Information Retrieval Report: Spotify Podcast Segment Retrieval

Alexander Belooussov Mateo Sierens

December 2022

1 Introduction

Podcasts are a rapidly growing audio-only medium that involves new patterns of usage and new communicative conventions and motivate research in many new directions [2, 5]. With Spotify adding podcasts to their music streaming platform in 2015, many podcasts became widely available to the average user. For the purpose of research, Spotify has made a dataset of 100000 podcast episodes available in both English and Portuguese. Transcripts were created from the audio using automatic speech recognition tools. The transcripts can be used for many different tasks, such as podcast search.

This is exactly what was done for the TREC Podcasts Track in 2020 and 2021 [3]. Two tasks were defined: segment retrieval and summarization. Segment retrieval is defined as follows: given a query, find a matching 2-minute segment within a podcast that the user desires. The segments always start on a minute (so a segment is 0-120, 60-180, 120-240, etc.). For the summarization task, participants had to generate short text transcripts that captured the essence of the episode. In this project, we focussed on the segment retrieval task. Important to note is that the Podcast Track has already concluded, so the results from participants are available online in the form of a paper. We decided to try both approaches that seemed to work, and novel approaches and compare the scores with a baseline, as well as the participants' results.

1.1 KeyBERT

A novel idea that we had was the use of KeyBERT. KeyBERT is a minimal and easy-to-use keyword extraction technique that leverages BERT embeddings to create keywords and key phrases that are most similar to a document [7]. KeyBERT is not a variation on BERT nor a neural model in itself. It uses BERT embeddings and cosine similarity to extract keywords from documents. First, document embeddings are extracted using BERT. Then all n-grams are generated for the document. For each of these n-grams, an embedding is made, again with BERT. These embeddings are compared to the document embedding

using cosine similarity. The n-grams with the highest similarity are picked as keywords/keyphrases. A visualization has been made in figure 1.

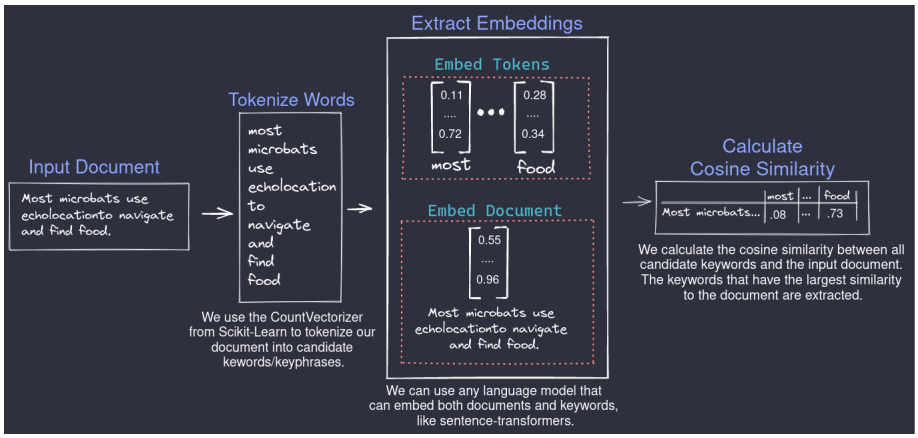

Figure 1: KeyBERT keyword extraction process [7]

The idea behind keywords being useful in this specific task is that queries are very short, sometimes consisting of only a couple of words. Meanwhile, the transcripts contain a lot of information about multiple topics. By extracting the keywords from these transcripts, the key concepts of an episode can be described in a couple of keywords which can be compared against the short query. There are multiple available methods for extracting keywords. The author himself states that KeyBERT is in no way unique, but what makes it different is that most of the other BERT-based methods require training a model from scratch. KeyBERT is very much plug-and-play, only requiring 3 lines to extract keywords but also having a lot of options for fine-tuning.

2 Dataset

In this section, we describe in detail the data which was available to us and which parts were used.

2.1 Transcripts

The podcast data consists of 4 parts: audio, transcripts, show RSS information, and metadata. We do not use the audio at all (the file size is also too big for any playful experiments). Each podcast has an automatically generated transcript. Each transcript is split up into multiple segments of varying lengths, amount of sentences, and start times. Multiple alternative transcripts may be provided, however, we did not find any examples where that is the case. In any case, we take the first element in the "alternatives" list. Each word has a start and end

time. The start times are crucial for the segment retrieval task. An example excerpt of a transcript is:


```
{
  "alternatives": [{
        "transcript": " Exquisitely crafted and quite well
        preserved despite 3000 years of being hidden underground.
        It was the profile of a royal woman wearing a crown. She
        had a thin neck and a face that was clearly elegant
        despite Millennia of wear and tear. This was the face
        of Queen Nefertiti since its Discovery. The bust has
        become one of the most common images associated with
        the study of ancient Egypt.",
        "confidence": 0.7545205950737,
        "words": [
              {
                  "startTime": "30s",
                  "endTime": "30.900s",
                  "word": "Exquisitely"
              },
              {
                  "startTime": "30.900s",
                  "endTime": "31.600s",
                  "word": "crafted"},
              {
                  "startTime":
                  "31.600s",
                  "endTime": "31.800s",
                  "word": "and"
              },
              ...]
        }]
}
```


The metadata is a tsv with the following information: show uri, show name, show description, publisher, language, rss link, episode uri, episode name, episode description, duration, show filename prefix, and episode filename prefix. This allows us to add additional information about the show and episode such as the title and description for each transcript. it also allows us to find the transcripts in the given file structure and provides each episode with an identifier in the form of the episode filename prefix. An example of the metadata is (reformatted to fit):


```
spotify:show:1JFy0Wo5Wd9d4OGHi4JV3O
Dougherty Show
"THE GREATEST PODCAST ON EARTH! Derek Dougherty and his friends
    convey unique perspectives that can only be heard on the
```

```
    ""Dougherty Show "" Brett had his body sculpted by Greek gods
    themselves. Zeus is probably his father. Dwyer is a cannabis
    enthusiast who just keeps it frank. Danny... Brother Dan by
    day, Danaconda at night. TAnd Derek is just here to bring a
    smile to everyone! The ""Dougherty Show "" vows to educate,
    fascinate, and amuse. Enjoy!  Support this podcast:
    https://anchor.fm/dougherty-show/support"
Dougherty Show
['en-US']
https://anchor.fm/s/a2e8cf4/podcast/rss
spotify:episode:6G53MbXuaJOg5eEYW7nzaI
Episode 3
Black Holes, AAF fallout, Kev Hart and Amy Schumer's Stand Up,
    How Derek got a bead stuck in his ear when he was a kid,
    The Joker Trailer and Logic's new album and Book!  ---
    Support this podcast:
    https://anchor.fm/dougherty-show/support
76.35611666666667
show_1JFy0Wo5Wd9d4OGHi4JV3O
6G53MbXuaJOg5eEYW7nzaI
```

The RSS information is an XML file containing information about the author, language, title, description, category, etc. for each show. Additionally, each show has information about its episodes, containing the title, episode number, description, summary, etc. We do not use this data since all the information that we use is contained in the metadata and transcripts.

2.2 Queries

The queries are given in an XML file which can be found on the TREC Podcast website[3]. There are 2 available sets of queries for the 2020 task. The first is a training set, which contains 8 queries. Presumably, this was the set for which the true relevance of each segment was available beforehand. The other available set of queries is labeled as the test set and contains 50 queries. Presumably, this is the set on which submissions were based and for which no true relevances were available (as they are now). We simulate the participation by only ever testing our intermediate progress on the train set while providing 5 runs (+ a baseline) on the test set.

Each query contains 4 fields. The first is the "num" field for identifying the query. The "query" field is the main query and is usually around 5 words long. The "type" field gives the category of the query. Lastly, the "description" field describes the query in a few sentences. We often concatenate the description to the query to provide more context.

There are 3 different types of queries in the dataset. The first type is "topical". These are queries that do not match one particular podcast but aim to find any podcast that matches a given topic. This type covers 6 out of 8 training

queries and 35 out of 50 test queries. An example of such queries is:

```
<topic>
    <num>3</num>
    <query>black hole image</query>
    <type>topical</type>
    <description>
        In May 2019 astronomers released the first-ever
        picture of a black hole. I would like to hear
        some conversations and educational discussion
        about the science of astronomy, black holes,
        and of the picture itself.
    </description>
</topic>
```

The second type is "refinding". These are queries of which the user knows the existence, but would like to find again. This type has 1 training query and 8 test queries. An example would be:

```
<topic>
    <num>4</num>
    <query>story about riding a bird</query>
    <type>refinding</type>
    <description>
        I remember hearing a podcast that had
        a story about a kid riding some kind of bird.
        I want to find it again.
    </description>
</topic>
```

The last query type is "known item". This is very similar to the previous type, except that other similar podcasts are also of interest. This type has 1 training query and 7 test queries. An example is:

```
<topic>
    <num>5</num>
    <query>daniel ek interview</query>
    <type>known item</type>
    <description>
        Someone told me about a podcast interview with
        Daniel Ek, CEO of Spotify, about the founding
        and early days of Spotify. I would like to find
        the show and episode that contains that interview.
        Other interviews with Ek are relevant as well.
    </description>
</topic>
```

It should be noted that "refinding" and "known item" were merged into a single category for the 2021 task.

2.3 Test Data

Ground truth data is also available on the TREC Podcast website [3]. This data is crucial for us to be able to evaluate our results. We limit ourselves to the 2020 task due to the unavailability of the 2021 test data (404 error). The ground truth consists of relevance assessments made by NIST assessors. A PEGFB scale (Perfect, Excellent, Good, Fair, Bad) is used. The perfect grade is only applicable to refinding and known item queries. Note that we consider a segment or podcast relevant if it gets at least a rating of 1 (fair). The data is given in tsv format, where the columns are query num, query type, episode, and relevance. The episode info contains the "episode filename prefix" joined with the start time. An example of the ground truth data is:

```
1 0 spotify:episode:0E2nqCXMkS218SE72APmNr_240.0 2
1 0 spotify:episode:0E2nqCXMkS218SE72APmNr_300.0 2
1 0 spotify:episode:0E2nqCXMkS218SE72APmNr_360.0 2
1 0 spotify:episode:0Th494DvnO5dU8vTi3QHm2_120.0 1
1 0 spotify:episode:199bOiXLOl4YsRaSNDNXvP_1200.0 2
1 0 spotify:episode:1ZA1QTtylexrVt75xiprNH_1020.0 2
1 0 spotify:episode:1ZA1QTtylexrVt75xiprNH_1500.0 2
1 0 spotify:episode:1ZA1QTtylexrVt75xiprNH_840.0 2
1 0 spotify:episode:1ZA1QTtylexrVt75xiprNH_900.0 2
1 0 spotify:episode:1ZA1QTtylexrVt75xiprNH_960.0 2
1 0 spotify:episode:1oOHCHjNtRWdEWRTWJhzyC_660.0 0
1 0 spotify:episode:1oOHCHjNtRWdEWRTWJhzyC_720.0 0
1 0 spotify:episode:1sbXZEnTGyGKNfpEogahoP_180.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_180.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_240.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_300.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_360.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_420.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_480.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_60.0 2
1 0 spotify:episode:1t6qb1pD2AfVVjyLn1Wisr_660.0 2
1 0 spotify:episode:1tBwWZksIEKXKakbizSLWe_60.0 2
...
```

—

It should be noted that the query type field is always set to 0 in the ground truth data. Additionally, each segment starts exactly at a multiple of 60. The values in the relevance column range from 0 (bad) to 4 (perfect).

3 First approach

In our first approach, we made 3 different indices on entire transcripts. The first index contains just the transcripts, the second additionally contains the episode

name and description, and the third also contains the show and episode name and description. In short, we use BM25, implemented in Pyserini [9], to fetch a certain amount of relevant podcast transcripts. These transcripts then get split into 2-minute-long segments, with 60 seconds overlap. These segments are then, together with the query, passed to KeyBERT and/or a BERT reranker. The score obtain from these transformers are then aggregated to achieve a final ranking. This process can be summarized in Figure 5.

3.1 BM25 extensions

The goal of this search is to find as many relevant podcasts as possible while retrieving the smallest amount of podcasts possible. Figure 2 shows how recall evolves with respect to the number of retrieved documents for a simple BM25 search. We can see that adding the show and episode information does help the search, but using the query description does not provide an immediate benefit.

We can increase performance by modifying the query using KeyBERT [7]. If we pass the query and its description to KeyBERT, it will find the most important n-grams together with a score. These n-grams can then be used as query terms, while the score is used to add a custom weight to each term. Increasing the n-gram length to include 3 words additionally increases the performance. Figure 3 Shows the improved recall curve when using KeyBERT.

Lastly, we can expand the query terms that we retrieve from KeyBERT using WordNet. Using WordNet, we retrieve synonyms, hypernyms, and hyponyms of words in the query terms. The weight of the newly found terms gets adjusted with regard to the amount of change from the original query. The new weight is given by the formula:

$$
\begin{aligned}
base &= \min(\text{len}(query_{original}, \text{len}(query_{new}) + 1 \\
dif &= \#\text{words from } query_{new} \text{ not in } query_{original} \\
weight_{new} &= \frac{weight_{original}}{base^{dif}}
\end{aligned}
$$

Intuitively, this means that a query with 2 words, of which one is changed to a synonym, is half as important as the original query. To limit the number of query terms, we only look at (at most) 3 synonyms, 3 hypernyms, and 3 hyponyms per word. The rightmost graph in Figure 3 shows the recall curve when using query expansion. Note how the recall is drastically improved for documents with perfect and excellent relevance. The curve does not improve for documents that have a relevance of good or fair. Good and fair documents largely relate to topical queries, where the set of appropriate segments is quite broad. The increase is mostly felt in documents for the refinding and known item queries. This led us to only apply query expansion for these types of queries (what we call type-aware query expansion).

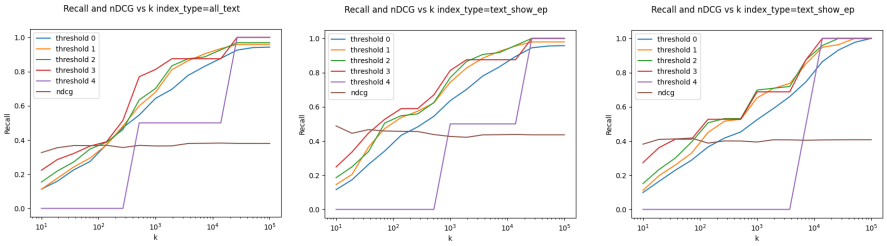

Figure 2: Left: Base BM25; Middle: BM25 using an index containing show and episode name and description; Right: BM25 with show and episode name and description and using query description; Each line represents the ability to find transcripts that have at least one segment of certain relevance. For example, there are only 2 documents that contain a segment with relevance 4.

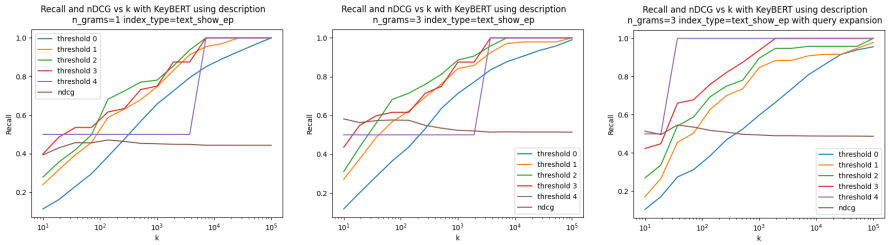

Figure 3: Left: Using KeyBERT; Middle: Using KeyBERT and keywords up to length 3; Right: KeyBERT and query expansion using WordNet

3.2 KeyBERT

As mentioned above, we can find a set of query terms using KeyBERT [7], but KeyBERT can also be used to rank the retrieved segments. KeyBERT allows setting a limited group of candidate keywords/keyphrases. This effectively allows us to use the model, which is the suggested pre-trained BERT model in our case, to calculate the similarity between the previously calculated set of query terms (passed as candidates) and the segment. KeyBERT will then return a subset of the candidates with a score. These scores are summed to get the score of the segment. Finally, the scores across all segments are scaled to lie between 0 and 1.

3.3 BERT reranker

Reranking using a BERT model is a common approach. We use the "amberoad/bert-multilingual-passage-reranking-msmarco" pre-trained model [1]. It is a sequence classification model, meaning that it can predict whether 2 strings are

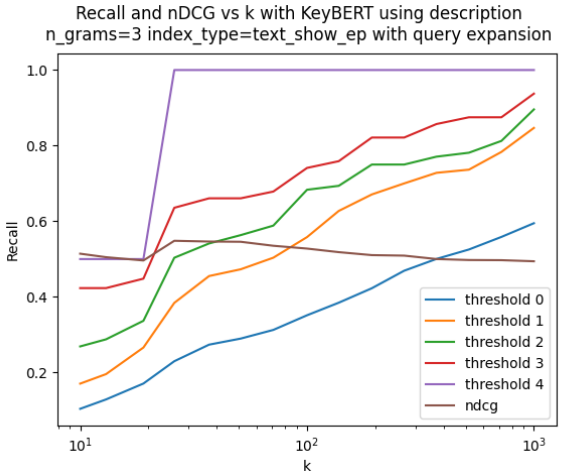

Figure 4: Recall curve for query expansion, looking at smaller values of k

related to each other. In this case, it is trained on the MS MARCO data set[4], which contains questions and answers from Bing. We pass the query and its description as the first sequence, and the segment with the show and episode name and description as the second sequence. It generates a score in the range of -10 to 10, which we then scale so that all scores lie between 0 and 1.

3.4 Score aggregation

We have implemented 4 different strategies to combine the different scores: rerank only, mean, min, and max. Mean takes the mean of all scores, max takes the highest available score, and min takes the lowest available score. Rerank only will ignore all scores except the one generated by the BERT reranker. Intuitively, taking the mean makes sense because the final ranking is based on all the scores. Taking the lowest value means that all models need to agree that a given segment is relevant, which makes this a valid way to combine the scores. Taking the highest value is prone to errors since a model can falsely claim a given segment is relevant, and the other models will have no say on the matter.

3.5 Results on train data

Some results of the training queries can be found in Table 1. All runs fetched 30 full podcast transcripts which were then segmented and ranked. This number was chosen based on the resulting recall when applying query expansion (see Figure 4). We retrieve the 1000 top segments as this is also the limit for submissions on the TREC task. The index on transcripts, show name and description, and episode name and description is chosen.

The best-scoring run has a recall that is higher for the refinding and known

item queries, while also scoring well on nDCG on all query types. Adding query expansion to topical queries negatively affects both recall and nDCG.

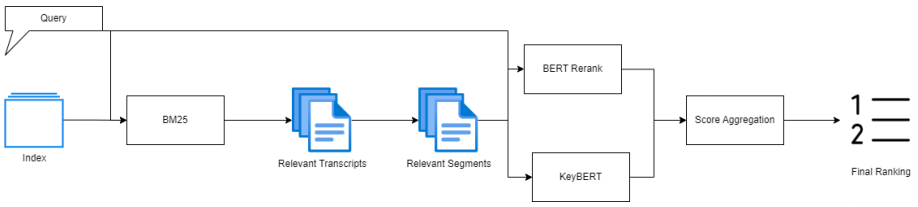

Figure 5: Schematic overview of our first approach

4 Second approach

This approach reuses a lot of the components which are mentioned above. The biggest difference is that now the index is not made on entire podcast transcripts, but on 2-minute-long segments instead. This also means that we can use the scores that are associated with the initial BM25 score in our final ranking. This method is shown schematically in Figure 6.

We have once again made it possible to augment the BM25 search with KeyBERT and query expansion using WordNet. The implemented aggregation strategies are the ones mentioned in the previous approach.

4.1 Baselines

Since the index is made on segments, it is possible to create a few baseline runs using only BM25[9]. We ran this on the 3 different segment indices to create the results in Table 1.

4.2 Results on train data

The results of the train queries can be found in Table 1. We find that query expansion does not have the same benefit as it did with the previous method. We also include a run where we doubled the number of segments that were retrieved by BM25. After this retrieval, all 2000 segments were ranked with BERT and KeyBERT and only the top 1000 were considered for the calculation of the results. We used the index containing show and episode names and descriptions unless stated otherwise.

5 Test Results

We simulate participation in the TREC task by only running 5 tests using the test queries. The parameters were chosen based on the results of the training

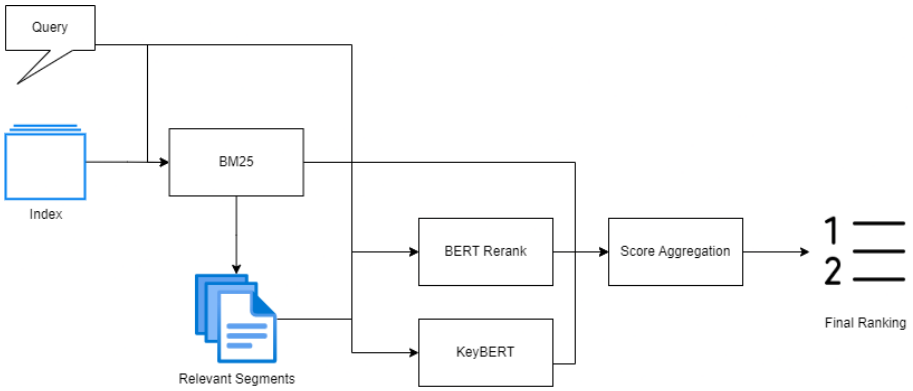

Figure 6: Schematic overview of our second approach

queries. Table 2 compares our results to some baselines and the best submission to the TREC Podcasts Track[6]. Figure 7 shows the scores of all submissions to the TREC Podcasts Track[8]. We retrieve up to 1000 results per query. Our test runs are better than the majority of submissions and are competitive with the best ones.

	Mean nDCG	Topical nDCG	Refinding nDCG	Known Item nDCG	Total Relevant Retreived (/217)	Topical Relevant Retreived (/188)	Refinding Relevant Retreived (/15)	Known Item Relevant Retreived (/14)
BM25 Baseline	0.4794	0.5787	0.2236	0.1399	172	160	10	2
Bm25 Baseline Ep info index	0.4519	0.5033	0.2619	0.3333	176	167	8	1
BM25 Baseline Show & Ep info index	0.4497	0.5080	0.2706	0.2789	172	165	6	1
First Approach KeyBERT	0.4635	0.5425	0.0000	0.4531	129	115	0	**14**
First Approach Rerank	0.4337	0.5460	0.0000	0.1934	106	105	0	1
First Approach KeyBERT, Rerank, Mean score	0.5425	0.6237	0.0000	0.5978	129	115	0	**14**
First Approach KeyBERT, Rerank, QE, Mean score	0.6354	0.6110	0.7610	**0.6556**	126	109	5	12
First Approach KeyBERT, Rerank, type-aware QE, Min score	**0.6534**	0.6318	**0.8243**	0.6119	132	115	5	12
First Approach KeyBERT, Rerank, type-aware QE, Mean score	0.6449	0.6237	0.7610	**0.6556**	132	115	5	12
Second Approach KeyBERT, Mean	0.5078	0.5664	0.3596	0.3047	186	166	11	9
Second Approach Rerank. Mean	0.6134	**0.7109**	0.3257	0.3155	172	165	6	1
Second Approach KeyBERT, Rerank, Mean	0.5746	0.6298	0.3638	0.4541	186	166	11	9
Second Approach KeyBERT, Rerank, QE, Mean	0.5951	0.6596	0.3899	0.4130	175	160	10	5
Second Approach KeyBERT, Rerank, Min	0.5477	0.6014	0.3178	0.4559	186	166	11	9
Second Approach KeyBERT, Rerank, Mean, No show/ep index	0.6102	0.6880	0.3014	0.4523	188	169	11	8
Second Approach KeyBERT, Rerank, Mean, Ep info index	0.5696	0.6248	0.3727	0.4351	188	169	10	9
Second Approach KeyBERT, Rerank, Mean, 2000 seg	0.5702	0.6222	0.3635	0.4652	**196**	**172**	**12**	12
Second Approach KeyBERT, Rerank, Mean, 2000 seg, No show/ep index	0.6056	0.6846	0.2917	0.4459	191	171	11	9

Table 1: Results of varying runs on the training data.

	Mean nDCG	Topical nDCG	Refinding nDCG	Known Item nDCG	Total Relevant Retreived (/2096)	Topical Relevant Retreived (/1626)	Refinding Relevant Retreived (/138)	Known Item Relevant Retreived (/332)	Precision@5	Precision@10	Precision@20	Precision@30
TREC BM25 Baselin	0.5218				1621				0.4958	0.4875	0.4510	0.4139
Our BM25 Baseline Transcript only index	0.5960	0.6223	0.5084	0.5641	1686	1292	118	276	0.4680	0.4700	0.4360	0.4020
UMD IR run3 (TREC best)	0.6682				1863				0.6292	0.5958	0.5531	0.5097
First Approach KeyBERT, Rerank, Min, Type-aware QE, Show & Ep info index	0.6042	0.6166	0.5270	0.6308	1083	804	54	225	0.5400	0.4680	0.3660	0.3073
First Approach KeyBERT, Rerank, Mean, Type-aware QE, Show & Ep info index	0.6197	0.6330	0.5456	0.6378	1086	806	54	226	0.5440	0.4640	0.3730	0.3187
Second Approach KeyBERT, Rerank Mean Show & Ep info index	0.6505	0.6700	0.5570	0.6601	1607	1211	118	278	0.5520	0.4920	0.4580	0.4053
Second Approach KeyBERT, Rerank Mean Transcript only index	0.6274	0.6727	0.5227	0.5202	1719	1320	125	274	0.5440	0.5000	0.4420	0.4027
Second Approach KeyBERT, Rerank Mean, 2000 seg Show & Ep info index	0.6477	0.6678	0.5537	0.6545	1695	1289	124	282	0.5520	0.5000	0.4600	0.4027

Table 2: Results of our test runs, compared to baselines and the highest scoring run that was submitted to the TREC Podcast Track.

	nDCG	nDCG at 30	precision at 10
UMD_IR_run3	0.67	0.52	0.60
UMD_ID_run4	0.66	0.49	0.56
UMD_IR_run1	0.62	0.45	0.53
UMD_IR_run5	0.65	0.50	0.58
UMD_IR_run2	0.59	0.42	0.51
run_dcu5	0.59	0.43	0.54
run_dcu4	0.58	0.42	0.54
run_dcu1	0.57	0.42	0.50
run_dcu3	0.57	0.42	0.50
run_dcu2	0.55	0.40	0.48
LRGREtvs-r_2 *	0.54	0.40	0.48
LRGREtvs-r_1 *	0.54	0.40	0.47
hltcoe4	0.51	0.43	0.54
LRGREtvs-r_3 *	0.50	0.32	0.41
hltcoe3	0.50	0.35	0.43
hltcoe2	0.47	0.38	0.45
hltcoe1	0.45	0.33	0.38
BERT-DESC-S	0.43	0.47	0.57
BERT-DESC-TD	0.43	0.47	0.56
BERT-DESC-Q	0.41	0.45	0.53
hltcoe5	0.38	0.30	0.37
UTDThesis_Run1	0.34	0.34	0.43
oudalab1	0.00	0.01	0.01
Baseline BM25	0.52	0.40	0.49
Baseline QL	0.52	0.40	0.48
Baseline RERANK-DESC	0.43	0.48	0.57
Baseline RERANK-QUERY	0.43	0.47	0.56

Figure 7: Results from submission to the TREC Podcasts Track[8]

6 Related Work

As stated the Podcast Track has concluded and the participants' results are available online. In this section, we will go over some approaches from the participants. All the papers are available here: `https://trec.nist.gov/pubs/trec29/trec2020.html`

In [10] the author explores the use of query expansion. As the transcriptions are generated by speech-to-text models, they are not fully reliable. Some words might be misinterpreted and make no sense, while they are important. In this work, they use 3 query expansion methods. The first method used the query descriptions to extract nouns and named entities and appended these to the query. The second approach consisted of using a web search engine (in this case

Google) to retrieve some documents related to the query and selecting the top 5 found terms based on a certain score assigned to each term within the retrieved documents. Their final approach was using WordNet for query expansion, which has also been discussed in this document.

As mentioned above, the best scores were achieved by [6]. They mainly focused on combining different approaches and using re-rankers. In their best submission, they performed different retrieval methods first. Then they combined the results of each of these methods, this result is called the lexical baseline. The lexical baseline is then re-ranked with 3 different methods, and these results are combined with the lexical baseline for the final result. This results in the top score in figure 7.

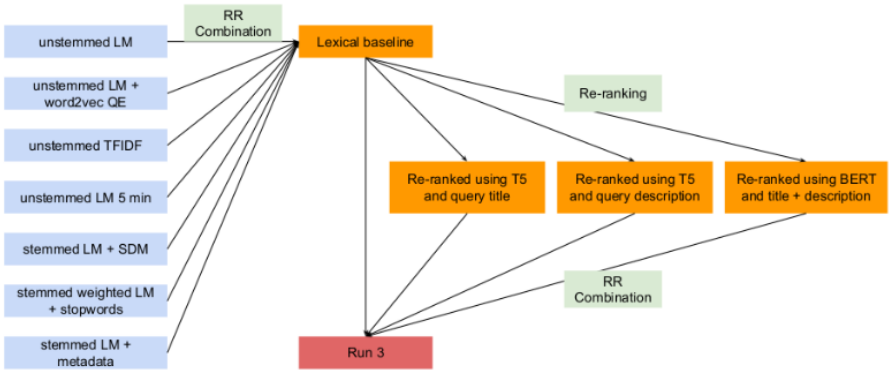

Figure 8: Combine Re-rank combine architecture [6]

7 Code

Code can be found at:
`https://github.com/AlexanderBelooussov/information-retrieval/`

References

[1] Passage reranking multilingual bert. `https://huggingface.co/amberoad/bert-multilingual-passage-reranking-msmarco`. Accessed: 2022-12-15.

[2] Spotify podcasts dataset. `https://podcastsdataset.byspotify.com/`. Accessed: 2022-12-16.

[3] Trec podcasts track. `https://trecpodcasts.github.io/`. Accessed: 2022-12-15.

[4] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268, 2016.

[5] Ann Clifton, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, Gareth Jones, Jussi Karlgren, Ben Carterette, and Rosie Jones. 100,000 podcasts: A spoken English document corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5903–5917, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[6] Petra Galuscáková, Suraj Nair, and Douglas W. Oard. Combine and rerank: The university of maryland at the trec 2020 podcasts track. In *TREC*, 2020.

[7] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020.

[8] R. Jones, Ben Carteree, Ann Clion, Maria Eskevich, G. Jones, Jussi Karlgren, Aasish Pappu, Sravana Reddy, and Yongze Yu. Trec 2020 podcasts track overview. *ArXiv*, abs/2103.15953, 2021.

[9] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362, 2021.

[10] Yasufumi Moriya and Gareth J.F. Jones. Dcu-adapt at the trec 2020 podcasts track. In *TREC*, 2020.