $\Theta = \mu - \sigma/2$ (cf. [Hea97]).
Kern & Granitzer's *TSF* algorithm identifies peaks in the ratio between inner segment similarity and outer segment similarity (dissimilarity). If a peak, resulting in a high inner segment similarity but low outer segment similarity, exceeds a predefined threshold Θ, the current position is marked as a boundary candidate. A candidate is selected as a boundary if there is no higher score for the next sentence positions within a window equal to M (cf. [KG09]). Preventing another parameter we use Hearst's threshold function.

3 Evaluation

3.1 Test Dataset Generation

For algorithm evaluation we follow the common method of creating synthetic test collections. In general, the creation of a consistent "gold standard" as a reference is a very complex and time consuming task. Frequently, human decisions result in subjectivity because humans do not always agree where boundaries should be placed or how fine grained an analysis should be [PH02]. In the context of discourse segmentation it has also been shown that human judgments are notoriously inconsistent [PL93]. To circumvent the problem of subjectivity and inconsistency, we create an artificial collection of test documents by randomly concatenating several distinct stories. Now boundaries are explicitly given as the position between two adjacent news stories.
Our benchmark consists of three different samples (I, II, III) based on a corpus of 200 distinct stories gathered from German stock market reports. The corpus contains 100 short stories with three up to six sentences and 100 longer stories with seven up to fifteen sentences. A sample is a set of 200 randomly generated test documents as the result of a random concatenation of a varying amount of text segments from our corpus. Sample I is characterized by a set of test documents that are composed of a random selection of five up to ten short stories from the corpus. Documents from Sample II consist of two up to

five concatenated longer stories. Sample III is a mixture of documents with a minimum of two and a maximum of ten randomly selected stories.

3.2 Procedure and Evaluation Metrics

Our benchmark procedure operates as follows: For every test document within the sample we simulate the search of every segment or story in that document. Assume we have a document containing three distinct stories. Each story is associated with a set of keywords, comparable to a query. The query as the description of a certain information need is used to simulate the search for the current story. As a result of the search, the algorithm returns a coherent text segment which we compare with the expected story the algorithm should deliver. In order to evaluate the performance, *precision (prec)* and *recall (rec)* are used to measure the retrieval performance. Precision is defined as the fraction of retrieved sentences that are correctly marked as relevant while recall is the fraction of the retrieved sentences that are relevant to the query. Obtaining a robust measure we averaged the scores of the individual results calculated for each of the 200 test documents.

3.3 Experimental Results

Parameter Settings
For performance comparison both segmentation strategies (A, B) in interaction with the *TextTiling* and *TSF* algorithm were evaluated. For every test sample (I, II, III) a separate evaluation run was applied. All parameter settings were the same within each evaluation run to ensure the comparability of the results. We applied the *Maximum Entropy Part-Of-Speech Tagger* [TM00] and a German Stemming algorithm [Cau99] in order to consider only the meaningful terms. Therefore the sentence vector transformation based only on the most frequent nouns, named entities, verbs and adjectives. Measuring the influence of the feature space F on the retrieval task we also varied its size $|F|$ $(100 - 800)$. The window size, equal to minimum segment size M, is the only parameter of the segmentation algorithms that has to be chosen. Obtaining the best results the minimum segment size was individually adjusted depending on the test sample (Sample I: $M = 2$, Sample II: $M = 5$, Sample III: $M = 3$).

Performance Comparison
Estimating the quality of our approach we set up a baseline algorithm that simply extracts all sentences between the first and the last occurrence of a keyword from the associated query. The linear structure of these test documents automatically results in a precision score equal to 1.00 (100 percent) that should not be overrated. For example in the case in which a document consists of more than one relevant text passage the baseline approach will achieve a lower precision, because the heuristic may also return the irrelevant portions of text between both segments. This case should not occur in our experiments because a test document in our samples only consists of text segments with different topics. Surpris-

ingly in terms of Recall the baseline approach achieves reasonable results (Sample I: 0.66, Sample II: 0.70, Sample III: 0.67).
Nevertheless, both segmentation strategies using the *TextTiling* [Hea97] and the *TSF* [KG09] algorithm outperform the baseline method. Generally, our segmentation strategies demonstrated at most 20 percent improvements over the baseline approach in terms of Recall. Giving clear statements with respect to the quality of the segmentation algorithm will not necessarily be easy. In general, both approaches gained comparable results but *TSF* slightly outperforms *TextTiling*. Basically, the only difference of *TextTiling* and *TSF* is the different measure for computation of lexical cohesion. While *TSF* takes the inner and outer similarity into account for computation *TextTiling* only exploits the outer similarity. It seems that recognizing inner similarity relations improve measuring lexical cohesion (cf. [KG09]). For both test samples different strategies seems to be effective. It emerged

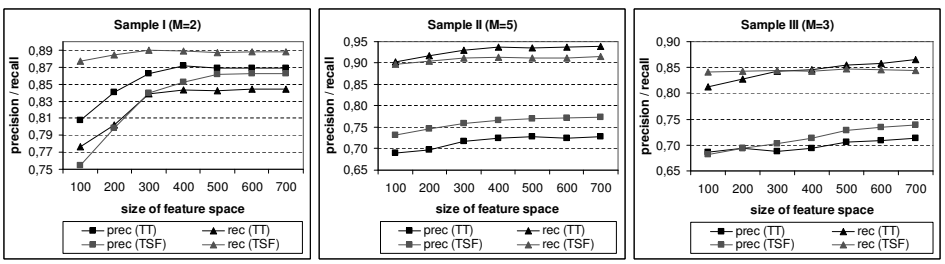

Figure 1: Retrieval performance applying Strategy A.

that Strategy A achieved much better results executed in higher fragmented environments (Figure 1). In contrast, the less fragmented Sample II consists of fewer but longer stories within the document. Strategy B applied on Sample II gained a better performance compared to Strategy A (Figure 2). It seems that Strategy A is more suitable in terms of highly fragmented documents (Sample I & II), whereas strategy B is an eligible alternative in applications with less fragmentation (Sample II). In highly fragmented applications, linear text segmentation algorithms tend to produce more segmentation errors than in less fragmented environments. This leads to the effect that in documents with many small segments segmentation errors are of more consequence.

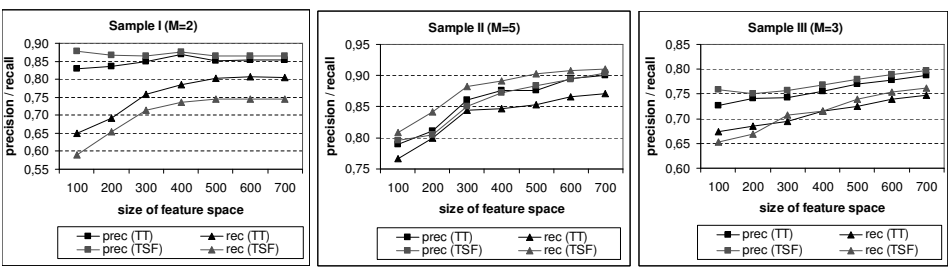

Figure 2: Retrieval performance applying Strategy B.

4 Conclusion and Perspectives

In this paper, we addressed the problem of filtering relevant portions of text in a stream of documents by using lexical cohesion based methods. We proposed two strategies for passage retrieval and compared their performance in interaction with two algorithms developed in the context of linear text segmentation (*TextTiling* [Hea97] and *TSF* [KG09]). In order to evaluate both strategies we created three different artificial test samples consisting of several randomly concatenated text passages from German stock market reports. It has become apparent that both strategies using the *TextTiling* and the *TSF* algorithm outperform the baseline approach and are suitable for solving our passage retrieval problem. In contrast to the baseline, both strategies resulted in a significant increase of recall (at most 20 percent). Finally, there is evidence that traditional IR similarity functions for measuring lexical cohesion reach their limit.
Currently, we are developing a search agent which will use our approach to support organizations in their information accomplishment. In contrast to traditional TDT, the agent organizes an incoming stream of news and blog articles with the focus of a certain user's interest to recognize only the emerging topics related to that information need.

References

[ACD+98] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic Detection and Tracking Pilot Study. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[Cau99] J. Caumanns. A Fast and Simple Stemming Algorithm for German Words. Technical report, Center für Digitale Systeme, Freie Universität Berlin, 1999.

[Hea97] A. M. Hearst. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64, 1997.

[KG09] R. Kern and M. Granitzer. Efficient Linear Text Segmentation Based on Information Retrieval Techniques. In *MEDES'09: Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pages 167–171, 2009.

[LALS08] S. Lamprier, T. Amghar, B. Levrat, and F. Saubion. Using Text Segmentation to Enhance the Cluster Hypothesis. *Artificial Intelligence: Methodology, Systems, and Applications*, pages 69–82, 2008.

[PH02] L. Pevzner and A. M. Hearst. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[PL93] R. J. Passonneau and D. J. Litman. Intention-Based Segmentation: Human Reliability and Correlation with Linguistic Cues. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 148–155, 1993.

[SWY75] G. Salton, A. Wong, and S. C. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[TM00] K. Toutanova and C. D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63–70, 2000.