Sentence Compression for the LSA-based Summarizer

Josef Steinberger *
jstein@kiv.zcu.cz

Karel Jezek*
jezek_ka@kiv.zcu.cz

Abstract: We present a simple sentence compression approach for our summarizer based on latent semantic analysis (LSA). The summarization method assesses each sentence by an LSA score. The compression algorithm removes unimportant clauses from a full sentence. Firstly, a sentence is divided into clauses by Charniak parser, then compression candidates are generated and finally, the best candidate is selected to represent the sentence. The candidates gain an importance score which is directly proportional to its LSA score and indirectly to its length. We evaluated the approach in two ways. By intrinsic evaluation we found that the compressions produced by our algorithm are better than baseline ones but still worse than what humans can make. Then we compared the resulting summaries with human abstracts by a standard n-gram based ROUGE measure.

Keywords: Test Summarization, Sentence Compression, Latent Semantic Analysis

1 Introduction

Research and development in automatic text summarization has been growing in importance with the rapid growth of on-line information services. The aim of automatic text summarization is to take a source text and present the most important content in a condensed form in a manner sensitive to the needs of the user and the task. In the past, most of the summarizers were producing extractive summaries at a sentence level. Lately, the summarization field is turning to more complicated tasks such as sentence compression. We developed a sentence extractive method based on the latent semantic analysis where each sentence is assigned by a score that represents its quality - the LSA score [12]. The method has been further enriched by information about anaphoric relations [13]. Here we propose a sentence compression algorithm for the summarizer.

Sentence compression can be linked to summarization at the sentence level. The task has an immediate impact on several applications ranging from summarization to caption generation [6]. Previous data-driven approaches [6, 10] relied on parallel corpora to determine what is important in a sentence. The models learned correspondences between long sentences and their shorter counterparts, typically employing a rich feature space induced from parse trees. In [11] an algorithm based on Rhetorical Structure Theory[1] is proposed. They use the fact that nuclei are more likely to be retained when summarizing than satellites. The output can be then obtained simply by removing satellites. The task is challenging since the compressed sentences should retain essential information and convey it grammatically.

---

* Department of Computer Science and Engineering, University of West Bohemia, Univerzitni 22, 306 14 Plzen

[1] Rhetorical Structure Theory is a descriptive theory about text organization. The theory consists of a number of rhetorical relations that tie together text spans, and a number of recursive schemas specifying how texts are structurally composed in a tree-like representation. Most relations are binary and asymmetric: they tie together a nucleus (central to the writer's goal) and a satellite (less central material).

We present here a simple sentence compression algorithm that works on a clause level. Our aim is to simplify sentences that have many clauses by removing unimportant ones. The first stage is to obtain compression candidates (e.g., sentences to which an original sentence can be compressed). Further, we try to select the best candidate. An ideal candidate preserves the LSA score of the full sentence and has a reasonable compression ratio.

In the next section we outline our LSA-based sentence extraction algorithm. In section 3 we describe the way how sentence compression candidates are obtained. Further, we discuss the influence of the sentence length on its quality score and we define the final formula for assessing the candidates. Section 5 covers experimental part of the research and reveals arising problems. In the end we outline our future plans.

2 LSA-based Sentence Extraction Algorithm

LSA [7] is a technique for extracting the 'hidden' dimensions of the semantic representation of terms, sentences, or documents, on the basis of their contextual use. It has been extensively used for educational applications such as essay ranking [7], as well as for NLP applications including information retrieval [1] and text segmentation [3]. More recently, LSA has been proposed for text summarization in [5].

The approach is the starting point for our own work. The heart of the method is a document representation developed in two steps. The first step is the creation of a term by sentences matrix $A = [A_1, A_2, \ldots, A_n]$, where each column vector A_i represents the weighted term-frequency vector of sentence i in the document under consideration. The vector $A_i = [a_{1i}, a_{2i}, \ldots, a_{ni}]^T$ is defined as:

$$a_{ji} = L(t_{ji}) \cdot G(t_{ji}), \tag{1}$$

where t_{ji} denotes the frequency in which term j occurs in sentence i, $L(t_{ji})$ is the local weighting for term j in sentence i, and $G(t_{ji})$ is the global weighting for term j in the whole document. There are many possible weighting schemes. The best performing combination we evaluated was binary local weight and entropy global weight:

$$L(t_{ji}) = 1, \text{ if term } j \text{ appears at least once in the sentence } i\text{; otherwise, } L(t_{ji}) = 0, \tag{2}$$

$$G(t_{ji}) = 1 - \sum_i \frac{p_{ji} \log(p_{ji})}{log(nsent)}, \tag{3}$$

where $p_{ji} = t_{ji}/g_{ji}$, g_{ji} is the total number of times that term j occurs in the whole document and $nsent$ is the number of sentences in the document. If there are m terms and n sentences in the document we will obtain an $m \times n$ matrix A for the document.

The next step is to apply Singular Value Decomposition (SVD) to matrix A. The SVD of an $m \times n$ matrix A is defined as:

$$A = U \Sigma V^T \tag{4}$$

where $U = [u_{ij}]$ is an $m \times n$ column-orthonormal matrix whose columns are called left singular vectors, $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is an $n \times n$ diagonal matrix, whose diagonal elements are non-negative singular values sorted in descending order, and $V = [v_{ij}]$ is an $n \times n$ orthonormal matrix, whose columns are called right singular vectors.

From a mathematical point of view, applying SVD to a matrix derives a mapping between the m-dimensional space spawned by the weighted term-frequency vectors and the r-dimensional singular vector space. From a NLP perspective, what the SVD does is to derive the *latent semantic structure* of the document represented by matrix A: a breakdown of the original document into r linearly-independent base vectors (which in LSA are called 'topics'). Each term and sentence in the document is jointly indexed by these 'topics'.

An important feature of SVD is that it can capture interrelationships among terms, so that terms and sentences can be clustered on a 'semantic' basis rather than on the basis of words only. Furthermore, as demonstrated in [1], if a word combination pattern is salient and recurring in document, this pattern will be captured and represented by one of the singular vectors. The magnitude of the corresponding singular value indicates the importance degree of this pattern within the document. Any sentences containing this word combination pattern will be projected along this singular vector, and the sentence that best represents this pattern will have the largest index value with this vector. Assuming that each particular word combination pattern describes a certain topic in the document, each singular vector can be viewed as representing such a topic [4] and the magnitude of its singular value representing the degree of importance of this topic.

The summarization method proposed in [5] uses the representation of a document thus obtained to choose the sentences to go in the summary on the basis of the relative importance of the 'topics' they mention, described by the matrix V^T. The summarization algorithm simply chooses for each 'topic' the most important sentence for that topic: i.e., the kth sentence chosen is the one with the largest index value in the kth right singular vector in matrix V^T.

2.1 Sentence Selection by Vector Length

The main problem with the summarization method proposed by Gong and Liu is that it requires to identify as many important 'topics' (right singular vectors) as the number of sentences we want to include in the summary. The result of this requirement is that sometimes a document summary will include sentences about 'topics' which are not particularly important. In addition, the number of dimensions has to be fixed in advance.

In our own approach, we modified Gong and Liu's method in order to remedy these problems. The improvement, introduced in our earlier work [12], was to change the selection criterion to include in the summary the sentences whose vectorial representation in the matrix $\Sigma \cdot V$ has the greatest 'length', instead of the sentences containing the highest index value for each 'topic'. Intuitively, the idea is to choose the sentences with greatest combined weight across all topics, possibly including more than one sentence about an important topic, rather than simply choose a single sentence for each topic.

More precisely: after computing the SVD of a term by sentences matrix, we compute the length s_k of each sentence vector in matrix $\Sigma \cdot V$.

$$s_k = \sqrt{\sum_{i=1}^{r} v_{k,i}^2 \cdot \sigma_i^2}, \tag{5}$$

where s_k is the length of the vector of k'th sentence in the modified latent vector space, and its significance score for summarization too. The level of dimensionality reduction (r) is essentially learned from the data. We then include in the summary the sentences with the highest values in vector s. We showed in previous work [12] that this modification results in a significant improvement over Gong and Liu's method.

3 Identification of Compression Candidates

If we want to compress a sentence we firstly need to identify a set of possible compression candidates (CC). They should preserve the grammaticality and the meaning of the full sentence. For this task we need a parser that can derive a sentence tree structure. We use Charniak parser [2] that is able to mark clauses and catch their dependencies. We describe the identification algorithm on the following sentence[2]:

As the western german population is now aware, German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards until there is a sustained upswing in the east even if at the cost of higher German inflation than the Bundesbank would wish.

The Charniak parser will capture the following tree structure:

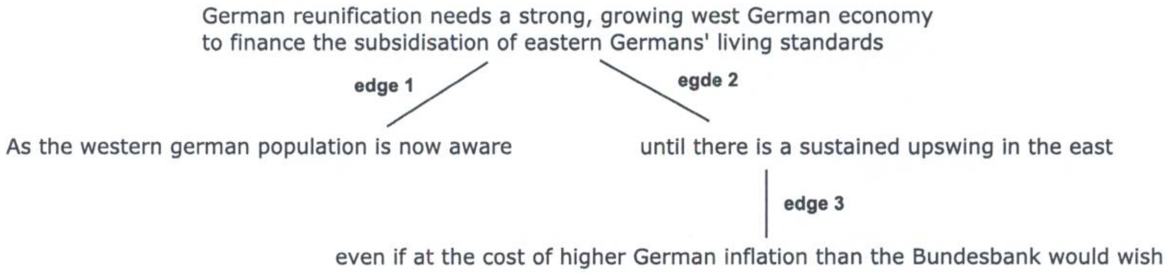
German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards
edge 1
egde 2
As the western german population is now aware
until there is a sustained upswing in the east
edge 3
even if at the cost of higher German inflation than the Bundesbank would wish

Fig. 1: Tree structure of an example sentence.

We can find three edges there. If we cut the tree in an edge we get a compressed sentence where all subordinate clauses of the edge are removed. And more, we can cut the tree more than once - in combination of edges. However, we can not delete dependent edges. If we apply these rules on the example sentence above we obtain the following six compression candidates:

CC1: German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards. (**edge 1** and **edge 2** were removed)

CC2: German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards until there is a sustained upswing in the east. (**edge 1** and **edge 3**)

CC3: German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards until there is a sustained upswing in the east even if at the cost of higher German inflation than the Bundesbank would wish. (**edge 1**)

CC4: As the western german population is now aware, German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards. (**edge 2**)

CC5: As the western german population is now aware, German reunification needs a strong, growing west German economy to finance the subsidisation of eastern Germans' living standards until there is a sustained upswing in the east. (**edge 3**)

The last candidate **CC6** is the full sentence. Other combinations of edges could not be applied due to the rules above.

---

[2] It is a sentence from DUC 2002 corpus, cluster 69, document FT923-6509.

4 Finding the Best Candidate

The summarization algorithm described in 2.1 favours long sentences. It is obvious because a long sentence is more likely to contain significant terms or topics. There is a correlation between sentence length and its summarization value, however, not that strong - 0.6. Regardless, the medium correlation encourages us to normalize the summarization value by the sentence length to avoid the negative correlation effect. We created all compression candidates of the sentences in our corpur. This gave us enough data to show the linear dependency of the summarization score on the sentence length (fig. 2). Notice that when we cut a sentence to 50% a random compression candidate contains approximately 50% of full sentence information, measured by LSA score.

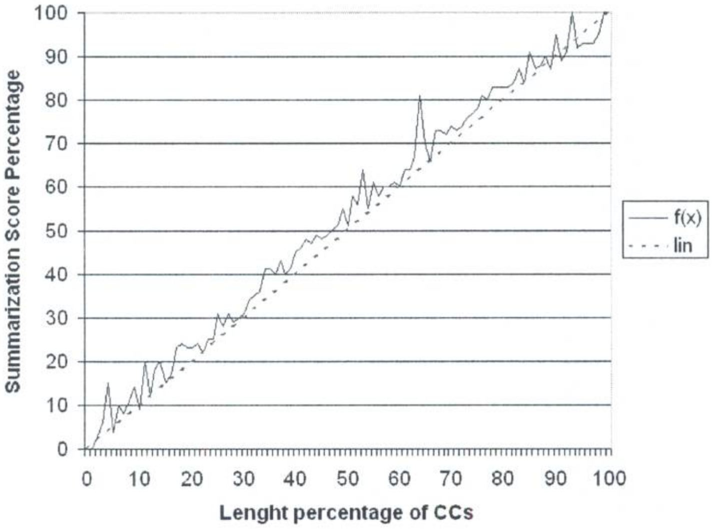

Fig. 2: Dependency of the candidate's summarization score on its length. Both the length and the summarization score are measured proportionally to the full sentence.

Now we can update the summarization score formula (5):

$$s_k = \frac{\sqrt{\sum_{i=1}^{r} v_{k,i}^2 \cdot \sigma_i^2}}{nwords}, \tag{6}$$

where $nwords$ is a number of words in the sentence k. This will measure an average summarization score for a word in a sentence. The difficulty is now that a long significant sentence with many unimportant clauses will be assessed by a low score. However, with compression algorithm that would be able to delete the unimportant clauses we can suppress this negative effect. After getting the compression candidates we assign each a summarization score. Firstly we need to create an input vector for the candidate. The process is the same as when procuding the input matrix A (see section 2). The vector has to be transformed to the latent space[3]:

$$v_q = \Sigma U^T q, \tag{7}$$

where q is the CC's weighted term-frequency vector and v_q is the vector transformed to the latent space.

---

[3] The process is the same as a query is transformed to the latent space in information retrieval.

Now, we can compute a summarization score for each candidate. We substitute the vector v_k by v_q in equation 6. The candidate with the highest summarization score within the sentence candidates is considered to be the best candidate of the sentence. In the example above **CC1** received the highest score. Among the best candidates we select the ones with the highest summarization score for the summary. A full sentence can be selected as well because it is allways among the candidates.

After selecting the best candidate we can find out what percentage of loss in summarization score we get when we compress a sentence. For this task we obtained the best compression candidates of all sentences in our corpus. The next figure shows a dependency of the best candidate's summarization score on the sentence length.

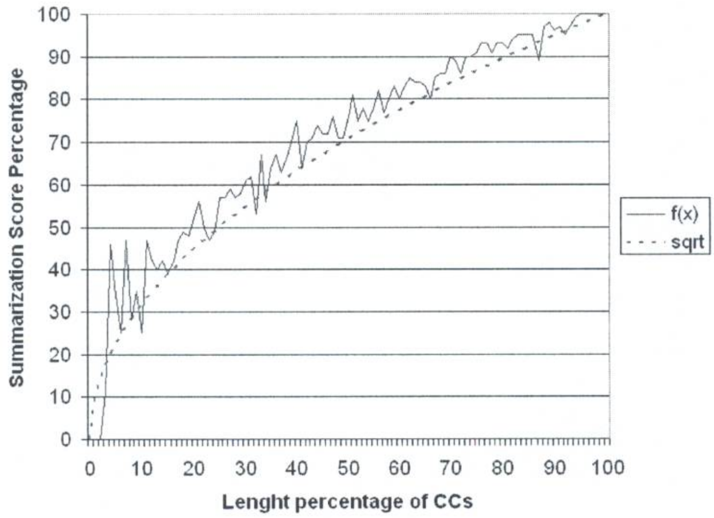

Fig. 3: Dependency of the best candidate's summarization score on its length. Both the length and the summarization score are measured proportionally to the full sentence.

The difference between this graph and the previous one is that here we take into account only the best candidate, however, in the previous we took all. We can observe that this dependency is defined by the square root. It means that when we shorten a sentence to 50% it will contain on average approximately 71% of full sentence information, measured by the LSA score.

5 Experimental Results

Requirements of sentence compression algorithms are to preserve the main content of the sentence and produce a grammatically correct sentence. We made the following experiment. We used the DUC 2002 corpus [9] for the evaluation. The corpus contains newspaper articles from different sources. We randomly selected 20 documents, each from a different cluster. The documents were parsed by the Charniak parser and summaries with compressed sentences were created. We needed to compare our compressions to those made by humans. For each summary sentence one compression by an annotator was written. And more we generated a baseline candidate where bottom-most leaves of the clause tree (see fig. 1) are cut. When we return to the example above **CC5** would be selected. Thus, for each summary sentence three compression candidates were produced - by a human, the LSA score and the baseline.

Three annotators assigned a correctness score from 1 to 5 to each candidate. It received the score 1 when unimportant information was removed, the score 2 when not so important information was missing, the score 3 when some important information was missing, the score 4 when the sense of the sentence was changed and the score 5 where the sentence had no sense. If they found a grammatical problem they marked the candidate as grammatically incorrect. The following table compares the compression approaches. Totally, 74 sentences from 20 documents were analyzed.

Compression approach	% of compressed sentences	% of average compression rate	correctness score (95% conf. interval)	% of grammatically incorrect sentences
Baseline	66.07	58.41	2.883 (2.577 - 3.189)	5.86
LSA score	51.79	63.64	2.193 (1.929 - 2.456)	7.66
Human	43.75	64.60	1.486 (1.363 - 1.610)	0

Tab. 1: A comparison of the compression approaches.

A critical problem in doing the compressions was that sometimes the main object of the sentence was removed. These compressions were assigned by the score 5. We can observe that the compression algorithm based on the LSA score performed significantly better than the simple baseline and significantly worse than humans. It compressed 51.79% of the total number of sentences (the rest of the sentences was left in their full form). When a sentence was compressed it was shortened on average to 63.64% of its original length (measured by the number of words). The correctness was a bit lower than 2. It can imply that on average some information was removed but it was not so important. Notice that even some human compressions were marked 2 - so the annotators advised to remove some information that is not so important for the full sentence sense. A few gramatical problems arose, however, 7% does not represent a serious problem. There were high correlations between annotator assessments - 76%, 81% and 85%.

In the following table we present a standard summarization evaluation by ROUGE [8]. It is a n-gram method that measures the similarity between automatically produced summaries and abstracts written by humans. For this evaluation we used the whole DUC 2002 corpus - totally 567 documents. We compared 3 settings of the system. The first does not sentence compression and uses the formula (5) to assign a score to a sentence. This system was presented in our previous paper [13]. The only difference of the second system setting is that it takes into account the sentence length adjustment - formula (6) but still it extracts full sentences. And finally, the last system use formula (6) and sentence compression.

Summarizer setting	ROUGE-1	ROUGE-2	ROUGE-SU4	ROUGE-L
Formula (5) without compression	0.43209	0.20735	0.16792	0.40079
Formula (6) without compression	0.42663	0.20361	0.16474	0.39832
Formula (6) with compression	0.43774	0.20955	0.16877	0.40879

Tab. 2: The ROUGE score evaluation. ROUGE-1 = a unigram similarity measure, ROUGE-2 = a bigram measure, ROUGE-SU4 = a bigram measure with skip unigrams, and ROUGE-L = a longest common subsequence measure. For detail see [8].

Here we can observe that when we included sentence length to the formula for assigning the sentence quality ROUGE scores came down. It is caused by decreasing the score of significant sentences with many unimportant subordinate clauses. However, when we include sentence compression that is able to identify these clauses the summary quality goes up. Long significant sentences are included but shortened and we can add more sentences to the summary because it is usually limited by the number of words. In our experiment the summary was limited by 100 words. However, there is a chance to include a sentence with different sense from its full form.

6 Conclusions

We presented a simple sentence compression approach for our summarizer that is based on the latent semantic analysis. Evaluation showed that it is able to produce solid compressions although it is still far from perfect. However, sentence compression is a very difficult task. There is a possibility to produce a sentence with different sense from its full form or to make it grammatically incorrect. Our future work in automatic sentence compression will focus on fixing the missing object problem. The task will be to add more resources because the LSA score is not sometimes enough. More, we plan to go deeper into the sentence structure to be able to identify not only unimportant clauses but also phrases or terms.

We thank our colleagues for their great annotation work.

Bibliography

1. M. W. Berry, S. T. Dumais, and G. W. O'Brien: *Using linear algebra for intelligent IR*, SIAM Review, 37(4), 1995.
2. E. Charniak: *A maximum-entropy-inspired parser.* Proceedings of NAACL, Philadelphia, US, 2000.
3. F. Y. Y. Choi, P. Wiemer-Hastings, and J. D. Moore: *Latent semantic analysis for text segmentation.* Proceedings of EMNLP, Pittsburgh, US, 2001.
4. C. H. Q. Ding: *A probabilistic model for latent semantic indexing.* Journal of the American Society for Information Science and Technology, 56(6), 597–608, 2005.
5. Y. Gong and X. Liu: *Generic text summarization using relevance measure and latent semantic analysis.* Proceedings of ACM SIGIR, New Orleans, US, 2002.
6. K. Knight and D. Marcu: *Summarization beyond sentence extraction: A probabilistic approach to sentence compression.* Artificial Intelligence, 139(1):91–107, 2003.
7. T. K. Landauer and S. T. Dumais: *A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge.* Psychological Review, 104, 211–240, 1997.
8. C. Lin and E. Hovy: *Automatic evaluation of summaries using n-gram co-occurrence statistics.* Proceedings of HLT-NAACL, Edmonton, Canada, 2003.
9. NIST: *DUC 2002 corpus.* http://www-nlpir.nist.gov/projects/duc/data.html.
10. S. Riezler, T. H. King, R. Crouch and A. Zaenen: *Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar.* Proceedings of HLT/NAACL, Edmonton, Canada, 2003.
11. C. Sporleder and M. Lapata: *Discourse chunking and its application to sentence compression.* Proceedings of EMNLP, Vancouver, Canada, 2005.
12. J. Steinberger and K. Jezek: *Text summarization and singular value decomposition.* Proceedings of ADVIS, Izmir, Turkey, 2004.
13. J. Steinberger, M. A. Kabadjov, M. Poesio and O. Sanchez-Graillet: *Improving LSA-based summarization with anaphora resolution.* Proceedings of EMNLP, Vancouver, Canada, 2005.