Automatic Linking of Podcast Segments to Topically Related Webpages

Carla McKeon[1], Claudio Rocha[1], and Gareth J. F. Jones[1,2(✉)]

[1] School of Computing, Dublin City University, Dublin, Ireland
{carla.mckeon32,claudio.rocha2}@mail.dcu.ie, gareth.jones@dcu.ie
[2] ADAPT Centre, Dublin City University, Dublin, Ireland

Abstract. Podcasts are becoming an increasingly popular source of information. However, they often rely on the topical knowledge of the listener in order for them to be fully understood. We describe an investigation into methods to augment the contents of podcasts with related information from the Web. We seek to identify webpages related to segments within a podcast. NLP techniques are used to analyze audio podcast transcripts and link these to related content. We propose and examine 10 methods for automatically generating search queries from transcript segments, which are then used to search for related content on the web. The relevance of retrieved webpages to retrieved content is evaluated using crowdsourcing via Amazon Mechanical Turk. Extracting key phrases directly from the podcasts using YAKE was the most successful approach with more than 90% returned pages assessed as relevant, with precision at rank 1 and rank 3 above 0.9.

Keywords: Automatic content linking · Key phrase extraction · Podcast summarization · Automatic query construction

1 Introduction

Podcasts are an increasingly popular form of audio media providing information, topical comment and entertainment to ever growing numbers of people. A podcast episode may cover multiple topics or themes over the course of its content. The full meaning of the topics discussed may not though be apparent to the listener if they do not have a reasonable background in the issue under discussion. In this situation the listener may turn to a web search engine, such as *Google* to seek further information about the topic in order to better understand the podcast. In this paper we investigate the development of a method to automatically link segments of podcast content to related webpages. These links would then be available to podcast listeners removing the need for them to carry out their own search if they wish to find further information about the content.

In order to identify content related to a podcast, a suitable query must first be created from the words spoken in the podcast. In our study we explore the use of number of NLP techniques to construct an effective query [1]. To retrieve

L. Longo and R. O'Reilly (Eds.): AICS 2022, CCIS 1662, pp. 381–393, 2023.
https://doi.org/10.1007/978-3-031-26438-2_30

webpages using these queries, we make use of the Google Search API. This returns a ranked list of webpages from the Google Search engine. To assess the relevance of the returned webpages to the segments of the podcast transcript used to construct the queries, we use crowdsourcing via Amazon Mechanical Turk (MTurk) [2,3]. Each online assessor judges relevance of the retrieved webpage for the segment on a scale between fully relevant and not relevant. These judgements are then used to evaluate the effectiveness of each query generation technique using standard precision based evaluation metrics. For this investigation we make use of a large collection of podcasts with transcripts made available for research purposes by Spotify [4].

This paper is structured as follows. The next section reviews existing work in podcast search, keyphrase extraction and content linking. Section 3 describes the creation of the experimental dataset used in our study, Sect. 4 introduces our automated query generation methods, Sect. 6 outlines our evaluation processes and the metrics used to evaluate the effectiveness of our automated linking methods, Sect. 7 gives experimental results and analysis, and Sect. 8 concludes the paper and makes suggestions for further work.

2 Related Work

In this section, we review relevant work in podcast search and summarization, information retrieval using automated query generation, and the development of test collections for the evalation of information retrieval.

2.1 Podcast Search and Summarization

Until recently there has been limited related research reported on the automated processing of podcasts for search and summarization. To encourage more work in this area Spotify released a large collection of podcasts with corresponding transcriptions created using automatic speech recognition (ASR) [4]. This collection contains on the order of 100,000 podcasts, and formed the basis of benchmark podcast tasks at the Text Retrieval Conference (TREC)[1] in 2020 and 2021 These tasks evaluated methods for the effective search of podcasts in response to user search queries and summarization of the podcast transcripts [5,6]. Recognising that podcasts are long and multi-topic, the search task focused on retrieval of relevant podcast segments using the ASR transcripts for a set of topical search queries. Participants were provided with 58 search queries describing user information needs The relevance of segments to each query was manually judged by assessors at NIST. Segments identified as relevant within this task form the starting point for our investigation of linking segments to related webpages.

Summarization methods are concerned with creation of concise shortened documents preserve the most salient information [7,8]. For our query generation processes, we begin by summarising podcasts segments to capture their key information and then extract queries containing this key information.

[1] https://trecpodcasts.github.io/.

2.2 Information Retrieval Through Key Phrase Extraction and Query Generation

The Retrieval from Conversational Dialogues (RCD) track at the FIRE 2020[2] conference explored the utility of information retrieval methods to retrieve more information about topics discussed in transcripts of interactive movie dialogues. The conversational structure of these movies is similar to that of many podcasts. Participants in the RCD task were required to retrieve a ranked list of potentially related documents from Wikipedia for a span of text [9]. The participation requirements for the RCD 2020 task [10] adopted a similar strategy to the one which we follow here. The main topics of discussion in movie dialogues needed to be identified using summarization methods. Key phrase extraction techniques were then used to create queries to search for related articles in Wikipedia. The first approach used different methods, mixing different techniques for summarization such as BERT and different key phrase extraction techniques such as YAKE and TextRank. Those models used a phrase frequency of 1–3 words. When comparing participant results, the method that obtained the best results was the one using a custom summary extraction method and the TextRank technique. For the second approach, only one key phrase extraction technique, TextRank, was tested, which was applied directly to the movie dialogue. After which the key topics extracted were again used as queries to retrieve relevant documents from the Wikipedia dataset.

There has been limited previous work on automated link creation. Probably the most closely related task to our work is the NTCIR-9 Crosslink task, which examined cross-lingual linking between Wikipedia documents [11].

2.3 Information Retrieval Test Collections

A standard test collection for the evaluation of information retrieval methods consists of a set of documents, a set of queries and the identities of the relevant documents from the collection for each query. The relevance of each document to the query must be judged manually since the contents of the query do not fully express the information need that it seeks to express. The relevance of each document to the query can either be judged in a binary manner as "relevant" or "non-relevant", or using a graded scale, where documents can be assessed between highly, partially, marginally or not relevant. A number of studies have been carried out examining the design of effective and reliable test collections for information retrieval. A key result for our study is that at least 25–50 queries are required for results to reliable [12]. For our investigation we thus sought to examine the effectiveness of link generation for at least 30 podcasts fragments from which we generated queries for automated search.

[2] https://rcd2020firetask.github.io/RCD2020FIRETASK/.

3 Experimental Dataset

In this section we introduce the Spotify podcast collection which forms the basis of our study, selection of the data from this collection used for our experimental investigation and preprocessing procedures applied to the selected data.

3.1 Podcast Dataset

As outlined earlier, we used the podcast dataset made available by Spotify for this investigation. This collection consists of 105,360 podcast episodes taken from 18,376 shows collected between January 2019 and March 2020. [4]. The podcast dataset can be requested from Spotify for research purposes[3].

The collection includes recordings of the original podcasts with an ASR transcript created using a standard online Google transcription service. The podcasts are all in English and vary in length, being on average 30 min long, with the shortest running for 10.5 s and the longest for over 5 h.

For its use in the TREC Podcasts tasks [5,6], the podcast collection was supplemented with 58 Topic statements intended to be representative of the sort of queries that a listener might issue to a podcast search engine. Since podcasts are long and listening to them to identify content relevant to such a query would be time consuming, the TREC search tasks focused on identifying relevant segments created from the podcast transcripts. Segments were created by dividing the transcripts 120 s pieces with a 50% overlap to ensure the presence of segments where topical content is contained with single segments, rather than being split between adjacent segments.

3.2 Content Selection

The TREC Podcast task defined highly relevant segments as being ideal entry points into the podcast for a listener, and as being fully on topic.

For our investigation we wished to focus on a set of segments which have significant topical interest for a defined topic. We thus decided to begin by selecting the segments from the TREC Podcast task assigned a relevance score of 4 for one of the search topics. This resulted in a set of 55 segments associated with 18 of the topics.

Since errors in the transcripts may impact on content analysis processes that will be used for the query construction stage, we further filtered these segments to remove those containing obvious ASR transcription errors. This filtering resulted in segments associated with all 18 topics remaining. Mindful of the cost of relevance assessment using crowdsourcing and the minimum number of queries required for reliable experimentation [12], we randomly selected 30 segments of the remaining segments for use in our investigation.

In an operational setting all segments in the podcast archive would be linked to related web content during the indexing setup of the podcast search engine in

[3] https://podcastsdataset.byspotify.com.

Table 1. Methods to extract queries

Method	Summ.	Keyword E.	Keyword G.
M1	BERT	YAKE	–
M2	BERT	Text Rank	–
M3	BERT	–	KeyBART
M4	BERT	–	T5-s.-OpenKP
M5	T5-P.Sum.	YAKE	–
M6	T5-P.Sum.	Text Rank	–
M7	T5-P.Sum.	–	KeyBART
M8	T5-P.Sum.	–	T5-S.-OpenKP
M9	–	YAKE	–
M10	–	Text Rank	–

advance of listeners entering search queries. This would of course include query construction with errorful podcast transcripts. Examining the impact of these transcription errors on the effectiveness of the linking process will be the focus on further experimental studies.

3.3 Data Preprocessing

In order for the transcripts to be used consistently in the construction of queries, they were preprocessed into a standard format and standard NLP analysis applied. Preprocssing involved tokenising the raw text into words. and converting each word to lower case to ensure that the system avoids interpreting the same word (e.g. “Book” and “book”) as two different words. Then converting the word into its base form using lemmatization, and combining this with part-of-speech (POS) tagging to give context to each token and to bring together different word forms (e.g. meeting - meet (core-word extraction), was - be (tense conversion to present), mice - mouse (plural to singular)).

4 Query Generation

In order to identify potentially relevant or interesting web content to link to podcast segments, a query representing the key content of the segment must be created for submission to a search engine to seek this content. The core element of our investigation is the proposal and evaluation of a number of methods for query generation from podcast transcripts.

Our methods can be grouped into three approaches: summarization followed by key phrase extraction, summarization followed by key phrase generation and key phrase extraction directly from the podcast segment. Overall 10 different query generation methods were examined, the details of these methods are summarized in Table 1. These methods are described in outline in the following subsections.

4.1 Summarization Methods

Since podcasts often contain colloquial, conversational, and noisy commercials and sponsorship segments, we examined a preprocessing step of summarization to reduce the transcript to its core content [13]. We investigated whether generating queries from segment summaries improves likelihood of relevance of the retrieved webpages. In this paper we examine two different summarization methods:

BERT. This is a pre-trained BERT model developed by Google's AI Language team and described in [14]. BERT has achieved groundbreaking results on various NLP tasks. In this paper, we use BERT for extractive summarization using the PyTorch transformers library (HuggingFace) to perform extractive summarization using the following process [15]: sentence embedding, running a clustering Algorithm (K-means), and locating the sentences near the cluster's centroid.

T5. The T5-podcast summarizer model is the result of fine-tuning t5-base [16] on the Spotify podcasts dataset [4]. This model is based on Google's T5, a text-to-text framework, pre-trained on the C4 Dataset (Colossal Clean Crawled Corpus). The main concept of this model is that the input and output will always be strings, while the output of the BERT model is a class label or span of the input.

4.2 Key Phrase Extraction Methods

The main objective of key phrase extraction in guery generation is to capture the main topic discussed in the podcast, including the key events reported, the entities involved in these events and, their outcome, impact and significance. The following constraints [17] were considered while fine-tuning the parameters in the key phrase extraction techniques:

- A key phrase can be a single word or a sequence of up to four consecutive words as they appear in the podcast
- A minimum of one and a maximum of three key phrases were selected
- A key phrase has to be either a noun phrase, verb, proper name or adjective

Extensive research surveys of key phrase extraction methods and comparison of their relative performance are provided in [18,19]. We implemented both an unsupervised statistical and a graph based approach as outlined below.

YAKE! Yet Another Keyword Extraction (YAKE) [20] is a statistical method which exploits frequency and positional information of each candidate key phrase (word n-grams not containing punctuation marks, nor starting or ending with a stop word).

Terms are scored by gathering all of the feature weights into a unique score as shown in Eq. 1.

$$Score(t) = \frac{T_{Rel} * T_{Position}}{T_{Rel} + \frac{TFreq_{Norm} + T_{Sentence}}{T_{Rel}}} \quad (1)$$

where T_{Rel} is a score given to the relatedness of the term in the context of the document, which downgrades terms that co-occur with unique terms in a given window size. Dividing both $TFreq_{Norm}$ + $T_{Sentence}$ by T_{Rel} gives a higher value to terms that appear more frequently and in many sentences. The level of importance of a term is higher as long as it is relevant, i.e. has a low T_Rel and achieves a high score on $TFreq_{Norm}$ + $T_{Sentence}$ score. T_{Rel} * $T_{Position}$ takes the position of the term in the sentence into account multiplying it by the term's relevance to context score.

The score for a candidate key phrase $k = t_1 t_2 ... t_n$ is then computed as shown in Eq. 2.

$$Score(k) = \frac{\prod_{i=1}^{\infty} Score(t_i)}{frequency(k) * (1 + \prod_{i=1}^{\infty} Score(t_i))} \quad (2)$$

We use the LIAAD[4] version of YAKE with parameters tuned as follows: setting a window size of 2, which is used to capture the context of the phrases, and threshold to 0.8 which helps to eliminate redundant queries using the concept of Levenshtein distance. The top three key phrases were extracted as representing the core content of the segment for construction of the query.

TextRank. TextRank [21] is a graph based extraction method where documents are modeled with weighting co-occurrence networks using a co-occurrence window of variable sizes (2–10). $TR(V_i)$ represents the TextRank score of the point V_i calculated as shown in Eq. 3.

$$TR(V_i) = (1 - d) + d \sum_{V_j \in In(V_i)} \frac{W_{ji}}{\sum_{V_k \in Out(V_j)} W_{jk}} TR(V_j) \quad (3)$$

Similar to the concept of PageRank [22], d is the probability of the phrase occurring at random in the document, and is between 0 and 1.

4.3 Key Phrase Generation

Key hrase generation differs from key phrase extraction. The former are models trained to learn the mapping between a pair of texts and generate new key phrase text, while the latter extracts the most relevant words from a given input.

BART [24] & T5 [16] are two of the most successful generation transformers developed to date. They can be fine tuned for text-to-text-generation problems such as our task of key phrase generation.

[4] https://github.com/LIAAD/yake.

KeyBART. A generative model for text generation that reproduces key phrases with connection to the input in the CatSeq format. The internal architecture of BART is built on a transformer encoder-decoder (seq2seq) model with a bidirectional encoder, similar to BERT and an autoregressive decoder. The pre-training of BART takes two stages: 1) corrupting text with an arbitrary noising function, and 2) learning a model to reconstruct the original text.

T5-small-OpenK. This model is a key phrase generation technique based on the T5-small model and fine-tuned using the dataset OpenKP. This generation transformer model is tuned as text to text to generate keywords, with the limitation of only working on documents using the English Language. This model has approximately 220 million parameters, which is approximately twice the number of parameters as BERT. The model was pre-trained on a different mixture of supervised and unsupervised tasks.

5 Webpage Retrieval

Once the query has been constructed for a podcast segment, the next stage is to use it to search for relevant webpages. For this task we make use of the Google Custom Search Engine API [23]. This is a restful API that retrieves results of a search query in a JSON object, with three types of data: search results, metadata containing information about the requested search and metadata containing information about Programmable Search Engine. Every search retrieves a maximum of 10 results per query. We extracted the search ranking, URL link and title of the page obtained for each query.

It should be noted that multiple calls to the Google API are not guaranteed to return the same documents. However, since the calls are made in quick succession, the features of the search engine are unlikely to change between calls.

6 Evaluating of Our Information Retrieval System

In this section we describe our method for assessing relevance of retrieved items, and the evaluation metrics used in our investigation.

6.1 Relevance Evaluation

To assess the relevance of webpages we used crowdsourcing with Amazon Mechanical Turk (MTurk)[5]. This service provides a platform providing access to online human workers how can complete assigned tasks referred to as Human Intelligence Tasks (HITs). Online workers recieve payment for successfully completed HITs. For our assessment of linking using our query generation methods, we form a pool of retrieved items for relevance assessment for each podcast segment in our test dataset. The pool was formed by selecting the top 3 ranked

[5] https://mturk.com/.

documents retrieved for each query, merging these into a pool of unique document entries, and then requesting MTurk workers to assess the relevance of each retrieved document in the pool. Relevance of each item was assessed in terms of its relation to the segment used to create the queries which retrieved the item. The worker was shown the transcript and the contents of each retrieved item in the pool one after the other. Workers were required to perform the assessment on a graded scale (Excellent(3), Good (2), Fair (1), Bad (0)).

Since workers may not carry out the tasks correctly, we implemented a number of quality control measures. We estimate that each HIT should take 3–5 min, therefore any HITs completed in under 45 s were rejected without payment, and the HIT re-submitted to MTurk. Similarly, we rejected workers who returned an identical answer every time. Additionally, in order to help ensure that we recruited reliable workers, we required that workers had an approval rating of at least 80% for their previous completed HITs on MTurk.

Since relevance judgements are subjective, we gathered more than one judgement of each podcast-segment webpage pair to improve reliability of the relevance scores. To do this, we assigned each HIT to three AMT workers, meaning that 3 rounds of assessments are recorded for each HIT. We took the average of the three judgements as the agreed relevance level for each assessed document.

6.2 Evaluation Metrics

Information retrieval is typically evaluated based on the top k documents returned for a query. We use three metrics to evaluate the effectiveness of our content linking retrieval method [25,26]:

P@k: Precision at K. Quantifies how many items in the top-k results are relevant, i.e. out of the returned list how many were relevant. TP is True Positives and FP is False Positives:

$$Precision@k = \frac{TP@k}{(TP@k) + (FP@k)} \tag{4}$$

MAP: Mean Average Precision. Calculates the average precision across multiple queries, with Q being the number of queries and AP(q) the average precision for query q:

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q) \tag{5}$$

MRR: Mean Reciprocal Rank. This is a measure of the rank position of the highest ranked relevant document, i.e. how far down the list we have to go to find a relevant document, with Q being the total number of queries, and rank_i the rank of the first relevant result:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{6}$$

Table 2. Results - relaxed relevance

Method	P@1	P@3	MAP	MRR	nDCG
M1	0.767	0.644	0.811	0.800	0.967
M2	0.767	0.744	0.811	0.797	0.948
M3	0.733	0.744	0.822	0.822	0.922
M4	0.700	0.700	0.789	0.789	0.930
M5	0.800	0.789	0.856	0.847	0.923
M6	0.7000	0.711	0.767	0.769	0.950
M7	0.800	0.711	0.844	0.840	0.950
M8	0.833	0.822	0.900	0.886	0.942
M9	**0.933**	**0.911**	**0.967**	**0.953**	0.953
M10	0.800	0.844	0.900	0.892	0.917

Table 3. Results - stricter relevance

Method	P@1	P@3	MAP	MRR	nDCG
M1	0.367	0.333	0.417	0.422	0.967
M2	0.367	0.378	0.422	0.400	0.948
M3	0.467	0.467	0.561	0.533	0.922
M4	0.333	0.411	0.467	0.469	0.930
M5	0.367	0.478	0.533	0.542	0.923
M6	0.367	0.378	0.472	0.458	0.9500
M7	0.333	0.356	0.428	0.422	0.950
M8	0.367	0.378	0.472	0.458	0.9500
M9	**0.700**	**0.678**	**0.778**	**0.761**	0.953
M10	0.467	0.556	0.611	0.603	0.917

nDCG: Discounted Cumulative Gain. This is a sum of relevance scores for the top-n documents, normalized by penalizing each score by its position, i.e. it gives a weight to relevant documents in order of ranking, with rel(n) being an indicator function which is 1 when the item ar rank K is relevant

$$nDCG@k = \sum_{i=1}^{n} \frac{rel_i}{log_2(i+1)} \tag{7}$$

7 Experimental Results

In this section, we present results using two different relevance levels. In the first relaxed setting, webpages assessed with relevance levels 1, 2 and 3 are all counted as relevant. In the second strciter setting, only documents assessed with relevance levels 3 & 4 are counted as relevant.

Setting 1. From Table 2, we see that for M9 at P@1 around 93% of the first retrieved webpages are relevant. M9 also obtained the best results for MAP and MRR showing that the relevant items are also found at higher positions. In relation to nDCG, the values are similar across all the methods, revealing that all have a similar pattern for the ideal order on the relevant retrieved webpages. On the other hand, M4 and M6 obtain the lowest results with a P@1 of 0.7, meaning that only 70% of the first retrieved webpages are relevant, with a value of MAP below 0.8.

Setting 2. Table 3 shows results for Setting 2, we can see that M9 is again the best performing method. The values across the different metrics are lower than Table 2, as would expected, but M9 is still able to perform relatively well compared with the other methodologies, being the only method to achieve values above 0.6 for P@1, P@3, MAP and MRR. On the other hand, M4 and M7 have the lowest results, with only around 33% of the first retrieved webpages being

relevant. It is worth noting that when analysing some queries, it is possible to verify that some techniques using summarization do not capture key ideas in the summary, and consequently cannot generate accurate queries, while the best performing method was applied directly to the raw podcast segment.

8 Conclusions and Further Work

This paper described an investigation into the automated linking of related webpages to the transcripts of 2 min segments of podcasts. We examined 10 methods of creating queries from the podcast segment transcripts, and evaluated their effectiveness for retrieving related webpages using the Google API.

In general terms, we demonstrated that it is possible to link relevant content to different podcast segments, and that applying a key phrase extraction technique directly on the raw segment obtained better results than summarizing the already short 2 min segment. For both relevance settings reported, the best performing method was able to retrieve more than 70% of relevant content in the first ranked webpage.

There are various directions for potential further work. An important area for expansion is to increase the depth of the pool of retrieved documents assessed from the top 3, to the top 10 document or potentially more. Also rather than relying on fixed 2 min length segments, automatic segmentation methods could be applied to form topically related segments, which may form the basis of better queries since they provide full coverage of the topical region in the podcast.

Acknowledgement. The contribution of Gareth Jones is partially supported by Science Foundation Ireland as part of the ADAPT Centre (Grant 13/RC/2106) at Dublin City University.

References

1. Siddiqi, S., Sharan, A.: Keyword and keyphrase extraction techniques: a literature review. Int. J. Comput. Appl. **109**(2), 18–23 (2015)
2. Alonso, O., Rose, D.E., Stewart, B.: Crowdsourcing for relevance evaluation. In: ACM SigIR Forum, vol. 42, no. 2, pp. 9–15 (2008)
3. Paolacci, G., Chandler, J., Ipeirotis, P.G.: Running experiments on amazon mechanical turk. Judgm. Decis. Mak. **5**(5), 411–419 (2010)
4. Clifton, A., et al.: 100,000 podcasts: a spoken English document corpus. In: Proceedings of the 28th International Conference on Computational Linguistics (2020)
5. Jones, R., et al.: TREC 2020 podcasts track overview. In: Proceedings of TREC 2020, NIST, Online (2020)
6. Karlgren, J., et al.: TREC 2021 podcasts track overview. In: Proceedings of TREC 2021, NIST, Online (2021)
7. Nenkova, A., McKeown, K.: Automatic summarization. Found. Trends Inf. Retr. **5**(2–3), 103–233 (2011)
8. Maynez, J., Narayan, S., Bohnet, B., McDonald, R.: On faithfulness and factuality in abstractive summarization. arXiv preprint arXiv:2005.00661 (2020)

9. Ganguly, D., Pal, D., Verma, M., Sen, P.: Overview of RCD-2020, the FIRE-2020 track on retrieval from conversational dialogues. In: Proceedings of FIRE 2020, Online (2020)
10. Kaushik, A., Ramachandra, V.B., Jones, G.J.F.: DCU at the FIRE 2020 retrieval from conversational dialogues (RCD) task. In: FIRE, pp. 788–805 (2020)
11. Tang, L.-X., Geva, S., Trotman, A., Xu, Y., Itakura, K.Y.: Overview of the NTCIR-9 crosslink task: cross-lingual link discovery. In: Proceedings of the NTCIR-9 Workshop (2011)
12. Buckley, C., Voorhees, E.M.: Evaluating evaluation measure stability. ACM SIGIR Forum **51**(2), 235–242 (2017)
13. El-Kassas, W.S., Salama, C.R., Rafea, A.A., Mohamed, H.K.: Automatic text summarization: a comprehensive survey. Expert Syst. Appl. **165**, 113679 (2021)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
15. Miller, D.: Leveraging BERT for extractive text summarization on lectures. arXiv preprint arXiv:1906.04165 (2019)
16. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
17. Piskorski, J., Stefanovitch, N., Jacquet, G., Podavini, A.: Exploring linguistically-lightweight keyword extraction techniques for indexing news articles in a multilingual set-up. In: Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation, pp. 35–44 (2021)
18. Papagiannopoulou, E., Tsoumakas, G.: A review of keyphrase extraction. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **10**(2), 1339 (2020)
19. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1262–1273 (2014)
20. Campos, R., et al.: YAKE! Keyword extraction from single documents using multiple local features. Inf. Sci. **509**, 257–289 (2020)
21. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–411 (2004)
22. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1–7), 107–117 (1998)
23. Allauddin, M., Azam, F.: Service crawling using Google custom search API. Int. J. Comput. Appl. **34**(7), 2011 (2011)
24. Lewis, M., et al.: Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
25. Sakai, T., Kando, N.: On information retrieval metrics designed for evaluation with incomplete relevance assessments. Inf. Retr. **11**(5), 447–470 (2008)
26. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. (TOIS) **20**(4), 422–446 (2002)