Widening the HolSum Search Scope

Martin Hassel and Jonas Sjöbergh
KTH CSC
{xmartin, jsh}@kth.se

Abstract

We investigate different areas of the high-dimensional vector space built by the automatic text summarizer HolSum, which evaluates sets of summary candidates using their similarity to the original text. Previously, the search for a good summary was constrained to a very limited area of the summary space. Since an exhaustive search is not reasonable we have sampled new parts of the space using randomly chosen starting points. We also replaced the simple greedy search with simulated annealing. A greedy search from the leading sentences still finds the best summary. Finally, we also evaluated a new word weighting scheme: the standard deviation of word distances, comparing it to the previously used $tf \cdot \log(idf)$ weighting. Different weighting schemes perform similarly, though the term frequency contributes more than other factors.

1 Language Independent Automatic Text Summarization

Today there is much research in automatic text summarization that is focused on knowledge-rich, and in practice language specific, methods. Methods using tools and annotated resources simply not available for many languages. Justifiably so, these knowledge-rich systems do in general perform better than earlier knowledge-poor approaches. It is however easy to see that there is a clear need for automatic summarization also for languages less in focus in this research area than the major European, Asian or Mid-Eastern languages.

One such attempt to develop a method for largely language independent automatic text summarization resulted in the HolSum summarizer (Hassel and Sjöbergh, 2005; Hassel and Sjöbergh, 2006), which can be implemented quickly using only a few very basic language resources. HolSum tries to capture the essence of a document being summarized by building a document space where a set of summary candidates can be evaluated against the original text. The HolSum summarizer thus takes the theoretically appealing approach of trying to optimize semantic similarity between the generated summary and the text being summarized, rather than lexical and syntactic similarity which many other systems and metrics do.

In this paper we evaluate several modifications to the HolSum approach, including changing the word space used and the search strategy for finding a good summary in the space of possible summary candidates.

2 Word Spaces

Word space models, most notably Latent Semantic Analysis (Deerwester et al., 1990; Landauer et al., 1998), enjoy considerable attention in current research on computational semantics. Since its introduction in 1990 Latent Semantic Analysis (LSA) has more or less spawned an entire research field. A wide range of word space models has since been developed, as well as numerous publications reporting exceptional results on many different tasks, such as information retrieval, various semantic knowledge tests, text categorization and word sense disambiguation.

The general idea behind word space models is to use statistics on word distributions in order to generate a high-dimensional vector space. In this vector space the words are represented by context vectors whose relative directions are assumed to indicate semantic similarity. The basis of this assumption is the *distributional hypothesis* (Harris, 1968), according to which words that occur in similar contexts also tend to have similar properties (meanings/functions). From this follows that if we repeatedly observe two words in the same (or very similar)

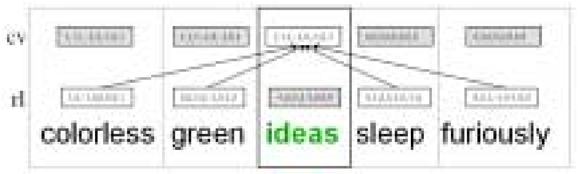

Figure 1: A Random Indexing context window focused on the token "ideas", taking note of the co-occurring tokens. The row marked as "cv" represents the continuously updated *context vectors* and the row marked as "rl" the static *random labels* (acting as addable meta words). Grayed out fields are not involved in the current token update.

contexts, then it is not too far fetched to assume that they also mean similar things (Sahlgren, 2006).

2.1 Random Indexing

In the HolSum summarizer the Random Indexing (Sahlgren, 2005) method is used to build a semantic vector space. This vector space is then used to choose a summary as close to the original text as possible from a set of summary candidates. Random Indexing (RI) presents an efficient, scalable and inherently incremental alternative to standard word space methods. As an alternative to LSA-like models that first construct a huge co-occurrence matrix and then perform the dimension reduction, Random Indexing instead accumulates context vectors continuously based on the occurrence of words (tokens) in contexts, without a need for a separate dimension reduction phase.

The construction of context vectors using RI can be viewed as a two-step process. First, each context (often each co-occurring word is considered a context) in the data is assigned a unique and (usually) randomly generated label. These labels can be viewed as sparse high-dimensional ternary vectors. [1] Their dimensionality (d) is usually chosen to be in the range of a couple of hundred up to several thousands, depending on the size and redundancy of the data you are working with. The labels consist of a very small number (usually about 1-2%) of randomly distributed +1s and -1s, with the rest of the elements of the vectors set to 0.

Next, the actual context vectors for the words are produced by scanning through the text and each time a token w occurs in a context (e.g. in a document or paragraph, or within a sliding context window), that context's d-dimensional random label is added to the context vector for the token w. Thus, when using a sliding context window all tokens that appear within the context window contribute (to some degree) with their random labels to w's context vector. Words are then represented by d-dimensional context vectors that are the sums of the random labels of the co-occurring words, see Figure 1. When using a sliding context window it is also common to use some kind of distance weighting in order to give more weight to tokens closer in context.

One of the strengths of Random Indexing is that we can in a very elegant way fold the document currently being processed into the Random Index, thus immediately taking advantage of distributional patterns within the current document. This removes the problem of lack of data due to unknown words, since all words in the text will have been seen at least once (when the text itself was added to the Random Index). We also have a system that learns over time. Sparse data is still something of a problem though, since a never before seen word will only have as many contextual updates as the number of times it occurs in the current document. This is however better than no updates at all.

As with LSA-like models, for good performance Random Indexing needs large amounts of text (millions of words) when generating the conceptual representations. Since Random Indexing is resource lean and only requires access to raw (unannotated) text, this is generally not a problem.

3 The HolSum Summarizer

3.1 Evaluating Candidate Summaries

HolSum makes use of Random Indexing to differentiate between different summaries. Random Indexing gives each word a context vector that in some sense represents the semantic content of the word. We make use of these vectors when calculating a measure of similarity between two texts. Each text is assigned its own vector for semantic content, which is simply the (weighted) sum of all the context vectors of the words in the text. This can be seen as projecting the texts into a high-dimensional vector space where we can relate the texts to each other. Similarity between two texts is then measured as the similarity between the directions of the semantic vectors of the texts, in our case between the vector for the full text and the vectors for each of the candidate summaries.

When constructing the semantic vector for a text, the context vector for each word is weighted with the term frequency and some measure of topicality

[1]The extremely sparse random labels are handled internally as short lists of positions for non-zero elements and are generated on the fly whenever a never before seen token is encountered in the context during indexing.

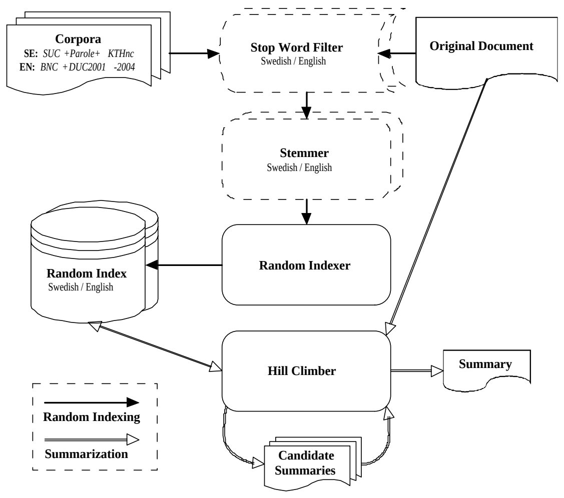

Figure 2: HolSum system layout. The candidate summaries are iteratively generated and evaluated (i.e. compared for semantic similarity against the original document).

(e.g. the inverse document frequency). If desired, other weighting criteria can easily be added, for instance for slanted or query based summaries where some words are deemed more important, or by giving words occurring early in the document, in document or paragraph headings etc. higher weight.

3.2 Finding a Better Summary

To find a good summary we start with one summary and then try to see if there is another summary that is "close" in some sense that is also a better summary. Better in this context means more similar to the original text, which is measured as described in the previous section. The reason we do not exhaustively pursue the best summary of all possible summaries is that there are exponentially many possible summaries. Comparing all of them to the original text would thus not be feasible.

It has been shown that the leading sentences of an article, especially within the news domain, are important and constitute a good summary (Edmundson, 1969; Brandow et al., 1995). Therefore, the "lead" summary, i.e. the first sentences from the document being summarized up to a specified length, was used in our experiments both as a baseline and as one of the starting points in our search for a better summary.

Using a standard hill climbing algorithm we then investigate all neighbors looking for a better summary. The summaries that are defined as neighbors to a given summary are simply those that can be created by removing one sentence and adding another. Since sentences vary in length we also allow removing two sentences and adding one new, or just adding one new sentence. This allows for optimizing the summary size for the specified compression rate.

When all such summaries have been investigated, the one most similar to the original document is updated to be the currently best candidate and the process is repeated. Any summary that is too short or too long (the wanted compression rate is given as a parameter to the program) is heavily penalized. Otherwise, the summaries tend to grow longer, since including more of the original text will make the summary more similar to it, and eventually include the whole text.

If no other summary is better than the current candidate the search is terminated. It is also possible to stop the search at any time if so desired and return the best candidate so far. A schematic layout of the complete system can be found in Figure 2.

In our experiments on the texts provided for the Document Understanding Conferences (DUC) the generated summaries are very short, about three sentences. This means that there are usually quite few, typically around four, search iterations. Some documents require quite many iterations before a local maximum is found, but these constitute a fairly small amount of the texts in the data set.

4 Evaluation

Even though the HolSum system was designed to be fairly language independent, here we only evaluate it on English. The reason is that large amounts of reference summaries and evaluation schemes have been developed for English. When large amounts of evaluation data is available it makes it easier to detect small effects of changes to a system, such as those we are investigating here. Several other summarization systems also exist for English, and can thus be used as reference points to see if the system performs well or not.

For English we build our conceptual representations for each word based on a large corpus, BNC – the British National Corpus (Burnard, 1995). We also add all the documents that are being summarized. The data used for building these representations is thus comprised of 100 million words from BNC and roughly 2 million words contained in 291 document sets provided for the Document Understanding Conferences 2001–2004 [2]. After stop word filtering and stemming this results in almost 290,000

[2]DUC, the Document Understanding Conferences, http://duc.nist.gov/

unique stems taken from 4,415 documents.

The HolSum summarization method has been evaluated in earlier experiments by Hassel & Sjöbergh (2006), showing promising results on manually written abstracts from the DUC.

For reasons of comparability we have chosen to evaluate using ROUGEeval (Lin, 2003) with the same data and model summaries. The evaluation was carried out by first using all manually created 100 word summaries provided for DUC 2004 as reference summaries, comparing our results to previous results on the same data set (Over and Yen, 2004; Hassel and Sjöbergh, 2006). Having reached a reasonable level of success we then compared against the complete set of human written 100 word summaries from DUC 2001–2004 in order to verify our method on a larger test set.

The evaluation has been carried out by computing ROUGE scores on the system generated summaries using the manual summaries from DUC as reference summaries. The ROUGE score is a recall based n-gram co-occurrence scoring metric that measures content similarity by computing the overlap of n-grams occurring in both a system generated summary as well as a set of model summaries. ROUGE scores have tentatively been shown to correlate with human evaluation (Lin and Hovy, 2003). As in DUC 2004, we have throughout the evaluations used ROUGEeval-1.4.2 with the following settings:

```
rouge -a -c 95 -b 665 -m -n 4 -w 1.2
```

In our experiments ROUGE scores are in the case of DUC 2004 calculated over 114 system generated summaries, one for each document set, and in the case of DUC 2001–2004 for 291 summaries. For reference, a human agreement score, see Table 1, has been calculated. For each document set ROUGE scores for each human written summary was calculated by treating the summary as a system summary and comparing it to the remaining human written ones. The mean value was then used. On average there are four human written summaries available for each set. Also, we evaluate a baseline (lead), which is the initial sentences in each text up to the allowed summary length.

We then generated a summary of each text in the data set and evaluated them compared to the 100 word reference abstracts provided. The length of these system generated summaries was allowed to vary between 75 and 110 words. We also evaluated the impact of the dimensionality chosen for the Random Indexing method by running our experiments for three different values for the dimensionality, building semantic representations using 250, 500 and 1000 dimensions. Our results show little variation over different dimensionalities though. For each dimensionality we also calculated the mean performance using ten different random seeds, since there is a slight variation in how well the method works with different random projections.

4.1 Keywords Come in Bursts

When constructing the semantic vector for a text the context vector for each word is weighted with the importance of this word by simply making the length of the vector proportional to the importance of the word. The weight could for instance be something simple, such as making the length of the vector be $tf \cdot \log(idf)$ as used in previous HolSum evaluations, i.e. the term frequency and inverse document frequency. The term frequency is the frequency of the term within the given document and gives a measure of the importance of the term within that particular document. The inverse document frequency, on the other hand, is a measure of the general importance of the term, i.e. how specific the term is to said document (Salton and Buckley, 1987).

In addition to the highly traditional $tf \cdot \log(idf)$ weighting scheme, we have also experimented with utilizing the “burstiness” of a word for term weighting. Ortuño et al. (2002) have shown that the spatial information of a word, i.e. the way in which it is distributed in the text (independently of its relative frequency), is a good measure of the relevance of the word to the current text.

The burstiness of a word is here based on the standard deviation of the distance (in words) between different occurrences of this word in the text. Words that occur only with large distances between occurrences usually have a high standard deviation by chance, so the standard deviation is divided by the mean distance between occurrences. The final weight of a word is thus:

$$tf \cdot \frac{\sigma}{\mu}$$

where μ is the mean and σ the standard deviation of the distances between occurrences, in words.

Here too we have evaluated on three different dimensionality choices, 250, 500 and 1,000. Generally, as low dimensionality as possible is desirable, since processing time and memory usage is then lower. In Table 1 it can be seen that the variation between different dimensionalities is quite low. It is largest for $tf \cdot \log(idf)$, where the mean value for dimensionality 250 is 32.0 and the mean value for 1,000 is 32.4 in the DUC 2001–2004 data set. This is nice, since it seems

	DUC 2004	DUC 2001–2004
Baseline: lead	31.0	28.3
Human	42.6	39.7
$tf \cdot \log(idf)$, 1000	34.1	32.4
$tf \cdot \log(idf)$, 500	34.2	32.3
$tf \cdot \log(idf)$, 250	33.9	32.0
Burstiness, 1000	33.9	32.2
Burstiness, 500	33.7	32.1
Burstiness, 250	33.6	31.9

Table 1: ROUGE-1 scores for different dimensionality choices of the context vectors. There are 114 documents from DUC 2004 and 291 from DUC 2001–2004.

to be unimportant to spend a lot of time optimizing the choice of this parameter.

For each choice of dimensionality the mean performance using ten different random seeds was calculated. The impact of the randomness of the method seems larger than the impact of the dimensionality choice. The largest variation was for the dimensionality 500, spanning 33.1–34.3 in ROUGE-1 scores for the DUC 2004 data set. Variations for the other dimensionalities were slightly less.

These results are unsurprisingly worse than those of the best systems of DUC 2004, which had ROUGE-1 scores of about 39. The top systems included summarizers using more advanced tools than those made available to HolSum, such as co-reference resolution or genre specific extraction patterns. Such tools seem to be quite useful, but since the HolSum system was meant to be language independent it uses only tokenization, stopword removal and stemming. The HolSum scores are however better than about half of the systems and well above the baseline.

One system (Jaoua et al., 2003; Jaoua et al., 2004) participating in the DUC 2004 used an approach similar to that of HolSum. A genetic algorithm was used to search through the space of possible extracts and coverage of high frequency words in the original text was used to rank summary candidates. The system achieved quite high ROUGE scores (higher than HolSum).

It can be noted that improving the ROUGE scores of HolSum is quite easy. Since the recall based measurements are never made worse by adding more words, simply making the summaries longer so as to fill up the allowed 100 words more fully gives higher scores. For ROUGE-1 using dimensionality 250 the scores are improved from 33.9 to 34.4 (DUC 2004) and from 32.0 to 32.7 (DUC 2001–2004) by simply always generating summaries of at least 100 words.

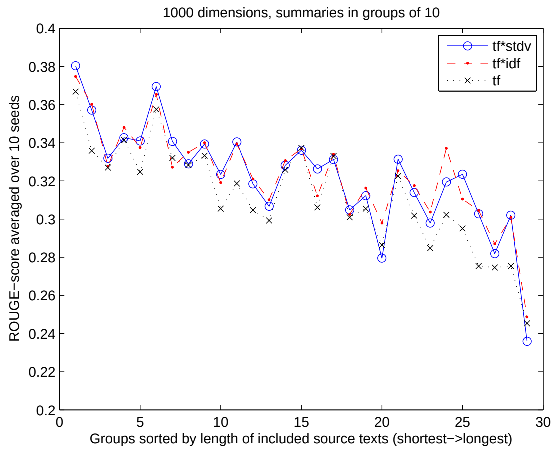

Figure 3: ROUGE-1 scores for three weighting schemes, divided into 29 groups of 10 summaries each sorted by compression rate. The leftmost group contains the summaries for the 10 shortest source texts while the rightmost group contains the summaries for the 10 longest.

Since we were not particularly interested in generating higher ROUGE scores without making better summaries, we did not use such tricks in the evaluations, though.

The choice between $tf \cdot \log(idf)$ or burstiness seems to have very little impact, the results are nearly identical in ROUGE-1 scores. This is further supported when plotting a graph showing the ROUGE scores for three different weighting schemes. The first weighting scheme is burstiness weighting, the second is $tf \cdot \log(idf)$ and the third is weighting only by the term frequency. In Figure 3 we can see that it is the term frequency that is pulling the most weight and that the inverse document frequency and the standard deviation seem to add roughly the same improvement.

Removing the term frequency weighting lowers the performance substantially. A small test using dimensionality 250 and burstiness weighting but no term frequency weighting gave a ROUGE-1 score of 30.5 (DUC 2004) and 29.3 (DUC 2001–2004), compared to 33.6 and 31.9 using both term frequency and burstiness.

It should however not come as much of a surprise that the term frequency has the most impact during the accumulation of the context vectors. Since we apply stop word filtering prior to this step we have already filtered out most of the highly frequent function words. This means that the remaining high frequency words are content words and as such good descriptors of the document being summarized.

In Figure 3 we can also see that summarizer performs best at low compressions rates. This is due to the fact that the more of the source text that is included in the summary, the higher the chance of selecting sentences with words also used in the human written summaries in the gold standard.

4.2 Widening the Search Space

One thought that immediately strikes you is that there might be better summaries, according to the given criteria, out there in summary space. It might simply be the case that these remain unfound when going down the path of always choosing the best neighbor. What if beyond one of the lesser neighbors lies an even better summary?

The method we used for investigating this theory is simulated annealing (Kirkpatrick et al., 1983), augmented with back-off heuristics. Instead of in each step choosing the best neighbor as our next transition point we may go to a randomly chosen neighbor, as long as it is better than the current summary. However, in doing this we also keep track of the best neighbor so far, and in the case that we are lead too far down a garden path,[3] we can always go back to the best neighbor previously visited and start our search anew. A ban list containing all visited summaries, excluding the best summary so far, effectively hinders us from going down the same path again (not that it would have mattered much, bar computing time). This means that the annealing procedure will always perform at least on par with the greedy search regarding Random Indexing similarity scores.

With simulated annealing the cooling schedule is of great importance (Laarhoven and Aarts, 1987). The cooling schedule is the factor that in each transition governs the probability of choosing a random better neighbor instead of the best neighbor. Two common formulas for calculating the cooling factor were used in these experiments. The first schedule was calculated using the following formula:

$$T_i = T_0 \left(\frac{T_N}{T_0}\right)^{\frac{i}{N}}$$

In this formula T_i is the probability of choosing a random better neighbor in step i, where i increases from 0 to $N = 100$ transitions. The initial probability T_0 is set to 100% and the lowest allowed probability to $T_N = 5\%$. This schedule starts with a high probability for random behavior and then rapidly reverts to a traditional greedy search. The second cooling schedule, using the same notation as above but with T_N set to zero, was designed to revert to a greedy search more linearly:

$$T_i = T_0 - i\frac{T_0 - T_N}{N}$$

The algorithm was in both cases set to break when no known neighbors are better than the current summary and no previous state or neighbor has been better, in terms of Random Indexing similarity, or the maximum number of 100 transitions has been reached. At this point the best state, current or previously visited, is returned. In most cases the maximum number of transitions was never reached.

	DUC 2004	DUC 2001–2004
Baseline: lead	31.0	28.3
Human	42.6	39.7
Original, 1000	34.1	32.4
Original, 500	34.2	32.3
Original, 250	33.9	32.0
Schedule 1, 1000	34.1	32.4
Schedule 1, 500	34.2	32.3
Schedule 1, 250	33.9	32.0
Schedule 2, 1000	34.2	32.4
Schedule 2, 500	34.2	32.3
Schedule 2, 250	34.0	32.0

Table 2: ROUGE-1 scores for the the two annealing schedules as well as the standard greedy search for reference.

As can be seen in Table 2 the resulting summaries were in almost all cases identical to the summaries generated using the bare greedy search algorithm. In the few cases (7 out of 2,910) where the summaries generated with a dimensionality of 500 differed, the second cooling schedule resulted in slightly higher ROUGE scores than the greedy search, but not enough to warrant the radically added computation time. For the same dimension the first schedule resulted in only one higher scoring summary.

Of course, a formula with a slower descent into a traditional greedy search could be used, but would probably lead to further increased run times. Simulated annealing using the two cooling schedules presented in this paper in general takes about three times as long to generate the 8,730 summaries evaluated in each run[4], compared to the standard greedy search.

[3] In our case ten transitions without finding a new summary that is better than best one seen so far.

[4] In each evaluation run the system generates summaries for 291 documents times 3 dimensionalities times 10 random projections (seeds).

	DUC 2004	DUC 2001–2004
Baseline: lead	31.0	28.3
Human	42.6	39.7
Original, 1000	34.1	32.4
Original, 500	34.2	32.3
Original, 250	33.9	32.0
Rand.sent., 1000	33.2	31.1
Rand.sent., 500	33.0	31.2
Rand.sent., 250	33.1	31.1
Rand.part, 1000	33.1	31.3
Rand.part, 500	33.2	31.3
Rand.part, 250	33.1	31.3

Table 3: ROUGE-1 scores for the two different random starting point strategies as well as the standard lead starting point for reference.

4.3 Different Points of Departure

Considering the approaches above, we have still only investigated a small fraction of the high-dimensional vector space representing all possible summaries. As stated in Section 3.2 it is simply not feasible to exhaustively search all possible summaries in pursuit of the best summary. Another option is to again put the greedy search to use, but this time giving it randomly chosen starting points. The idea here is that there may be better starting points than the leading sentences of the original text, thus taking other paths to possibly better summaries.

We have tried two approaches: the first simply chooses sentences randomly from the source text and concatenates them into an initial summary of desired length. The second, slightly less naive, approach picks a random sentence in the source text and extracts it and the following couple of sentences to use as the initial summary for that text. After this the algorithm proceeds as before, transforming the initial summary until no better summary is found.

As can be seen in Table 3, the results from both approaches are strikingly similar. Since they are also quite a lot worse than the original approach, this gives further support to the notion that the leading sentences of a document constitutes a stable starting point.

5 Conclusions

We have evaluated a new weighting scheme for the HolSum framework. Using the burstiness of a word instead of the log(*idf*) part of the standard $tf \cdot \log(idf)$ weighting gave results very similar to the original version. Further studies showed that the main contribution comes from the term frequency, while both the burstiness and the inverse document frequency add small improvements.

The standard HolSum method performs a greedy search starting with the leading sentences of the text. We examined if beginning the search with other summary candidates would perhaps lead to different local maxima. Indeed, this is the case, but starting with the leading sentences was much better than the considered alternatives. It is of course also possible to do several greedy searches starting from different summary candidates and then take the best result, but since the leading sentences perform much better than the other alternatives, this might not make much difference.

We also evaluated other search strategies than the original simple greedy search. Using simulated annealing to see if better results can be achieved when the risk of getting stuck early on in a local maxima is lowered was tested. There was no detectable improvement from using these more advanced search strategies.

All in all, using the leading sentences as a starting point and then finding better summaries using a simple greedy search seems to work quite well. If inverse document frequencies are not available, using the term burstiness instead (which can easily be calculated from the text itself) gives almost the same results.

While the HolSum framework does not perform quite as well as the most advanced summarization systems available for English, it has some merits. It is very easy to implement and requires only very basic resources. Only word and sentence tokenization, stemming and stopword removal, and access to large amounts of unannotated text was used. Thus, the system can be used on many other languages, as long as raw text and some form of tokenization (not necessarily into words) is available. If more advanced resources such as stemming are available these can also be added with almost no extra effort.

HolSum also has the intuitively appealing property of trying to optimize semantic similarity between the generated summary and the text being summarized, though this is not always what you want from a summary.

References

Ronald Brandow, Karl Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Inf. Process. Manage.*, 31(5):675–685.

Lou Burnard. 1995. The Users Reference Guide for the British National Corpus.

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

H. P. Edmundson. 1969. New methods in automatic abstracting. *Journal of the Association for Computing Machinery*, 16(2):264–285.

Zelig S Harris. 1968. *Mathematical Structures of Language*. New York: Wiley.

Martin Hassel and Jonas Sjöbergh. 2005. A reflection of the whole picture is not always what you want, but that is what we give you. In *"Crossing Barriers in Text Summarization Research" workshop at RANLP'05*, Borovets, Bulgaria.

Martin Hassel and Jonas Sjöbergh. 2006. Towards holistic summarization: Selecting summaries, not sentences. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Genoa, Italy.

Maher Jaoua, Fatma Jaoua Kallel, and Abdelmajid Ben Hamadou. 2003. Une méthode de condensation automatique des documents multiples: cas des dépêches de presse. In *proceedings of CIDE6*, Caen, France.

Fatma Kallel Jaoua, Maher Jaoua, Lamia Belguith Hadrich, and Abdelmajid Ben Hamadou. 2004. Summarization at LARIS laboratory. In *Proceedings of the Fourth Document Understanding Conference (DUC'04)*, Boston, Massachusetts, USA.

Scott Kirkpatrick, C. Daniel Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680.

Peter J. M. Laarhoven and Emile H. L. Aarts, editors. 1987. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284.

Chin-Yew Lin and Eduard Hovy. 2003. The potential and limitations of automatic sentence extraction for summarization. In Dragomir Radev and Simone Teufel, editors, *HLT-NAACL 2003 Workshop: Text Summarization (DUC03)*, Edmonton, Alberta, Canada, May 31 - June 1. Association for Computational Linguistics.

Chin-Yew Lin. 2003. ROUGE: Recall-oriented understudy for gisting evaluation. http://www.isi.edu/~cyl/ROUGE/.

M. Ortuño, P. Carpena, P. Bernaola-Galvan, E. Munoz, and A. Somoza. 2002. Keyword detection in natural languages and DNA. *Europhysics Letters*, 57:759–764.

Paul Over and James Yen. 2004. An introduction to duc 2004 intrinsic evaluation of generic new text summarization systems. http://www-nlpir.nist.gov/projects/duc/pubs/2004slides/duc2004.intro.pdf.

Magnus Sahlgren. 2005. An Introduction to Random Indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005)*, Copenhagen, Denmark, August 16.

Magnus Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Department of Linguistics, Stockholm University.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA.