

Comparing Performance of Text Summarization Methods on Polish News Articles

Adam Dudczak², Jerzy Stefanowski¹, and Dawid Weiss¹

¹ Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2,
60-965 Poznań, Poland

² Poznan Supercomputing and Networking Center, ul. Noskowskiego 10, 61-704
Poznań, Poland

Abstract

This paper presents the goals, results and conclusions from an experiment where several shallow text summarization methods have been applied to news articles written in Polish. Specifically, we focused on various techniques of salient sentence selection as these algorithms are most popular in the English-spoken world and are highly efficient in practice. The quality of automatically generated summaries was evaluated by comparing them against a reference set of man-made summaries. This reference set of summaries is a valuable resource on its own as it comes from an unrestricted survey where user groups from different backgrounds had been asked to select “most appropriate” sentences to form a summary.

Keywords: text summarization, lexical chains, weighting techniques

1 Introduction

In the last decade one can observe a growing interest in processing electronic documents and mining their content. It particularly relates to Web resources, where users are flooded by huge numbers of documents and are unable to find necessary information without the help of machine methods. Notice that besides sending queries to a search engine, the user could be interested in exploring the topics (content) of a given collection of documents. Both these activities require new solutions for presenting a more insightful view of the content of text documents.

In our previous work on descriptive document clustering we stressed the role of meaningful labels (short phrases) for describing document groups (see Weiss (2006)). A similar idea could be applied to enhance the presentation of individual documents—search results could include a short summary of the document in addition to, or as a replacement for the query-contextual snippet of text presented by most search engines nowadays. Such a technique would be even more useful in the absence of a specific query (when browsing a large collection of documents), as it is the case with BEATCA¹ system for visualizing large collections of documents, for example.

¹<http://www.ipipan.eu/~klopotek/BEATCA/>

Automatic text summarization has been studied for over 50 years now (Zechner (1996)). The conclusion so far seems to be that literary-quality summaries, based on deep understanding of the original text, are very difficult to produce using machine methods. Therefore a number of simpler alternatives have been proposed, most of them relying on *selecting whole sentences* from the original text and presenting it as a summary of the entire document. Although some logical relationships between sentences may be violated or dropped during this process, experimental results with structured texts show that summaries generated in this way sufficiently represent the original content Kupiec *et al.* (1995).

Methods based on sentence selection have been widely adopted for English, but their application to Polish is surprisingly rare (see Ciura *et al.* (2004)). The inflection and syntax make the task more complicated than in English, but it would be interesting to know how the existing methods perform on Polish texts. This constitutes the paper's main goal. Three specific methods are considered: (1) a heuristic based on the position of sentences in paragraphs; (2) well-known term weighting schemes *tf-idf* and *okapi bm25*; (3) lexical chains.

To evaluate the generated summaries we compared them against baseline summaries manually created by humans using the same logic present in automatic summarization algorithms (salient sentence selection). Such a reference evaluation set had not been available for Polish, the second goal of this work was to collect it and make it freely available for future research on the subject.

2 Text Summarization by Sentence Selection

There are many ways of doing text summarization and previous research in this field gives a great deal of comparative material Alonso *et al.* (2003); Zechner (1996); Spärck Jones (1998). Authors in Alonso *et al.* (2003) point out the two most significant factors affecting classification of summarization methods. First, it is necessary to consider the level of complexity of natural language processing used by summarization methods. Certain approaches are based on analysis of statistical features of texts (eg., word frequency), others utilize information of connections between various building blocks of a document (eg., word co-occurrence), finally the most advanced algorithms build the structural model of discourse (eg., rhetorical structure theory Mann and Thompson (1987)). A bit different approach to classify summarization algorithms is derived from the type of information used to create the summary. In this taxonomy authors distinguish three types of algorithms: lexical-based (word frequencies, cue words and phrases), structural-based (position of sentences in paragraphs) and those based on information derived from deeper text understanding. These classes are not fully disjoint and state of art summarization systems combine many different methods to provide comprehensive summaries.

Most summarization algorithms based on sentence selection operate according to roughly the same core routine. First, the text is pre-processed, sentence and paragraph boundaries are marked. Then an "importance" function is calculated for each sentence. Finally, sentences are ranked and highest-scoring sentences are picked to become the summary of the document.

The “importance” function calculated for each sentence is crucial for proper selection of salient sentences. This function is typically calculated as an aggregation of certain statistical properties of the text, for instance position of the sentence with regard to its paragraph and the whole document, presence or absence of certain key phrases (“summarizing”, “concluding”), sentence length, number of referred proper names, specificity of words appearing in the sentence, etc. We will describe three approaches to ranking sentences later in this section. Note, however, that all these features are fairly shallow, require no understanding of the text being processed and are usually extremely fast to compute. The trade-off for speed here is quality. The coherence of the output summary can suffer because of several reasons. First, since there is no understanding of the meaning of the text, the sentence ranking function can be easily tricked when synonyms or different wording is used throughout the document. Second, even if sentences forming the summary make sense and are grammatically intact on their own, their artificial composition next to each other may be incorrect and misleading. Our intuition was to keep these problems in mind, allow users to select sentences *they* think should appear in a summary and see how automated methods come out in the challenge. We chose three algorithms, briefly characterized below.

Simple sentence position

News articles, such as the ones used in our experiments, should be written according to a very rigid layout. An article begins with a short summary, introducing major content highlights. The following paragraphs expand upon these topics and somewhere at the end of the paper, its conclusions are concisely laid out to the reader. Knowledge of this pattern can be used to drive a simple summarization method. Assuming the desired length of the summary is n sentences, the algorithm works as shown in the pseudo code below.

Require: D – input document, $n \geq 1$ – number of sentences in the summary.
 $P \leftarrow$ an array of paragraphs from D , each consisting of an array of sentences.
 $S \leftarrow \emptyset$ {Sentences selected for the summary}

```

loop
  for all  $p \in P$  do
     $s \leftarrow$  first sentence from  $p$ 
    add  $s$  to  $S$  and remove it from  $p$ 
    if  $p$  is empty then
      remove  $p$  from  $P$ 
    end if
    if  $|S| = n$  or  $P$  is empty then
      return  $S$ 
    end if
  end for
end loop

```

Sentence importance using word weighting

Given a certain document we can distinguish a set of words that are characteristic to its topic, that distinguish this particular document from most other documents in the collection. These words are usually called *keywords* and there is a vast amount of research done to find them in the field of information retrieval. Typ-

ically, each word is assigned a numeric weight. Sentences are then ranked using an aggregate function over the sentence's word weights. Out of many possible word weighting schemes we chose the ones used commonly in classic information retrieval: *tf-idf* and *bm25 okapi*.

In *tf-idf* (term frequency, inverse document frequency), every word in a document is assigned a weight equal to:

$$\text{tf-idf}(t_j) = \frac{D(t_j)}{|D|} \times \log \left(\frac{C}{C(t_j)} \right) \quad (1)$$

where C is the number of documents in a collection, $C(t_j)$ is the number of documents containing term t_j , $|D|$ is the total of all words in the document and $D(t_j)$ denotes how many times t_j occurred in document D . Intuitively, this weighting scheme attempts to boost words that occur frequently in a document and are specific to it (do not show up in a number of other documents). The problem with *tf-idf* is that words in documents of different lengths are scored differently (longer documents are preferred). The *bm25 okapi* weighting scheme introduces a special component to Equation 1 that makes word weighting for documents of different lengths fair. Details can be found in Sparck Jones *et al.* (2000), we omit them here. In our experiments the values of collection statistics above were calculated from all of Polish Wikipedia.

Once the weighting scheme for individual words is known, the summarization algorithm follows as in pseudo code below Brandow *et al.* (1995):

Require: D – input document, $n \geq 1$ – number of sentences in the summary.

```

 $S \leftarrow$  all sentences in  $D$ 
assign weights to all (or some, i.e., only nouns) of document words,
for all sentences  $s \in S$  do
    calculate aggregate score for words in  $s$ 
end for
sort sentences in  $S$ , better score first
return  $n$  top sentences, skipping highly word-overlapping sentences.
```

Sentence importance with lexical chains

The methods presented so far focused on positional and occurrence-based statistics of the text, completely ignoring the aspect of interactions *between* words. Certain language phenomena, like synonyms, hipernyms or hyponyms, could easily confuse these methods. Lexical chains are an attempt to capture the actual semantic meaning of words and their groups appearing in the text. The method was first introduced and used for text summarization in Barzilay and Elhadad (1997), we provide a quick insight into how it works below.

A well edited text will be perceived by the user as a coherent flow of thoughts. To achieve this goal authors carefully select the language structures and words to be used, often borrowing words from semantically related fields or taking synonyms of the same word to avoid repetitions. We can speak of two types of *lexical cohesion*:

- *reiteration*, understood as a relationship between a word and its synonyms and hipernyms,

- *co-occurrence*, when two or more words frequently appear in proximity to each other in the text, collocations are a strong example of co-occurrence.

When two or more words are involved in one of the above, we call such group a *lexical chain*. In Barzilay and Elhadad (1997) authors emphasize that, in general, identification of lexical chains is not trivial; in our experiment we focused on detecting reiterations.

An algorithm for extracting lexical chains from the text is a heuristic that iteratively groups related sets of words into the most likely *interpretations*, containing unambiguous resolution of each word's sense. Relatedness and senses of words are determined using a thesaurus (we used a publicly available Polish thesaurus from <http://synonimy.ux.pl>). A crude version of the algorithm's pseudo code is presented below.

```

Require:  $D$  – input document,  $n \geq 1$  – number of sentences in the summary.
 $LC(D) \leftarrow \emptyset$  {Output set of lexical chains}
for all paragraphs  $P \in D$  do
  interpretations  $I(P) \leftarrow \emptyset$ 
  for all sentences  $S \in P$  do
    for all words  $w \in S$  do
      temporary list of interpretations  $I_t \leftarrow \emptyset$ 
      for all word senses  $s(w)$  do
        if  $I(P)$  contains interpretation  $I$  which contains word  $w$  in sense  $s(w)$  then
          add  $w$ 's position to  $I(P)[I][s(W)]$ 
        else if  $I(P)$  does not contain word  $w$  in any sense then
          for all interpretation  $I \in I(P)$  do
            put  $I \cup s(w)$  to  $I_t$ 
          end for
        end if
      end for
    end for
     $I(P) \leftarrow I(P) \cup I_t$ ;
    prune weakest interpretations from  $I(P)$ 
  end for
   $LC(P) \leftarrow$  lexical chains from  $I(P)$ 
   $LC(D) \leftarrow LC(D) \cup LC(P)$ 
  prune weakest chains from  $LC(D)$ 
end for

```

The above procedure basically builds all potential combinations of word senses locally and greedily picks the best candidates at sentence, paragraph and document level. The details are not of crucial importance to this paper, so we omit them here (see Dudczak (2007); Dudczak *et al.* (2008) for full description).

After extraction, lexical chains are assigned a score proportional to their length and document-frequency of words the chain contains. Since the information which word belonged to which chain is preserved at the time of chain extraction, the score of each chain can be applied back to sentences. Each sentence therefore receives a final score proportional to how many words from lexical chains it contains and how strong these chains are. The summarization procedure then follows in the usual way, sorting sentences and selecting those with the highest score.

TABLE 1: Characteristics of articles and summaries used in the experiment.

article ID	1	2	3	4	5	6	7	8	9	10
no. of words	419	246	236	397	632	467	732	276	859	1061
no. of sentences	22	16	16	20	44	29	49	14	54	56
sentences for summary	5	4	4	5	10	7	11	3	11	12
collected summaries	27	28	28	28	29	29	27	29	30	30

3 Creating Corpus of Summaries for Polish News Articles

The evaluation phase required a “reference” set of summaries. We assumed that it should be prepared by humans and that humans should follow the same logic of “important” sentence selection as the algorithms under evaluation (comparing literary abstracts would be much more complex, see Kupiec *et al.* (1995)).

We selected 10 diverse press articles originating from main Polish journals “Gazeta Wyborcza” and “Rzeczpospolita”. Some papers are typical short “news” (e.g., about Ségolène Royal during French presidential election) while others are longer, with more expanded structure, vocabulary and topics (e.g., history of one pioneer of cinema). All the articles were divided into paragraphs. For each text we established the length of the desired summary to 20% of the original text (literature recommendations), survey participants had to select the required number of sentences. The basic characteristics of the articles and summaries are given in Table 1.

To facilitate the process of collecting summaries, we published an on-line service called “Lakon”.² Exactly 60 volunteers took part in the experiment between May 15th and June 14th of 2007, submitting a total of 285 summaries (between 27 and 30 for a single article—see details in Table 1). The submitters were mainly students and graduates of Poznań universities, including potential “experts” in the field—21 people from philology and journalism departments and librarians.

First we analyzed summaries from different authors. For each press article we took a pair of different summaries and calculated the number of common sentences. Then we averaged this number and normalized it by the summary length. The averaged shared parts were, in the order of article IDs from table 1: 39%, 55%, 46%, 56%, 41%, 54%, 37%, 55%, 45% and 49%. These results are good compared to previous research on English scientific papers. We then checked the most often chosen sentences and their positions with respect to paragraphs. It turned out that participants usually chose the sentences located at the beginning of each paragraph. This was especially true for articles numbered 1–4 (news stories), for longer articles (IDs 7, 9, 10) the choices are slightly more distributed. This is consistent with results reported in Perez-Breva and Yoshimi (2002) which showed that first sentences in texts of English press articles are also often chosen for a summary.

²We omit the details here, the Reader is referred to <http://www.cs.put.poznan.pl/dweiss/lakon>.

TABLE 2: Number of sentences picked for the two reference sets M and R .

article ID	1	2	3	4	5	6	7	8	9	10
average selection	7.11	8.62	8	8.24	7.84	8.12	6.91	8.7	7.86	6.79
$ M $	5	4	4	5	10	7	11	3	11	12
$ R $	6	6	7	5	14	9	19	4	18	17

TABLE 3: Coverage of evaluated summarization methods.

ID	bm25	tf-idf	LC	Pos	First	Rand
1	0.2	0.4	0.6	0.8	0.6	0.27
2	0.25	0.5	0.25	0.5	0.25	0.18
3	0.5	0.5	0.5	0.75	0.5	0.22
4	0.4	0.6	0.6	0.6	0.6	0.23
5	0.3	0.4	0.4	0.5	0.3	0.2
6	0.43	0.43	0.29	0.43	0.43	0.27
7	0.55	0.55	0.36	0.36	0	0.22
8	0.67	0.67	0.67	0.33	0.33	0.31
9	0.36	0.45	0.27	0.64	0.45	0.21
10	0.5	0.5	0.5	0.42	0.33	0.19

4 Evaluation of Automatic Summarization Methods

During the user survey we collected a number of summaries for each article. From this data we then built two different reference sets of summaries against which automatic methods were compared.

The first reference set, called a “model summary” and denoted by M further on, was created in the following way. For a given article we calculated the number of times each sentence was selected by users, choosing n most frequently chosen sentences, where n was the required summary length (see Table 1). In case of draws between sentences n and $n + 1$, the one occurring earlier in the text was winning. We also manually validated each final summary to ensure no two sentences carried nearly identical information.

The second reference set, called a “relevant sentence set” and denoted by R further on, was created in a different way to address two issues. First, the overlap between summaries generated by users was high, but not *very* high. Second, some summaries were really short. For each article we again calculated the number of selections of each sentence and an additional *average* number of selections over sentences chosen at least once. The summaries in the relevant sentence set contain sentences that had been selected more frequently than the average. Table 2 provides more insight into final summaries in these different evaluation sets.

The quality of an automatically generated summary S was evaluated by comparing it against reference sets M and R , counting three measures (the notation $|X|$ indicates the number of sentences):

$$coverage = \frac{|S \cap M|}{|S|}, \quad recall = \frac{|S \cap R|}{|R|}, \quad precision = \frac{|S \cap R|}{|S|}.$$

All these measures were inspired by typical evaluation functions in information retrieval. Coverage is defined assuming one perfect ground truth summary

TABLE 4: Recall of evaluated summarization methods.

ID	bm25	tf-idf	LC	Pos	First	Rand
1	0.17	0.33	0.5	0.67	0.5	0.25
2	0.33	0.33	0.33	0.33	0.17	0.21
3	0.43	0.43	0.43	0.43	0.43	0.22
4	0.4	0.6	0.6	0.6	0.6	0.23
5	0.29	0.36	0.36	0.57	0.29	0.2
6	0.56	0.44	0.22	0.44	0.44	0.25
7	0.37	0.32	0.26	0.32	0.32	0.23
8	0.75	0.75	0.5	0.5	0.5	0.28
9	0.28	0.39	0.33	0.39	0.39	0.21
10	0.47	0.41	0.41	0.29	0.24	0.19

TABLE 5: Precision of evaluated summarization methods.

ID	bm25	tf-idf	LC	Pos	First	Rand
1	0.2	0.4	0.6	0.8	0.6	0.3
2	0.5	0.5	0.5	0.5	0.25	0.31
3	0.75	0.75	0.75	0.75	0.75	0.38
4	0.4	0.6	0.6	0.6	0.6	0.23
5	0.4	0.5	0.5	0.8	0.4	0.28
6	0.71	0.57	0.29	0.57	0.57	0.32
7	0.64	0.55	0.45	0.55	0.55	0.39
8	1	1	0.67	0.67	0.67	0.37
9	0.45	0.64	0.55	0.64	0.64	0.35
10	0.67	0.58	0.58	0.42	0.33	0.26

M , precision reflects the existence of an extended set of good summaries, recall determines how many relevant sentences were automatically discovered.

In our experiment summaries were generated using three different methods mentioned in Section 2.³ We will denote the paragraph positional method as *Pos*, word weighting methods as *tf-idf* and *okapi bm25*, finally lexical chains as *LC*. The comparison was extended with two simple baseline heuristics: plain random choice of n sentences (*Rand*) and selection of the first n sentences from the original text (*First*). The results of are presented in Table 3, 4 and 5, the discussion of results appears in the next section.

5 Discussion and Final Remarks

Let us summarize the results of our experiments. It is a bit surprising that the simplest method based on sentence position (*Pos*) worked best on the average, in particular on shorter articles (ID 1, 3). Looking at previous research on English, these results seem to be justified—for news articles the positional methods worked best as reported in Baxendale (1958). The performance of the positional method was worse for longer articles (ID 5 and above) and articles with non-standard structure (ID 8), the keyword weighting method worked better there. Interestingly,

³Full text of all generated summaries is available in extra materials attached to Dudczak (2007).

feedback gained from users of the survey also indicates the difficulty of selecting summary sentences increased with the length of the summarized article.

Methods based on word weighting worked better for texts with many proper names (person or organization names, dates, trademarks). These methods were more efficient for longer texts, where the paragraphs were longer and more sophisticated (article 10).

Utilizing lexical chains did not improve summarization quality. We believe the reason for this is in the thesaurus that did not contain many terms or proper names (weighting schemes may better accommodate to dynamically changing vocabulary). The lexical chain extraction routine also works better for longer texts, where reiteration of terms is evident. In case of news stories this might have been a problem. Looking at values of recall, one can notice that the values there were less discriminating than precision or coverage.

Considering computational costs the lexical chains were definitely the most demanding approach (we discuss technical aspects in Dudczak *et al.* (2008)), requiring approximately 10 times more time to complete than word weighting methods.

We are aware that the number of testing texts in our experiments could be higher but even this number required large efforts with encouraging human participant to cooperate. The set of used articles and all summaries generated by experiment participants are available under <http://www.cs.put.poznan.pl/dweiss/lakon>. We hope that the results are enough to support our general observation that salient sentence selection methods, being quite popular for processing English papers, are also effective to summarize Polish language news texts. This result is a continuation of previous work on more complex summarizers, presented by prof. Nina Suszczańska research group at the Silesian University of Technology Ciura *et al.* (2004). We are not aware of any other research attempts to provide efficient summarizing algorithms for Polish.

Further research on the topic should include: extending the experiments to include longer texts, using multi-criteria evaluation functions, and considering additional (yet shallow) language analysis for detection of anaphoric relationships.

References

- Laura ALONSO, Irene CASTELLÓN, Salvador CLIMENT, Maria FUENTES, Lluís PADRÓ, and Horacio RODRÍGUEZ (2003), Approaches to Text Summarization: Questions and Answers, *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 22:79–102.
- Regina BARZILAY and Michael ELHADAD (1997), Using Lexical Chains for Text Summarization, *Intelligent Scalable Text Summarization Workshop (ISTS'97)*, pp. 10–17.
- P. B. BAXENDALE (1958), Machine-made index for technical literature – an experiment, *IBM Journal of Research and Development*, 2(4):354–361.
- Ronald BRANDOW, Karl MITZE, and Lisa F. RAU (1995), Automatic condensation of electronic publications by sentence selection, *Inf. Process. Manage.*, 31(5):675–685, ISSN 0306-4573.
- Marcin CIURA, Damian GRUND, Sławomir KULIKÓW, and Nina SUSZCZANSKA (2004), A System to Adapt Techniques of Text Summarizing to Polish, in *International Conference on Computational Intelligence, Istanbul, Turkey*, pp. 117–120.

- Adam DUDCZAK (2007), Zastosowanie wybranych metod eksploracji danych do tworzenia streszczeń tekstów prasowych dla języka polskiego, Master thesis. Poznań University of Technology.
- Adam DUDCZAK, Jerzy STEFANOWSKI, and Dawid WEISS (2008), Automatyczna selekcja zdań dla tekstów prasowych w języku polskim, Technical Report RA-03/08, Institute of Computing Science, Poznan University of Technology, Poland.
- J. KUPIEC, J. PEDERSON, and F. CHEN (1995), A trainable document summarizer, in *Proceedings of the 18th ACM-SIGIR Conference*, pp. 68–73.
- W. C. MANN and S. A. THOMPSON (1987), Rhetorical Structure Theory: A Theory of Text Organization, Technical Report ISI/RS-87-190, Information Sciences Institute.
- Luis PEREZ-BREVA and Osamu YOSHIMI (2002), Model Selection in Summary Evaluation, Technical Report AI Memo 2002–023, Massachusetts Institute of Technology, AI laboratory.
- Karen SPÄRCK JONES (1998), Automatic Summarising: Factors and Directions, in I. MANI and M. MAYBURY, editors, *Advances in Automatic Text Summarisation*, MIT Press, Cambridge MA.
- Karen SPARCK JONES, Steve WALKER, and Stephen E. ROBERTSON (2000), A probabilistic model of information retrieval: development and comparative experiments, Part 2, *Information Processing and Management*, 36(6):809–840.
- Dawid WEISS (2006), *Descriptive Clustering as a Method for Exploring Text Collections*, Ph.D. thesis, Poznan University of Technology, Poznań, Poland.
- K. ZECHNER (1996), A literature survey on information extraction and text summarization, Available on-line: <http://www.csi.uottawa.ca/tanka/ArtDB/infoextr.ps.gz>.