Dynamic Segmentation for Efficient Retrieval of Podcasts

The Repping Algorithm

Stephan Repp
Independent
Trier, RLP, Germany
repping@repp.app

Ernst Georg Haffner
Hochschule Trier
Trier, RLP, Germany
e.haffner@hochschule-trier.de

Abstract

In the following article, we present a method that makes it possible to find specific segments in a podcast from a large collection using a query (keywords or question). What differentiates our method is that there is no segmentation process at the beginning, but rather the segmentation is done dynamically according to the query entered. The core of our method is that for each term a position-based index is spanned over each individual document. These indices are laid over the individual documents like small threads of information. This multitude of threads maps the inner semantic structure of each individual document in the collection. The corresponding response segments are then individually determined according to the query at runtime using this index. Our initial tests have shown that this method significantly outperforms all current podcast-retrieval methods.

Keywords

Podcast Retrieval, Segment Retrieval, Dynamic Segmentation, Speech Browsing, Video Browsing, Query-based Search, Semantic Indexing, Multimedia Information Retrieval, Question Answering, Transcript Search, Fine-grained Content Access, Floating Index

The definitive Version of Record was published in *Proceedings of the 2024 International Conference on Multimedia Retrieval (ICMR '24)*,
DOI: 10.1145/3652583.3658047

1 Introduction

Standard search engines deliver several results for one search: Generally they provide complete pages or entire documents, highlighting relevant passages of text. However, if only specific passages within the selected document are of interest, the only option is often the old- fashioned, classic keyword search via a browser. Thus, although traditional search engines offer a convenient way to search for documents, they do not allow one to easily search or browse specific areas within the documents. In the case of multimedia files (podcasts, videos), the situation is compounded by the fact that this content is often not yet indexed by search engines. Here, the search engine will not find any results unless there is meta information about the media files entered by hand. If a video or podcast is to be indexed by the searcher, then he or she must work through the entirety of the content to find the relevant segment if no metadata is available. This takes quite a long time for feature films or extensive podcasts.

In addition, it still does not provide a comparison of this segment found with another segment from another multimedia document. The crucial problem of which passage in the whole corpus is the most relevant segment for the corresponding query has not been solved. Our approach provides an individual list of exact locations from a document collection for a query. These locations are ranked by relevance and the user receives a list of possible answers to his or her query. A document can also contain several reference locations for one search.

Classical approaches first run through a text segmentation and then rank or index the resulting segments accordingly. Errors can occur during the segmentation process, which are amplified as the process continues or make the result unusable.

Our method is different in that no segmentation process takes place in advance, but rather the segmentation is carried out dynamically with the search query. It is a massive process that does not consist of two separate processes. The basis for this is that an index is created for each term via the documents. For each term, this index contains a time (for podcast/videos) or a character position (for texts), when this term occurs frequently. When searching for word combinations, the temporal intersection of the ranges is used. This results in a large number of possible combinations for a search query. We have succeeded in optimizing the method, so that it calculates both the ranking and the intersection very quickly. It is also possible to add new documents to the index without completely recalculating the entire index.

We call the method "repping." The term "repping" is, on the one hand, based on the word "repp." Repp – also known as rep, rip, or reps – is a fabric woven in fine cords or ribs across the width of a piece. These fine cords hold the fabric together and the fabric appears whole. Our method is also based on many fine strips that together form a whole: the contents of a document. The second meaning for the term "repping" is based on the word "reppin," which means to represent something. The index generated represents the whole dimension of a document.

2 Related Work

The research status of our field is well described in the two summaries of the TREC Podcasts Track [11, 38]. For these conferences, relevant podcasts and the corresponding transcripts with the time stamps were provided. These could then be used to compare the different methods. Unfortunately, the podcast track no longer takes place, but the data provided by the conference is used in our experiments. These conferences showed that methods with pretrained transformers such as BERT performed best [2, 3]. In these methods, segments of the podcast transcripts are first created. The resulting text segments are then used as input for the transformer [5, 16]. This approach shows surprisingly good results. A disadvantage of this, however, is the large amount of computing power required [36].

Information retrieval is a rather old science that is constantly evolving. A very big milestone was what is known as the exact term matching method. The Okapi BM25 method thus represents a breakthrough in information retrieval. As a baseline, 120 second-long text segments are retrieved with the Okapi BM25 method. This procedure represents the baseline, which we aim to exceed [9, 38]. However, there are some disadvantages of the exact term matching method. As the term implies, only terms that exactly match the query can be found in the corpus. A solution involves extending either the search space or the search query with different methods available. Another improvement is to include the information from the provided meta information (headings in texts, summary, etc.) in the weighting of the

retrieved terms [40].
In these procedures of podcast tracks from the TREC conference, the documents are first divided into individual segments and then indexed using various indexing methods. This first step of dividing the documents into meaningful segments is a big challenge. [19] present an overview of the current segmentation methods of texts. Segmentation of text is about extracting coherent blocks of text [18]. The segment is called a "segment boundary" [35] or "passage" [15]. Two other studies refer to a segment as a "subtopic" [37] and "region of interest" [17].
There are many reasons why splitting up a document can be useful for text analysis. One of the main reasons is because it makes them smaller and more coherent than whole documents [15]. Another reason is that each segment is used as a unit of analysis and access [15]. Text segmentation has been used for processing text in topic identification [1, 4], language detection [20], and information retrieval [8], among other things.
What all methods have in common is that the boundaries are set statically in the document. Only then does the actual indexing process of these segments usually take place. The process segmentation is very error-prone. What exactly a segment is, is often not clearly defined and is also task dependent. It is desirable to have a process that can create the segments very dynamically depending on the query.
Linguistic research has shown that word repetition in a text is an indication of thematic context. A change in vocabulary and its specific distribution within a document usually indicates a topic change [6, 7, 13, 14, 31, 39].
In our opinion, current approaches do not take these features into account to a sufficient degree. Therefore, we assume that certain areas have similar vocabularies and the size of the segments found can vary depending on the query. The goal of indexing is to capture the semantic structure of documents in depth in a structured manner. These points are the starting points of our method, which will be described in the following:

3 Algorithm

In order for podcasts to be indexed, a transcript is generated at the beginning with the help of a speech recognizer. This transcript with the corresponding time stamps of the individual words represents the basis for the indexing. Texts can be indexed in the same manner and the position of the words represents the correspondence of the time marks. First, a type of clustering is carried out, which captures cohesive and similar areas of a transcript/texts (called "`text`" in the following) and stores them as an index. The clustering of the same words at certain intervals are taken into account. Each entry in the index provides information about certain locations in the text. An index entry is a range from the first occurrence to the last occurrence of a word within a certain text range. The spacing of the words must not exceed a certain interval. The procedure consists of the following steps: Preprocessing, Clustering, Calculation of the index and Searching the corresponding segments for a query. Both Preprocessing and Clustering are described in detail in the following publications [21, 22, 27–29].

3.1 Preprocessing

After standard preprocessing – reduction of the words to their basic form and filtering of the stopwords – these terms are stored in a list L. The list L contains all terms that occur in the texts without the stopwords S. Stopwords could also be taken into account. Since they occur quite often, they are not marked later in the index as highly relevant. Consequently, they then have little effect on the result and we have filtered them out. There are n different terms. t_i is a selected term and V is the vocabulary of the texts.

$$L = \{t_i \in V | 1 \leq i \leq n\} \setminus S \tag{1}$$

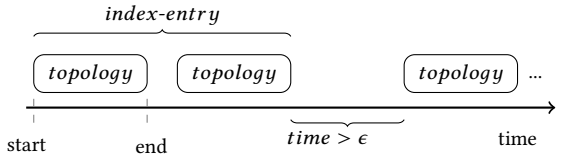

Figure 1: Example of Index-Entries

3.2 Clustering of the thematically similar areas

In this step, for each term t_i, the ranges are generated in which the term occurs in clusters, see figure 1. To do this, a threshold ϵ is set that defines the distance between the terms. If the distance between the occurrence of adjacent terms t_i is greater than ϵ, a new index entry is created. The distance ϵ refers only to equal terms t_i that are close to each other. This clustering is used to locate the areas where the speaker talks about a term. For each index entry r_k, the term number $f(t_i, k)$ of a term t_i, the start time and the end time are recorded, see figure 2. Here k denotes one index entry uniquely, m is the number of all index entries.

$$1 \leq k \leq m \tag{2}$$

The table 1 and the figure 2 show an example of an index for a podcast. The first index "topology" contains the term "topology" in a podcast 12 times; the index entry starts at the start time 100s and ends at the end time 3000s. In addition to this index entry, the same text contains another "topology" index entry with a start time of 3200s and an end time of 5000s. In this index entry, the term "topology" is mentioned 7 times. This example also shows that there is an overlap between the individual index entries of different terms. Thus, the figure 2 shows for example that the term "topology" and the term "star" overlap.

3.3 Calculation of the relevance of an index entry

In our first publications, we used the index entries created in this way for the automatic generation of learning objects. A semantic search engine was built on these learning objects. This provided the time position in the lecture video in response to a question [23, 24, 24, 29].
In our research [22] only the simple number of occurring terms (term number) was used for the weighting. For an given archive of 24 lecture videos in German language and 1860 minutes total duration, it could be shown on a lecture browser that this is sufficient for browsing the lecture videos.
We have developed this approach further and use a modified formula of the Okapi BM25 method. Each index entry is assigned a specific relevance, i.e. each entry has its individual $score$. Our formula is based on the Okapi BM25 function [10, 38].
The Okapi BM25 function computes the relevance of documents with respect to a query. Since our problem involves segments within documents, we first compute only the relevance of the segments. The subsequent query is then determined using this index. We have adapted the Okapi BM25 function for the relevance calculation of the individual segments as follows:
The `score` for an index entry r_k of the term t_i is calculated with:

$$\texttt{score}(r_k, t_i) = \texttt{IDF}(t_i) \cdot \left(1 + k_3 \cdot \frac{f(t_i, k) \cdot (k_1 + 1)}{f(t_i, k) + k_1 \cdot \left(1 - b + b \cdot \frac{avgdl}{|r_k|}\right)}\right) \tag{3}$$

$$\texttt{IDF}(t_i) = \ln\left(k_2 + \frac{N}{n(t_i)}\right) \tag{4}$$

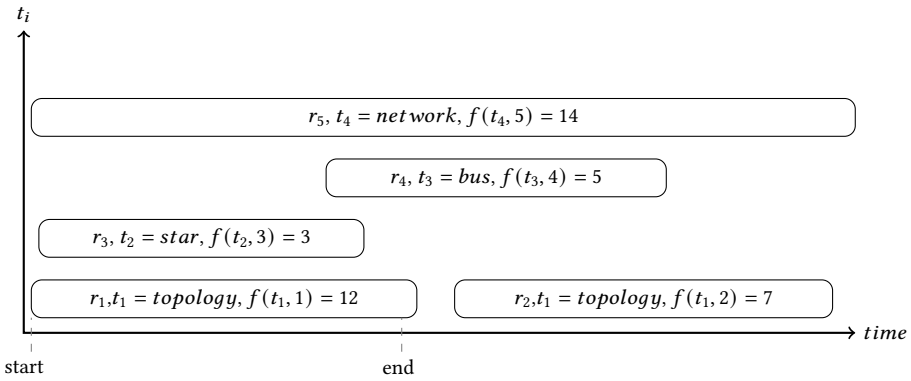

Figure 2: Example of the Index

$\texttt{IDF}(t_i)$ is the inverse document frequency. Here, N corresponds to the number of documents in the corpus, $n(t_i)$ is the number of documents in which the term t_i occurs. This first part of the formula reflects the frequency of occurrence of the term t_i in the documents N. The other part of the formula reflects the relevance of the individual index entries. It captures to what extent the index entries and their parameters contribute to the relevance (*score* value).

Where $f(t_i, k)$ is the number of occurrences of the term t_i between *start* and *end* of an index entry r_k, $|r_k|$ is the number of all terms in the corresponding range. This means **all** terms (not just t_i terms) occurring in the corresponding time interval from *start* to *end*; *avgdl* is the average number of all occurring terms in the corresponding range.

For the determination of the parameters *avgdl*, $k1$, $k2$, $k3$, b and ϵ we have performed initial tests and estimated the values. The question of the optimal parameters has not been explored conclusively yet. In particular, balancing the relevance of the document (first part of the formula $\texttt{IDF}(t_i)$) and the relevance of the individual areas (second part of the formula) has not yet been clarified in detail. Since the precise determination of these parameters is very complex and time-consuming, we have not yet been able to do this with our resources. It took several years of extensive research to obtain general parameter values for the the Okapi BM25 formula.[10, 32–34, 38] and many more publications. This lengthy and laborious process can only be worked out collaboratively in the research community.

In contrast to the original Okapi BM25 formula, we have exchanged $|r_k|$ in the fraction with *avgdl*. This means that a range with a higher number of terms is weighted higher. When the speaker talks about the topic for a longer time, this signals a higher relevance. The swap has a positive effect on the search result. Investigating this in more detail is also the subject of future extensive and time-consuming research.

Our research shows that our approach is promising and it would be worthwhile to continue developing this approach. The following values for the parameters are considered a first attempt to achieve a good result. There is still considerable potential here for increasing the performance of the method. We have set the values as follows: $avgdl = 10$, $k1 = 1$, $b = 0.75$. For the conversion into a performant and executable software we have chosen $k2 = 9$. This ensures that $\texttt{IDF}(q_i)$ is greater than or equal to 1. Similarly, we chose the parameter $k3$ so that the second part is likewise always greater than or equal to 1, $k3 = 1000$. This ensures that the ranking can be stored as an integer value. We determined a distance for ϵ of about 3 minutes [22].

We have now created an index entry we call *repp* for repetition. We call the resulting relation $R(k, docID, t_i, start, end, \texttt{score})$ the *repp-index*, one element is one *repp*. This contains all the entries that overlay each document like little semantic threads, capturing semantic structure. The table 1 contains an example of this relation.

Table 1: *Repp-index*: $R(k, \textbf{docID}, t_i, start, end, \textbf{score})$, $f(t_i, k)$

k	$docID$	t_i	$start$	end	$f(t_i, k)$	$\texttt{score}(r_k, t_i)$
1	3	topology	100	3000	12	64
2	3	topology	3200	5000	7	12
3	3	star	400	2000	3	50
4	3	bus	4000	5500	8	10
5	3	network	10	5600	3	7
...	...	...	...	...	...	...

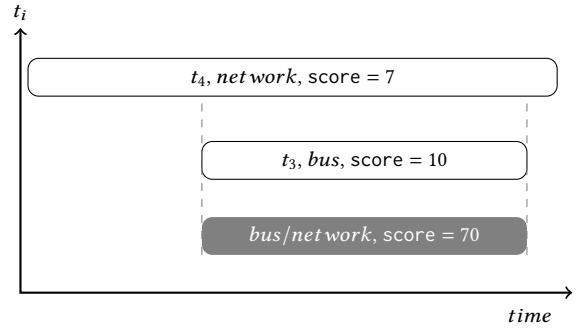

Figure 3: Example of a Search

3.4 Searching the corresponding segments

The next step is to search for the segments in the corpus. First, all entries for the corresponding search query Q are selected in the *repp-index* R. Now, only those documents in the *repp-index* which contain the entire search terms are included. Then the intersection is formed from the two (or more) *repps* found. That is, the intersection of the temporal overlap of the segments is created, see figure 3. This creates new segments. These new segments are given a new score measure that is calculated by multiplying the two individual score values. The sorted result, sorted according to the scoring value, represents the result.

In the following example, the two search terms "network" and "bus" are searched. The segment found (bus/network) is now the temporal intersection of the two *repps* found, see figure 3. If the result set has several such

segments, they are output sorted by score in descending order. In detail, the search is done as follows, see also the algorithm 1:
A search query Q contains the terms.

$$Q = \{q_j \in L | 1 \leq j \leq o\} \tag{5}$$

o is the maximum allowed number of search terms in a query, q_j is a query term. The first step is to find the documents in which the search terms all occur. To do this, the documents for a q_j are determined.

$$D_j = \{D_j \subseteq D | q_j \in D_j\} \tag{6}$$

D is the set of all documents, D_j is the subset of documents in which the query term q_j occurs. The result of the intersection over all sets of the documents found are the documents that contain all terms.

$$M = D_1 \cap D_2 \cap D_3 \cdots \cap D_o \tag{7}$$

M is the intersection of all documents in which all search terms q_j occur. The algorithm now determines the segments of the search query sorted by the score. For this purpose, the intersection of the $repps$ is determined for each document (from line 1), i.e. the overlapping areas of temporal occurrence (lines 8-19). The new $score$ value is determined by multiplication (line 9) and the new resulting segments are entered into the relation RS (RS.add, line 11, 13, 15, 17), depending on how the areas overlap. The sorted relation according to the $score$ (line 22) represents the result of the search.
It is also possible to perform the search with synonyms for a term. A term can be expressed by several other terms that mean the same thing. For this, the $repps$ found of the two synonyms (from the set of documents found, M) are first united with each other using a union set and then considered **a** new segment $repp$.
The presented algorithm was first implemented with an ordinary database. This shows a considerable runtime for the query. In addition, the runtime increases exponentially with the number of stored index entries.
We therefore developed an efficient data structure that maps the index. In addition, we created our own key-value database and combined this with a cache. So a $O(n)$ solution has been found that does not rely on a database. This enabled us to process the queries in ms. Additionally, it is possible to add a new document to the index without having to recalculate the index. Describing the technical realization is beyond the scope of this publication.
Our implementation is done on a $AMDRyzen^{TM}77700$ with 64 GB DDR5. The 100,000+ podcasts in the following experiments were indexed within 5 hours. Calculating the CO_2 emissions for these 5 hours on the corresponding processor is not exactly easy. The page Machine Learning Impact Calculator only offers a calculation for much more powerful processors [12].
We therefore assume that the indexing of the following experiments required less than 0.1 $kgCO_2eq$ and that this CO_2 consumption is very efficient for solving the task. The answer to a query is provided in near real time (much smaller than $100ms$).

4 Results

The guiding data for the evaluation are the data from both the 2020 and 2021 TREC Podcasts Tracks, see page https://trecpodcasts.github.io/.
For the evaluation, the two conferences provide over 100,000 podcast episodes with the appropriate meta (audio, automatic transcripts, etc.). A number of questions were provided for the evaluation as part of the conference. The task was to find relevant podcasts from the provided archive and, in addition, relevant time periods within these podcasts.
Assessments were made on the PEGFB graded scale (Perfect, Excellent, Good, Fair, Bad) as approximately follows: Excellent (3), Good (2), Fair (1) and Bad (0).
Generally, topics were formulated in three types: topical, re-finding, and known-item in the Podcast Task 2020. In the track of 2021, only topical and known-item are distinguished. Here the question types of re-finding and known-item were combined into known-item.
For this question type (known-item), there are only the categories Perfect (4): the segment is the intended item and Bad (0): the segment is not intended item.
For our evaluation, we only distinguish between topic and known-item (re-finding and known-item) for the 2020 data. The metrics for evaluation is $nDCG$ on the first thirty items found. Normalization is based on the ideal ranking of all relevant segments. In addition, the precision of the first ten segments found is calculated.

Table 2: Comparing to TREC 2020

	DCG_{30}	P_{10}
repping	**0.50**	**0.73**
UMD_IR_run3	0.52	0.60
UMD_IR_run4	0.50	0.58
UMD_IR_run1	0.49	0.56
BERT-DESC-S	0.47	0.57
...	...	...
Baseline BM25	**0.40**	**0.49**

Table 3: Comparing to TREC 2021

	DCG_{30}	P_{10}
repping	**0.51**	**0.65**
tp_mt_f1	0.37	0.40
t_mt5_f2	0.37	0.40
osc_tok_vec	0.35	0.41
TUW-hyprid-cat	0.34	0.39
..	..	..
Baseline BM25	**0.25**	**0.29**

Table 4: Precision at 10

	2020	**2021**
precision at 1	0.76	0.80
precision at 2	0.77	0.77
precision at 3	0.79	0.73
precision at 4	0.80	0.71
precision at 5	0.77	0.69
precision at 10	0.73	0.65
precision at 15	0.72	0.61
precision at 20	0.69	0.59
precision at 30	0.67	0.56

Tables 2 and 3 show our results compared to the results of the two TREC Podcasts Track from 2020 and 2021. The complete list of precision values are shown in table 4. This shows that our method's precision values are significantly better than those of the other methods. The DCG_{30} is significantly better for the 2021 TREC Podcasts Track data, and the second best value for the 2020 TREC Podcasts Track.
Next, we examine the location in the result set that correctly answers the question. In the following analysis we also include the quality of the answer, see the table 6. 3/4 of the question is answered with the first hit (quality of the answer "better than nothing"). Almost 2/3 of the question is answered

Algorithm 1 Repping

Require: $M \neq \emptyset, R \neq \emptyset, Q \neq \emptyset$
Ensure: Result Set of ranked Segment RS

for each docID in M **do**
  $A \leftarrow R \bowtie_{R.t=Q.q \wedge R.docID=docID} Q$
  $B \leftarrow$ `clone of` A
  for each a in A **do**
    $start \leftarrow a.start$
    $end \leftarrow a.end$
    del Element a in B
    for each b in B **do**
      `score` $\leftarrow b.$`score` $\cdot\ a.$`score`
      if $b.start \geq start \wedge \leq end$ **then**
        $RS.add(b.$`docID`$, b.start, b.end,$ `score`$)$
      else if $start \geq b.start \wedge end \leq b.end$ **then**
        $RS.add(b.$`docID`$, start, end,$ `score`$)$
      else if $start \geq b.start \wedge end \geq b.end \wedge start \leq b.end$ **then**
        $RS.add(b.$`docID`$, start, b.end,$ `score`$)$
      else if $b.start \geq start \wedge b.end \geq end \wedge b.start \leq end$ **then**
        $RS.add(b.$`docID`$, b.start, end,$ `score`$)$
      end if
    end for
    $B \leftarrow RS$
  end for
end for
sort RS by `score` descending

Table 5: Result of known-item Questions

Question answered:	within the first hits	correct	
2020 15 Questions	1	10	67%
	2	11	73%
	3	12	80%
	10	13	87%
2021 10 Questions	1	3	30%
	2	5	50%
	3	5	50%
	10	6	60%
Summary 25 Questions	1	13	52%
	2	16	64%
	3	17	68%
	10	19	76%

with the first hit if the answer is better than fair. Here, it would still have to be examined whether there is an answer at all in the entire corpus that is Excellent (3) or Perfect (4).

Some questions of the data of the TREC Podcasts Track of 2020 and 2021 contain questions to which there should be only **"one"** correct answer (type of questions know-item). If we now take a closer look at these questions and examine at which position in the answer list the questions are answered, we get the following result which is shown in the table 5. 2/3 of these questions can be answered correctly within the first 3 hits.

Our results can be checked with the help of the search engine at, Demo Search. For this, you will need a key which can be requested via repping@repp.app. The key $asdjkjcil089zskcmh1092$ can be used for the duration of the conference. This is necessary because the corresponding rights to the podcast do not belong to the authors and may only be used for research purposes.

5 Conclusion

With our experiments we were able to show that our method provides significantly better results than all current methods, both with the data from the two TREC podcast tracks from 2020 and from 2021. The evaluation of the segments has been carried out according to the same PEGFB method as was used for the two TREC podcast tracks. Since the evaluation of the hits is also always a subjective decision, our results can be retrieved and reviewed at xx.xx. The two steps of segmentation and indexing are merged into one step in our method, or the corresponding segment is computed at runtime and according to the search query. The costs for the computations are low.

By using a synonym lexicon, the results were able to be further improved. The ranges of synonyms can be integrated into the search by a union set. Similarly, an increase in results could be achieved by including the titles and descriptors of the corresponding podcasts [40].

The parameters of the ranking formula k1 and b have been derived from the literature. Research work is still required here to optimally tune these parameters. Once this has been done, an improvement of the search results can be expected. This also applies to the parameters k2 and k3, which have been roughly estimated by initial experiments. Likewise, the $repp\text{-}index$ could be an input for a Deep Bidirectional Transformer (for example BERT). This would make it possible to capture and evaluate the documents in a much more far-reaching manner.

With our new approach, the dimension of the search can be significantly expanded, and it is now possible to really browse in a podcast (or in the documents); the boundaries of the document have been broken down. The granularity of the search is reduced to the smallest possible level, and the semantic relationships can now finally be mapped and thus captured. Likewise, it is possible to visualize the index and locate areas of high relevance. The method delivers results in near real time for a combined keyword search as well as for complex questions. These results fit semantically with the corresponding question.

Our new method can achieve significant increases in segment retrieval and

Table 6: Position answered the Question correct.

	2020	2021
50 Question / Answer better than Bad (1,2,3,4)		
Question answered with the 1 hit	76%	80%
Question answered within the first 5 hits	92%	90%
Question answered within the first 10 hits	94%	92%
50 Question / Answer bether than Fair (2,3,4)		
Question answered with the 1 hit	64%	66%
Question answered within the first 5 hits	86%	86%
Question answered within the first 10 hits	94%	90%
50 Question / Answer better than Good (3,4)		
Question answered with the 1 hit	38%	30%
Question answered within the first 5 hits	60%	48%
Question answered within the first 10 hits	70%	64%

thus in the capturing of content. As a result, it is much easier to retrieve voice messages, lectures, videos and podcasts and, more importantly, search them in a much more targeted manner. This provides many advantages for digital learning, the indexing of audio books, the understanding of lectures and much more. On the other hand, there are disadvantages to the technology, which makes it much easier to monitor people and what they say. Here, the undesirable statements made by target persons could be quickly filtered out from large amounts of data. We very much hope that our findings will be used in a positive sense for the development of free democratic societies. This method is not limited simply to podcasts, but can be applied to all kinds of text, eg. transcripts of videos, especially in our past publications lecture videos in german language [23, 25, 26, 30].

6 Limitation

The collection of test data sets for the two TREC podcast tracks includes a wide range of topics and formats. However, the podcasts come from only one provider that offers common commercial podcast to a wide range of listeners. Accordingly, specialized areas such as scientific lectures, highly theoretical discussions, parliamentdebates, songs and more are not represented in our test data.

The tests are based on podcast in the English language. It can be assumed that the other Indo- European language families are subject to the same basic law of word repetitions (*repps*) and therefore our method is equally promising here. But what laws govern other language families and other special subjects?

As previously mentioned, the optimal parameters of the *score* function for these other language families and likewise for special formats must be examined and adapted if necessary. This question has not been answered by our work and thus provides impetus for new research topics.

Our method is based on text. It is thus not only possible to index podcasts, but also text archives and film libraries. This could not be examined with the underlying test data.

References

[1] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-Based Document Segmentation with Probabilistic Latent Semantic Analysis. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (McLean, Virginia, USA) *(CIKM '02)*. Association for Computing Machinery, New York, NY, USA, 211–218. doi:10.1145/584792.584829

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.

[3] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805

[4] Dominik Flejter, Karol Wieloch, and Witold Abramowicz. 2007. Unsupervised Methods of Topical Text Segmentation for Polish. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*. Association for Computational Linguistics, Prague, Czech Republic, 51–58. https://aclanthology.org/W07-1707

[5] Petra Galuscáková, Suraj Nair, and Douglas W. Oard. 2020. Combine and Re-Rank: The University of Maryland at the TREC 2020 Podcasts Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication, Vol. 1266)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec29/papers/UMD_IR.P.pdf

[6] M.A.K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London, UK. doi:10.4324/9781315836010

[7] Arezki Hammache and Mohand Boughanem. 2021. Term position-based language model for information retrieval. *Journal of the Association for Information Science and Technology* 72, 5 (2021), 627–642. arXiv:https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.24431 doi:10.1002/asi.24431

[8] Xiangji Huang, Fuchun Peng, Dale Schuurmans, Nick Cercone, and Stephen E. Robertson. 2003. Applying Machine Learning to Text Segmentation for Information Retrieval. *Inf. Retr.* 6, 3-4 (2003), 333–362. doi:10.1023/A:1026028229881

[9] Gareth J. F. Jones. 2019. *About Sound and Vision: CLEF Beyond Text Retrieval Tasks*. Springer International Publishing. 307–329 pages.

[10] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments - Part 2. *Inf. Process. Manag.* 36, 6 (2000), 809–840. doi:10.1016/S0306-4573(00)00016-9

[11] Jussi Karlgren, R Jones, B Carterette, A Clifton, M Eskevich, GJF Jones, Sravana Reddy, Edgar Tanaka, and MI Tanveer. 2022. TREC 2021 Podcasts Track Overview. In *Text REtrieval Conference (TREC)*. NIST Special Publication. https://trec.nist.gov/pubs/trec30/trec2021.html

[12] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the Carbon Emissions of Machine Learning. *arXiv preprint arXiv:1910.09700* (2019).

[13] Yuanhua Lv and ChengXiang Zhai. 2009. Positional language models for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 299–306. doi:10.1145/1571941.1571994

[14] Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics* 17, 1 (1991), 21–48.

[15] Hyo-Jung Oh, Sung Hyon Myaeng, and Myung-Gil Jang. 2007. Semantic passage segmentation based on sentence topics for question answering. *Information Sciences* 177, 18 (2007), 3696–3717. doi:10.1016/j.ins.2007.02.038

[16] Paul Owoicho and Jeff Dalton. 2020. Glasgow Representation and Information Learning Lab (GRILL) at TREC 2020 Podcasts Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication, Vol. 1266)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec29/papers/uog_msc.P.pdf

[17] Oyebade K. Oyedotun and Adnan Khashman. 2016. Document Segmentation Using Textural Features Summarization and Feedforward Neural Network. *Applied Intelligence* 45, 1 (jul 2016), 198–212. doi:10.1007/s10489-015-0753-z

[18] Deepak P., Karthik Visweswariah, Nirmalie Wiratunga, and Sadiq Sani. 2012. Two-Part Segmentation of Text Documents. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (Maui, Hawaii, USA) *(CIKM '12)*. Association for Computing Machinery, New York, NY, USA, 793–802. doi:10.1145/2396761.2396862

[19] Irina Pak and Phoey Lee Teh. 2018. *Text Segmentation Techniques: A Critical Review*. Springer International Publishing, Cham, 167–181. doi:10.1007/978-3-

319-66984-7_10

[20] Moayad Yousif Potrus, Umi Kalthum Ngah, and Bestoun S. Ahmed. 2014. An evolutionary harmony search algorithm with dominant point detection for recognition-based segmentation of online Arabic text recognition. *Ain Shams Engineering Journal* 5, 4 (2014), 1129–1139. doi:10.1016/j.asej.2014.05.003

[21] Stephan Repp. 2009. *Extraktion von semantischen Informationen aus audiovisuellen Vorlesungsaufzeichnungen : Sprachtranskripte der Vorlesungsvideos als Informationsressource.* dissertation. Hasso Plattner Institut, Universität Potsdam. magna cum laude.

[22] Stephan Repp, Andreas Groß, and Christoph Meinel. 2008. Browsing within Lecture Videos Based on the Chain Index of Speech Transcription. *IEEE Trans. Learn. Technol.* 1, 3 (2008), 145–156.

[23] Stephan Repp, Serge Linckels, and Christoph Meinel. 2008. Question answering from lecture videos based on an automatic semantic annotation. In *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008, Madrid, Spain, June 30 - July 2, 2008*, June Amillo, Cary Laxer, Ernestina Menasalvas Ruiz, and Alison Young (Eds.). ACM, 17–21.

[24] Stephan Repp, Serge Linckels, and Christoph Meinel. 2008. Question Answering from Lecture Videos Based on Automatically-Generated Learning Objects. In *Advances in Web Based Learning - ICWL 2008, 7th International Conference, Jinhua, China, August 20-22, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 5145)*, Frederick W. B. Li, Jianmin Zhao, Timothy K. Shih, Rynson W. H. Lau, Qing Li, and Dennis McLeod (Eds.). Springer, 509–520.

[25] Stephan Repp and Christoph Meinel. 2006. Segmenting of Recorded Lecture Videos - The Algorithm VoiceSeg. In *SIGMAP 2006 - Proceedings of the International Conference on Signal Processing and Multimedia Applications, Setúbal, Portugal, August 7-10, 2006, SIGMAP is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, Pedro A. Amado Assunção and Sérgio M. M. de Faria (Eds.). INSTICC Press, 317–322.

[26] Stephan Repp and Christoph Meinel. 2006. Semantic Indexing for Recorded Educational Lecture Videos. In *4th IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2006 Workshops), 13-17 March 2006, Pisa, Italy*. IEEE Computer Society, 240–245. doi:10.1109/PERCOMW.2006.122

[27] Stephan Repp and Christoph Meinel. 2008. Segmentation of Lecture Videos Based on Spontaneous Speech Recognition. In *Tenth IEEE International Symposium on Multimedia (ISM2008), December 15-17, 2008, Berkeley, California, USA*. IEEE Computer Society, 692–697. doi:10.1109/ISM.2008.20

[28] Stephan Repp and Christoph Meinel. 2008. Segmentation of Lecture Videos Based on Spontaneous Speech Recognition. In *Tenth IEEE International Symposium on Multimedia (ISM2008), December 15-17, 2008, Berkeley, California, USA*. IEEE Computer Society, 692–697.

[29] Stephan Repp and Christoph Meinel. 2009. Automatic Extraction of Semantic Descriptions from the Lecturer's Speech. In *Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), 14-16 September 2009, Berkeley, CA, USA*. IEEE Computer Society, 513–520.

[30] Stephan Repp, Jörg Waitelonis, Harald Sack, and Christoph Meinel. 2007. Segmentation and Annotation of Audiovisual Recordings Based on Automated Speech Recognition. In *Intelligent Data Engineering and Automated Learning - IDEAL 2007, 8th International Conference, Birmingham, UK, December 16-19, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4881)*, Hujun Yin, Peter Tiňo, Emilio Corchado, William Byrne, and Xin Yao (Eds.). Springer, 620–629. doi:10.1007/978-3-540-77226-2_63

[31] J. Reynar. 1998. *Topic Segmentation: Algorithm and applications.* Ph. D. Dissertation. University of Pennsylvania, Pennsylvania.

[32] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. 1995. Large Test Collection Experiments on an Operational, Interactive System: Okapi at TREC. *Inf. Process. Manag.* 31, 3 (1995), 345–360. doi:10.1016/0306-4573(94)00051-4

[33] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. 1998. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive. In *Proceedings of The Seventh Text REtrieval Conference, TREC 1998, Gaithersburg, Maryland, USA, November 9-11, 1998 (NIST Special Publication, Vol. 500-242)*, Ellen M. Voorhees and Donna K. Harman (Eds.). National Institute of Standards and Technology (NIST), 199–210.

[34] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994 (NIST Special Publication, Vol. 500-225)*, Donna K. Harman (Ed.). National Institute of Standards and Technology (NIST), 109–126. http://trec.nist.gov/pubs/trec3/papers/city.ps.gz

[35] Martin Scaiano, Diana Inkpen, Robert Laganière, and Adele Reinhartz. 2010. Automatic Text Segmentation for Movie Subtitles. In *Advances in Artificial Intelligence*, Atefeh Farzindar and Vlado Kešelj (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 295–298.

[36] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM* 63, 12 (nov 2020), 54–63. doi:10.1145/3381831

[37] Fei Song, William M. Darling, Adnan Duric, and Fred W. Kroon. 2011. An Iterative Approach to Text Segmentation. In *Advances in Information Retrieval*, Paul Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Mudoch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 629–640.

[38] K. Sparck Jones, S. Walker, and S.E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing and Management* 36, 6 (2000), 779–808. doi:10.1016/S0306-4573(00)00015-7

[39] Nicola Stokes. 2004. *Applications of Lexical Cohesion Analysis in the Topic Detection and Tracking Domain.* Ph. D. Dissertation. Department of Computer Science, University College Dublin, Dublin.

[40] Yongze Yu, Jussi Karlgren, Ann Clifton, Md. Iftekhar Tanveer, Rosie Jones, and Hamed R. Bonab. 2020. Spotify at the TREC 2020 Podcasts Track: Segment Retrieval. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication, Vol. 1266)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST).

The definitive Version of Record was published in *Proceedings of the 2024 International Conference on Multimedia Retrieval (ICMR '24)*,
DOI: https://doi.org/10.1145/3652583.3658047