DCU at the FIRE 2020 Retrieval from Conversational Dialogues (RCD) task

Abhishek Kaushik, Vishal Bhat Ramachandra and Gareth J. F. Jones

ADAPT Centre, School of Computing, Dublin City University, Dublin 9, Ireland

Abstract

The Retrieval from Conversational Dialogues (RCD) track at FIRE 2020 focused on two objectives: to contextualize the information in movie dialogues to identify the main topic of discussion, and to retrieve relevant information to queries focused on these key topics extracted from a transcript of the movie dialogues. We describe details of the ADAPT Centre's participation in the RCD track. We used a combination of techniques to identify the key topics of dialogues and the standard BM25 algorithm to retrieve potentially relevant paragraphs from an indexed Wikipedia archive. In both of the tasks, we used 5 different methods. In task one, our model, F8_4_model4 method performed better than other models with weighted bleu score 0.1090. In task 2, F7_4_model3 model outperformed other models with precision at 5 (P@5) score 0.0417 and Mean reciprocal rank (MRR) score 0.1086.

Keywords

topic identification in dialogues, dialogue linking, automated query construction

1. Introduction

The full meaning of the contents of dialogues is not always apparent to listeners, particularly those not familiar with information related to the details discussed in these conversations. The Retrieval from Conversational Dialogues(RCD) track at FIRE 2020 seeks to develop methods to identify topics within conversational spoken dialogues within movies and to retrieve relevant information relating to these topics from a Wikipedia dataset [1].

This paper overviews existing benchmark tasks related to the RCD track, and describes details of our participation in Task 1 and Task 2 of this track.

2. Related work

While the RCD task itself is new, it is related to a number of existing benchmark tasks. In this section we review the details of these related activities.

2.1. TREC 2019 Conversational Assistance Track

Conversational search has been a topic growing research interest recent years, and in 2019 TREC introduced the Conversational Assistance Track (CAsT) which seeks to develop a stan-

FIRE 2020: 12th meeting of Forum for Information Retrieval Evaluation, 16th-20th December 2020, Hyderabad, India.
abhishek.kaushik2@mail.dcu.ie (A. Kaushik); vishal.bhatramachandra2@mail.dcu.ie (V. B. Ramachandra); Gareth.Jones@dcu.ie (G. J. F. Jones)

dard benchmark for the evaluation of conversational search methods. In this track, the main objective is to satisfy the user's information need by understanding a sequence of questions in a conversational format and retrieving relevant documents associated with contextualized query at each turn in a conversation [2, 3]. Participant submissions focused mainly on three aspects:

- Query Understanding: Appropriate interpretation of the queries across the conversations is particularly challenging. The most popular query understanding technique was deep learning, with 57 percent of runs using it. NLP toolkits were used in half of the submitted runs, but there was no improvement in performance observed.
- Retrieval and Ranking: Using any training data led to improvements in the score. The unsupervised learning approaches used in 43% of the runs were not effective.
- Conversational Context: Almost all runs utilized information of previous turns to interpret their context and attempt to improve retrieval effectiveness. The title of the conversational topic also played a crucial role in identifying context. Only a small number of runs involved using all the turns and metadata, since the long descriptions that were created became noisy and were hard to use effectively.

In the process of ranking, Bert-based neural models were found to perform the best for this task. However, in some situations, these was outperformed by traditional information retrieval systems. Neural re-ranking approaches were also shown to be effective.

2.2. CoQA: A Conversational Question Answering Challenge

In this challenge, a new dataset was introduced for building Conversational Question Answering (CoQA) systems [4]. This dataset was built for the evaluation of systems that have to understand a series of conversations and answer questions about this conversation. The key features of this data set were: firstly, answers were text-free; and secondly, follow-up questions were more human conversation. In the end, it was given a rational, which acted as a evidence or support to answer the questions. This data had 8,000 conversations from which 127k questions with answers were extracted from seven different domains. After experimentation, it was observed that increasing the size of the history used in answering the questions decreased the models' performance. However, when previous turn data was considered, there was an increase in the model's performance. This was observed to reach higher performance when two previous turn conversations are shown, implying that most questions in a conversation have limited dependency within a bound of two turns. The Combination of Document Reader (DrQA) model [5] and PG-net model performed best [6].

2.3. QuAC : Question Answering in Context

This challenge was inspired by the student and teacher relationship, where a student poses a series of highly contextual questions, and the teacher answers these questions. The Question Answering in Context dataset contained 14k QA, which were crowdsourced (100k QA pairs in total). The questions used in this dataset were highly contextual, open-ended, and in some cases even unanswerable from the text [7]. While creating the dataset, the developers of the task simulated a teacher and student interaction where the student was shown the title of the topic and first paragraph of the topic. In contrast, the teacher had full access to the section text. There could be situations where the teacher did not have an answer to the question, which was marked as no answer. Taking into account context is very important in this challenge. BiDAF++ [8] performed best in this task. Adding more context information increased the model's performance. In this case, up to 3 previous question contexts were provided. The main takeaway from this challenge is that the model performs better when considering the previous turns or history of conversation to a certain degree.

2.4. TREC 2019 Deep Learning Track

This track focused on ad-hoc ranking in large datasets. It consisted of two tasks: document retrieval and passage retrieval. This track had the goal of studying ad-hoc ranking in a large data setting [9]. The end to end retrieval task involves 'phrase1' which involves a ranking task (by using BM25) and then in 'phrase2' re-ranking it by using the ML model. In many cases, a combination of traditional IR methods and deep learning techniques performed well.

The document retrieval task had two sub-tasks: Full retrieval and top 100 re-ranking. The same applied for the passage retrieval sub-task except that it had to re-rank 1000 passages. Both the tasks were judged on four criteria: perfectly relevant, highly relevant, related, and irrelevant.

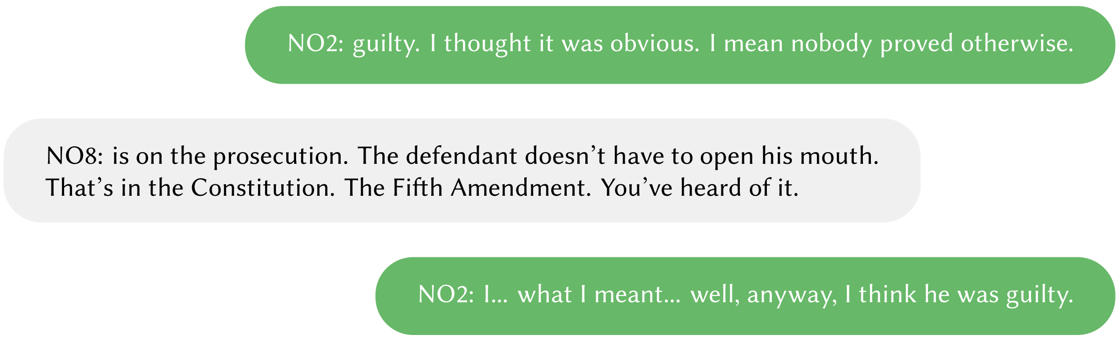
The metric used for evaluation was Normalized Discounted Cumulative Gain (NDCG), especially NDGC@10, which was used since it uses 4-level judgment. In both of the tasks, a model employing large scale pre-trained models such as BERT or XLNET, outperformed other models that used word embeddings or traditional information retrieval methods. We can infer that using a neural network model or a pre-trained model and other NLP techniques can increase a model's effectiveness. The submissions to this task helped us in task 2 of the FIRE 2020 RCD track challenge.

2.5. Know What You Don't Know: Unanswerable Questions for SQuAD

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset. This consists of questions posed by crowdworkers using a set of Wikipedia articles. The answer to the question is either a segment or span of text from a document or the question may be unanswerable from the available document set. The complexity of the SQuAD tasks have been gradually increased to the point where the current SQuAD 2.0 combines questions freely created by crowdworkers from the earlier SQuAD1.1 with unanswerable questions written deliberately to look similar to answerable ones [10]. To be able to work well with SQuAD 2.0, a question answering system must not only be able to answer questions, but also be determine when the answer is not available within the document set.

3. Task Description

In this section we give brief summaries of the details of the FIRE 2020 RCD tasks.


NO2: guilty. I thought it was obvious. I mean nobody proved otherwise.

NO8: is on the prosecution. The defendant doesn't have to open his mouth. That's in the Constitution. The Fifth Amendment. You've heard of it.

NO2: I... what I meant... well, anyway, I think he was guilty.

Figure 1: Conversation sample from training data

3.1. Task 1

The objective of the FIRE 2020 RCD Task 1 is to automatically contextualize two-party or multi-party dialogue systems, and then to use information retrieval methods to retrieve more information about the key-phrases identified as the most important in the conversation, which can be two-party or multi-party dialogues in nature.

Specifically this task focuses on conversations in movie dialogues. The conversations selected for the task are extended long exchanges between one or more actors. For example, from the movie "12 Angry Men": shown in Figure 1[1], the discussion topic is the "fifth amendment". Task 1 focuses on identifying these discussion topics of this type.

3.2. Task 2

Task 2 focuses on using the key topics identified from analysis of the conversations as the basis for queries to search for more information about these topics. Specifically, the task requires an indexed Wikipedia dump to be searched for documents in response to an automatically formed query and returned in ranked order.

4. Methodology

In this section we describe details of our methods for tackling these tasks.

4.1. Task 1

4.1.1. Approaches

For Task 1, we adopted two approaches. We examined the use of 5 different methods with parameter tuning. The details of the approaches are given below:

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

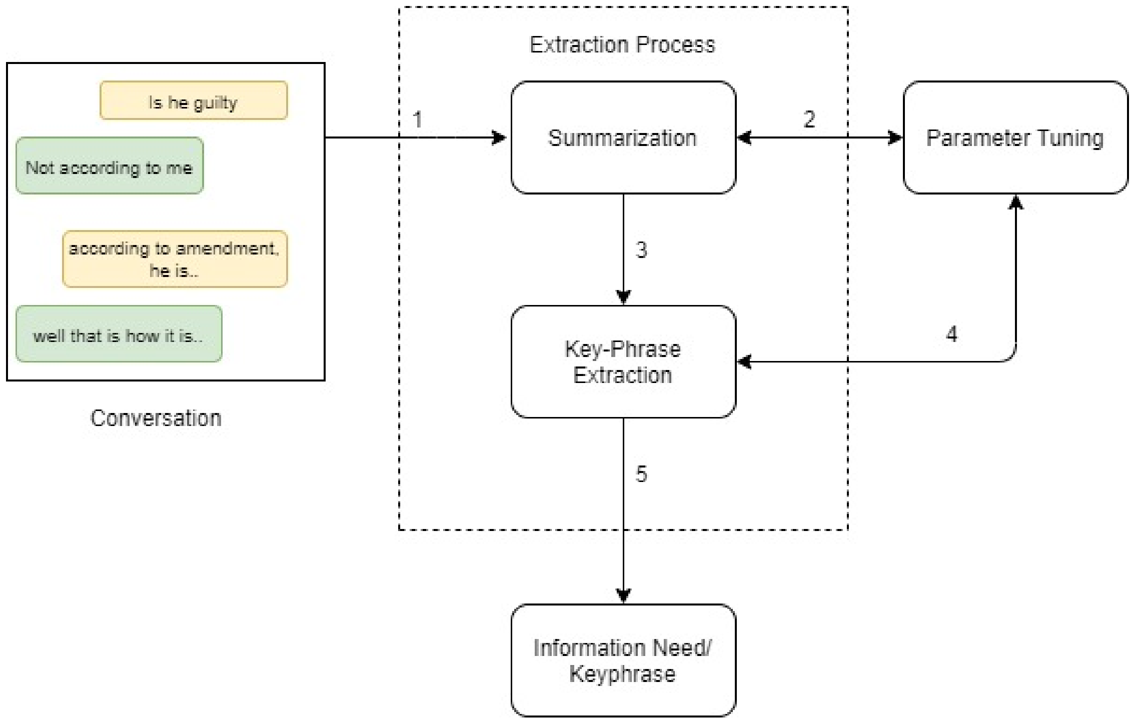


Figure 2: Flow chat of First Approach in Task 1

1. First Approach: As shown in Figure 2, the conversation was directly passed through summarization components which extracted important sentences from the conversation. These sentences were then passed through key-phrase extractor components. The summarization and key-phrase extractor can be tuned by using different parameters, described in detail in Section 1. The final output was taken to be the information need.
2. Second Approach: As shown in Figure 3, the conversation was directly passed through key phrase extractor components. The key-phrases were then filtered based on parameter tuning, which were taken to be the information need.

4.1.2. Parameter Tuning

We divided the parameter tuning methods into two categories: External parameters and Internal parameters. The best five configurations were selected after evaluation based on Jaccard distance on train (comparing the gold query and predicted query).

1. External Parameters: These parameters (as shown in Table 1) effect the output generated by the algorithms. For example, changing the ratio of summary in any summarizing methods to extracted important sentence from complete conversation about a particular information need. The external parameters are as follows:

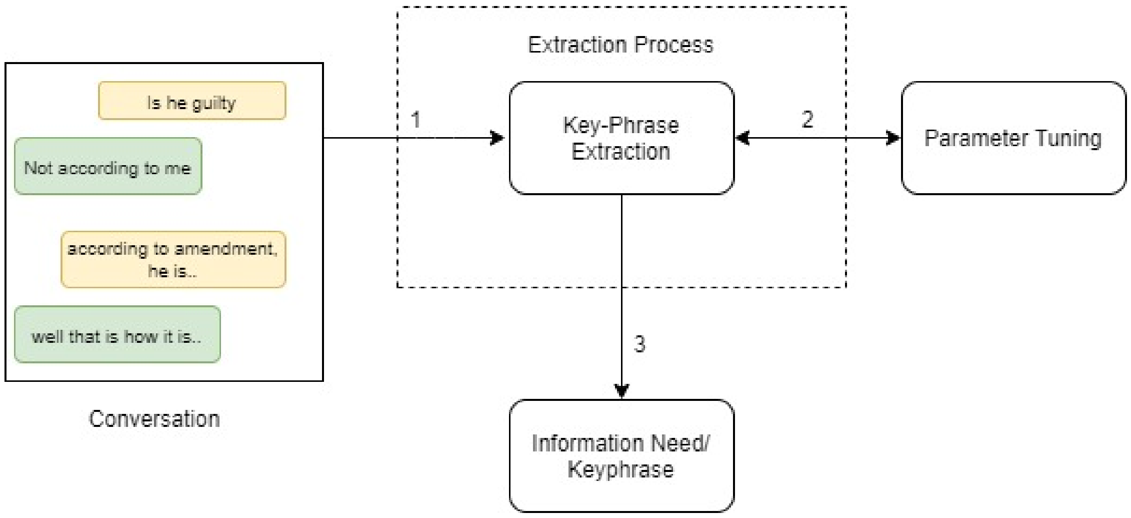
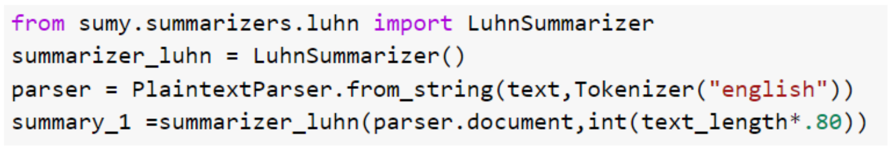


Figure 3: Flow chat of Second Approach in Task 1

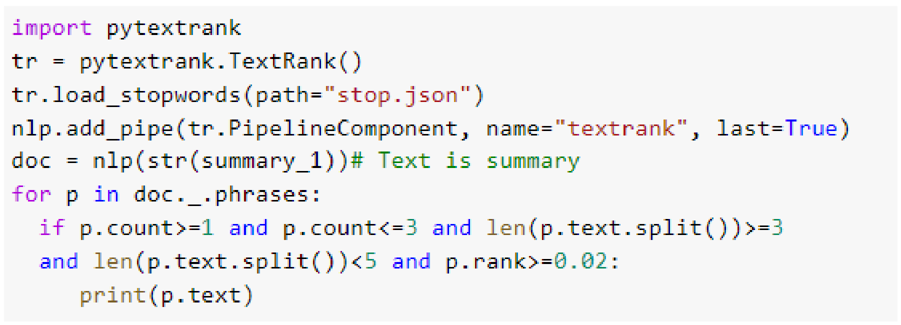
a) Word Length: This signifies the word count of the key-phrases extracted. In this study, the investigation observed that key-phrases which contained 3-5 words were best match key-words as per the information need. For an example, the extracted keyphrase is ”the fifth amendment” and the word length is 3. We observed that using queries of word length more than five words had increased Jaccard distance [11] from the extracted information need to the gold information need. The python tool kit was used to calculate the Jaccard distance [2].
b) Summary Percentage: This defines the ratio of the output (summary) produced by the summarizer in comparison to the input paragraph (conversation). This extracted output was passed through the key-phrase extractor as shown in Figure 2. We also varied the summary ratio between 50% to 80% to understand its effect on the key-phrase extractor. We observed that the performance of the key-phrase extractor in extracting the information need varies as per the summary ratio. In our investigation, it was found that a summary ratio of 70% gave the best result for the key-phrase extractor.
c) Phrase Frequency: This signifies how many times this particular phrase was repeated in a particular conversation. It was found that phrases which are repeated between 1-3 times in conversation are potential key-phrases of an information need. In the current study, we found that the key-phrases were not repeated more than 3 times in a conversation. Sometimes key-phrases are indicated by a pronoun which makes the process more complex and lengthy. In our similar previous work [3], we manually replaced all the pronouns with the corresponding key-word assuming that this would enable us to extract better statements of information need. However, this approach did not prove to be beneficial for extracting statements of information

[2]https://pypi.org/project/textdistance/



```
from sumy.summarizers.luhn import LuhnSummarizer
summarizer_luhn = LuhnSummarizer()
parser = PlaintextParser.from_string(text,Tokenizer("english"))
summary_1 =summarizer_luhn(parser.document,int(text_length*.80))
```

Figure 4: Code snippet example of summary extraction by Luhn method using sumy toolkit



```
import pytextrank
tr = pytextrank.TextRank()
tr.load_stopwords(path="stop.json")
nlp.add_pipe(tr.PipelineComponent, name="textrank", last=True)
doc = nlp(str(summary_1))# Text is summary
for p in doc._.phrases:
  if p.count>=1 and p.count<=3 and len(p.text.split())>=3
  and len(p.text.split())<5 and p.rank>=0.02:
     print(p.text)
```

Figure 5: Code snippet example of key-phrase extraction by TextRank method using pytextrank toolkit

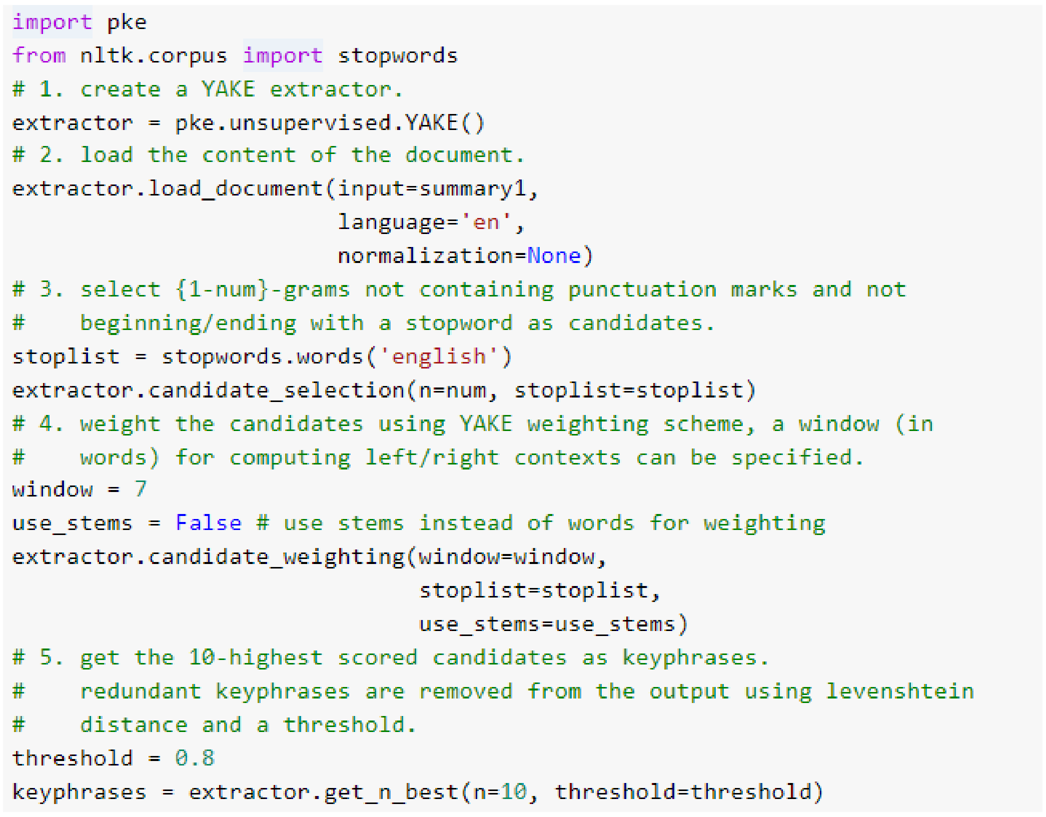
need and model performance was not improved.

d) Summary Extraction: This signifies the summary extraction methods used to extract important sentences from the conversation. In this study, we used four different summary extraction approaches: Bert (Bidirectional Encoder Representations from Transformers) [12], Custom (lab based) [13], Luhn [14] and Kullback-liber [15]. Details are discussed in the following Sections 4.1.3, 4.1.4, 4.1.5 and 4.1.6.

e) Phrase Extraction: In the current investigation, we used two important methods for phrase extraction: TextRank [16] used in methods 4.1.4, 4.1.5, 4.1.6, 4.1.7 and the Yake method [17, 18] used in method 4.1.3. Details are given in Sections 4.1.3 and 4.1.4.

2. Internal Parameters: These parameters affect the behaviour of the algorithms. For example, varying the configuration of the Yake method [17, 19] by changing its window size and threshold. Yake is a lightweight unsupervised automatic key-word extraction method which is based on statistical text features extraction from a document to select the most important key-phrases of a document.

   a) Window size (Yake): The size of the words used for computing co-occurrence counts.
   b) Threshold (Yake): Levenshtein distance, which helps in measuring the difference between key-words. A key-phrase is considered redundant if its distance from other key-phrases which ranked higher in the list is greater than a threshold. The default value is 0.8.
   c) N top scoring word (Yake): Top N key-phrases of the document (summary conversation) to be displayed based on the score.
   d) Rank Score (TextRank): We can use this as a threshold value to consider key-words with rank score higher than certain value.



```python
import pke
from nltk.corpus import stopwords
# 1. create a YAKE extractor.
extractor = pke.unsupervised.YAKE()
# 2. load the content of the document.
extractor.load_document(input=summary1,
                        language='en',
                        normalization=None)
# 3. select {1-num}-grams not containing punctuation marks and not
#    beginning/ending with a stopword as candidates.
stoplist = stopwords.words('english')
extractor.candidate_selection(n=num, stoplist=stoplist)
# 4. weight the candidates using YAKE weighting scheme, a window (in
#    words) for computing left/right contexts can be specified.
window = 7
use_stems = False # use stems instead of words for weighting
extractor.candidate_weighting(window=window,
                              stoplist=stoplist,
                              use_stems=use_stems)
# 5. get the 10-highest scored candidates as keyphrases.
#    redundant keyphrases are removed from the output using levenshtein
#    distance and a threshold.
threshold = 0.8
keyphrases = extractor.get_n_best(n=10, threshold=threshold)
```

Figure 6: Code snippet example of key-phrase extraction by Yake method using Pke toolkit

e) K1 and B value (BM25): K1 controls non-linear term frequency normalization and B controls to what degree document length normalizes tf values.

f) EPS and Minimum Samples in Density-based spatial clustering of applications with noise (DBSCAN): EPS is the minimum distance for two points to be considered neighbours and min_samples is the minimum number of samples required for a point to be considered a core point.

In this section, we described in outline of the five methods used to extract the information need and based on the parameter tuning as shown in the Table 1. The following are the details of our models:

4.1.3. Method 1

Y7_2_Model5_2: This model used the first approach discussed in section 4.1.1 and workflow shown in Figure 2. The detailed parameter configuration for this specific method is shown in Table 4.1.2. In this approach, we utilized a tool powered by the Hugging Face Pytorch transformers library to run extractive summarizations [12] which is available as a Python API[3],

[3]https://pypi.org/project/bert-extractive-summarizer/

Run	Word Length	Summary (%)	phrase Frequency	phrase Extraction	Summary Extraction
Y7_2_Model5_2	3-5	50	NA	Yake	Bert Based
F8_4_model4	3-5	70	1-3	Text rank	Custom
F7_4_model3	3-5	80	1-3	Text rank	Kullback-Lieber
F6_4_model2	3-5	80	1 -3	Text rank	Luhn
F5_0_Model1	3-4	NA	1-3	Text rank	NA

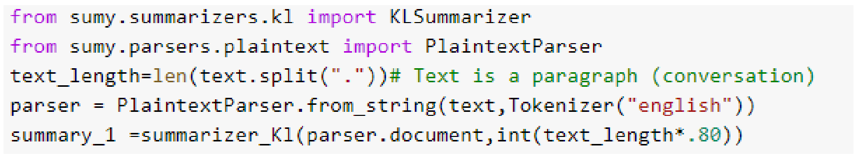
Table 1
Parameter Tuning in Models

running a k-means clustering algorithm and finding sentences that are closest to the cluster's centroids. After obtaining a summary from each paragraph, we extracted the essential phrases by using an unsupervised method known as Yake[4] [17]. This approach is based on a statistical text feature extraction method to select the most relevant Key phrases. In this setting, we used different parameter tuning for the Yake version of PKE [18]. The best results for this model were obtaining by setting a window size of 7, which is used for capturing the context of he phrases, and threshold to 0.8, this helps to eliminate redundant queries using the concept of Levenshtein distance. As shown in figure 6, yake python code of key-phrase extraction from extracted summary. In this method, we extracted the top scoring (based on Yake top ten key-phrases) unique phrases whose word length varies between 3-5 words. Pke python tool kit provided an option to select a key-phrase candidate based on its word length with the hyper parameter n in the *extractor.candidate_selection* function. In this method, we have extracted 3 lists of top 10 key-words based on n = *3,4,5*. These top 10 key-phrases were extracted using the pke function *extractor.get_n_best*. From all 3 extracted lists, the top scoring key-phrase in the list was considered to be information need for the conversation.

4.1.4. Method 2

F8_4_model4: This model used the first approach discussed in section 4.1.1 and workflow shown in Figure 2. The detailed parameter configuration for this specific method is shown in Table 4.1.2. In this approach, we extracted important segments by using an algorithm (shown in pseudo code 1) based on the normalized TF-IDF score of each sentence. The best scoring 50% of the sentences were selected. These sentences were divided into clusters using a DB SCAN algorithm with eps (distance parameter) set to 0.1, min_sample (number of samples to be considered closest to core point) set to 2. A cosine similarity score was calculated between clusters and movie names, and the top-scoring 70% (summary percentage) of the clusters were selected to extract key-phrases using the textRank algorithm [20]. All extracted clusters were combined together to form a summary. We utilized the Pytext rank tool powered by the Spacy and TextRank [5] algorithms to extract the essential phrases by constructing a lemma graph to represent links among the candidate phrases [16] from the summary. As shown in Figure 5, the extracted summary was passed through the TextRank algorithm, by setting up the filters. As per the filters, only those key-

[4]https://github.com/boudinfl/pke
[5]https://pypi.org/project/pytextrank/



```
from sumy.summarizers.kl import KLSummarizer
from sumy.parsers.plaintext import PlaintextParser
text_length=len(text.split("."))# Text is a paragraph (conversation)
parser = PlaintextParser.from_string(text,Tokenizer("english"))
summary_1 =summarizer_Kl(parser.document,int(text_length*.80))
```

Figure 7: Code snippet example of summary extraction by Kullback-Lieber (KL) divergence method method using sumy toolkit

phrases were considered whose word count lay between 3-5 words. The key-phrase frequency should not exceed more than 3 times. The key-phrases were selected whose *p.rank score* is greater than 0.2. Only key-phrases which met these conditions were considered as the information need. This model was the top scorer in the Task 1 with the weighted Bleu Score 0.1090.

Algorithm 1: Pseudo code Custom Summarization Method

```
Result: Custom Summarization Algorithm
    1. Split the whole Paragraph into sentences
    2. Each sentence is consider as individual document
    3. Calculation of Term frequency of each word of document
    4. Calculation of Inverse Document frequency of each word
    5. Calculation of TF-IDF score of each word
    6. Calculate the normalized TF-IDF score of each sentence
    7. Top 50% Top scoring sentence extracted
    8. DBScan clustering algorithm is applied on extracted sentence with parameters eps = 0.1
       and minimum sample =2
    9. DBScan divided the sentences into $n$ cluster
   10. Cosine similarity score has been calculated between the $n$ number of clusters and movie
       name
   11. Based on the cosine similarity score top 70% out of $n$ cluster has been extracted for key
       phrase extraction
```

4.1.5. Method 3

F7_4_model3: This model used the first approach discussed in section 4.1.1 and workflow shown in Figure 2. The detailed parameter configuration for this specific method is shown in Table 4.1.2. In this approach, we utilized the Sumy tool [6] to generate an extractive summary using a Kullback-Lieber (KL) divergence method for summarization [15]. The KL method greedily adds sentences to a summary so long as it decreases the KL Divergence. As discussed earlier, each conversation was treated as a document.

Similarly to method 2, we used the Pytext rank method [16] to extract the essential key-phrases from the summary. In this approach, we used 80% of the summary setting, in which the summarizer retains 80% of the original content using KL divergence method as shown in

[6]https://pypi.org/project/sumy/

Figure 7.. The extracted summary passed through the TextRank algorithm as shown in Figure 5, by setting up the filters. As per the filters, only those key-phrases were considered whose word count lay between 3-5 words. The key-phrase frequency should not exceed more than 3 times.The key-phrases were selected whose *p.rank score* was greater than 0.2. Only key-phrases which met these conditions were considered as the information need.

4.1.6. Method 4

F6_4_model2: This model used the first approach discussed in section 4.1.1 and workflow shown in Figure 2. In this approach, we extracted a summary using a method based on the work of Luhn [21] as shown in Figure 4. Similar to method 3, we used the Pytext rank method [16] to extract the essential key-phrases from the summary. To extract the summary by Luhn method, we used open source python tool kit [7] by following the parameters shown in Table 1. Then the extracted summary was used as input to pytext key phrase extractor which extracted the important keywords phrase. Three important parameters of PageRank were considered to select the key-phrases: RageRank score, word length and phrase frequency as shown in Table 4.1.2. Key-phrases whose *p.rank score* greater than 0.2, word length was between 3-5, and considered key-phrase whose frequency lies between 1-3.

4.1.7. Method 5

F5_0_Model1: This model used the second approach discussed in section 4.1.1 and workflow shown in Figure 3. In this approach, we utilized the Pytext toolkit [16] to extract critical key-phrases as shown in Figure 5. In this model, different approach has been adapted to investigate the potential of text rank algorithm in extracting the important key-phrases. Without using any summarizations, we directly pass the complete conversation paragraph into the pytext toolkit and extracted the key-phrase and later on these key-phrases were filtered out based on the parameter tuning as shown in Table 4.1.2. The key-phrases were selected whose *p.rank score* was greater than 0.2. The selected key-phrases were further filtered by the phrase frequency which should not exceed than 3 times. The filtered key-phrases were then further netted were selected based on word length in the key-phrase. Key-phrases were finally only selected as the information need if their word length lay between 3-4 words in the key-phrase. This model is among the top models in Task 2 with the MAP (Mean Average Precision) 0.0021.

4.2. Task 2

In this section our approach to tackle Task 2. Task 2 is to retrieve relevant information to queries focused on these key topics extracted from a transcript of the movie dialogues.
For this task, we used python open source tool kit for the Pyterrier [22] to index the Wikipedia dataset to be searched. The Wikipedia dump consists of approximately 73 GB data. This was first cleaned before extracting pairs of paragraph ids and their associated text.

These paragraphs (document units) were then indexed into Terrier [8] [22]. For ranking in our experiments, we used the standard BM25 algorithm, with k1 set to default value (1.2) and

[7]https://pypi.org/project/sumy/

[8]https://github.com/terrier-org/pyterrier

the b also set to default value (0.75) in the Terrier system. The queries extracted from the conversations in Task 1 were then entered into Terrier to search the indexed dataset.

5. Results

In this section, we present the results of our experiments performed in the training and test phase.

5.1. Training Task

1. Task 1 Results: Tables 2 and 3 present training and test results of Task 1 respectively calculated for the Jaccard Coefficient for keyword similarity [20] and Weighted Bleu score [23] to calculate the similarity score between the predicted and gold query. In this track, we choose the best models based on our training data performance.
   a) Train: From Table 1, we can see that model F5_0_Model1 performs the best, and the next best performing model is Y7_2_Model5_2 for the training data.

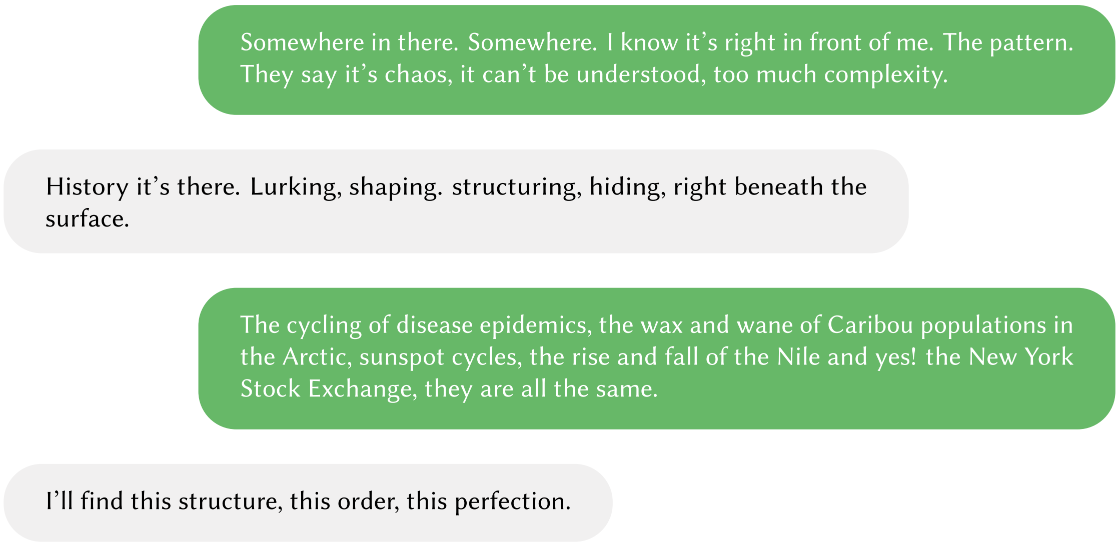
   b) Test: From Table 3, we can see that the F8_4_model4 model performs best, and F5_0_Model1 is the next best performing for the test data. We can conclude that the weighted score for test data is better than for the training data, which can also be seen in Figure 11.

Run	Weighted Bleu score	Word-level Jaccard Overlap
Y7_2_Model5_2	0.1660	0.1333
F8_4_model4	0.1362	0.0958
F7_4_model3	0.1502	0.1333
F6_4_model2	0.1502	0.1333
F5_0_Model1	0.2122	0.1625

Table 2
Results of Task 1 for train data

Run Name	Weighted BLEU Score	Word-level Jaccard Overlap
F6_4_model2	0.0636	0.0487
F7_4_model3	0.0989	0.0727
F5_0_model1	0.0984	0.0487
Y7_2_model5_2	0.0020	0.0000
F8_4_model4	0.1090	0.0636

Table 3
Results of Task 1 for test data


Somewhere in there. Somewhere. I know it's right in front of me. The pattern. They say it's chaos, it can't be understood, too much complexity.

History it's there. Lurking, shaping. structuring, hiding, right beneath the surface.

The cycling of disease epidemics, the wax and wane of Caribou populations in the Arctic, sunspot cycles, the rise and fall of the Nile and yes! the New York Stock Exchange, they are all the same.

I'll find this structure, this order, this perfection.

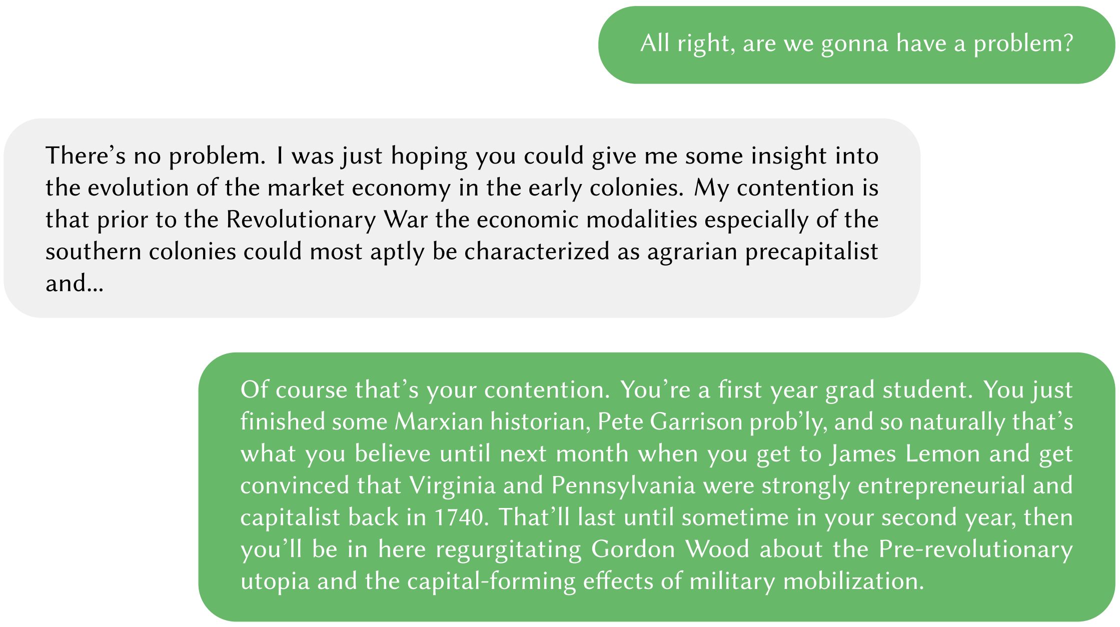
Figure 8: Conversation sample from test data

5.1.1. Analysis of the F8_4_model4 best performing model for the test data

The conversation shown in Figure 8 is part of a dialogue within the test data. When trying to extract a potential query from the conversation, F8_4_model4 (our best performing model) predicts "the New York Stock Exchange" as the output, which matches with the expected result.

The conversation shown in Figure 9 is taken from the test data, in this case the F8_4_model4 model predicted "the second year" as the output, whereas the actual expected output is "southern colonies." We can tell from examining the conversation that our approaches fail when there are multiple discussion topics in a single turn. It is a challenge to identify a single topic of discussion when we have numerous. Transforming information need into the right query to retrieve the most relevant results is a highly complex process which includes a huge amount of cognitive efforts for the users [24]. Generally, human conversation is unstructured which makes it difficult to extract the right key-phrase which can represent itself as a perfect query to retrieve to relevant document. In our approach, we consider a set of conversation as text document and pass through the summarizer to get the most important sentence which further used to extract important key-phrase using some statistical methods. This approach may loss some important information which could be vital to form the right query. To minimize the loss, we extracted up to 80% of the total sentence by passing through summarizer functions and also keep the strict the knob on the key phrase repetition in conversation and the length of the key-phrase including internal parameters of summarizer and key-phrase extractor.

1. Task 2 Results: For this task, we use three metrics to evaluate the effectiveness of our retrieval system based on our extracted queries.
   a) MAP: Mean Average Precision (MAP)
   b) P@5: Precision at 5.
   c) MRR: Mean Reciprocal Rank (MRR).


All right, are we gonna have a problem?

There's no problem. I was just hoping you could give me some insight into the evolution of the market economy in the early colonies. My contention is that prior to the Revolutionary War the economic modalities especially of the southern colonies could most aptly be characterized as agrarian precapitalist and...

Of course that's your contention. You're a first year grad student. You just finished some Marxian historian, Pete Garrison prob'ly, and so naturally that's what you believe until next month when you get to James Lemon and get convinced that Virginia and Pennsylvania were strongly entrepreneurial and capitalist back in 1740. That'll last until sometime in your second year, then you'll be in here regurgitating Gordon Wood about the Pre-revolutionary utopia and the capital-forming effects of military mobilization.

Figure 9: Conversation sample from test data

Run Name	MAP	P@5	MRR
F5_0_Model1	0.0021	0.0400	0.0922
Y7_2_Model5_2	0.0001	0.0160	0.0023
F7_4_model3	0.0016	0.0417	0.1086
F6_4_model2	0.0013	0.0160	0.0518
F8_4_model4	0.0003	0.0400	0.0704

Table 4
Results of Task 2 test data

From Table 3, we can see that retrieved results for the F5_0_Model1 are the best in terms of MAP score and the F7_4_model3 in terms of MRR and P@5 scores. Looking at Figure 10, we can conclude that the F7_4_model3 performs best and Y7_2_model5_2 performs the worst.

6. Discussion

While it is difficult to contextualize a long conversation, we can broadly look into the techniques used to identify the topic of discussion. The target contextual word feature, such as the length, differs from dialogue to dialogue. There are situations where there is more than one important topic of discussion, where it is will be extremely hard to choose any one among them as the most important. We had to take into consideration, the key phrase word length and frequency of key phrase in overall conversation. In our investigation, we have found that the Key-phrase seems

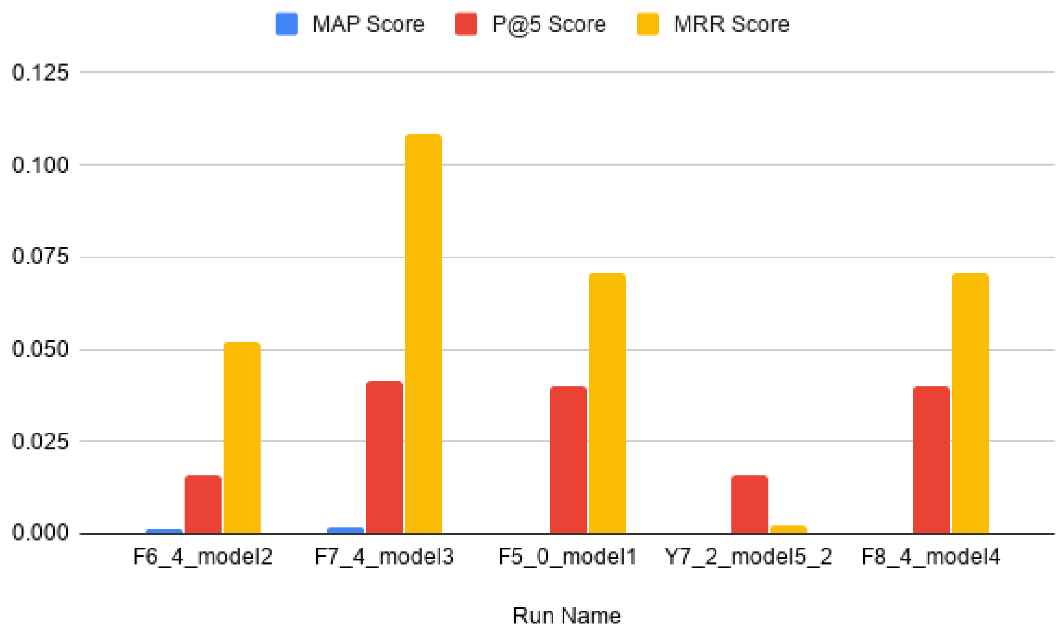


Figure 10: Task 2 score comparison

to be important whose frequency lies between 1-3 and the word length lies between 3-5 words. Both the factors are crucial to extract information need. The dataset used for the task is too small to train a neural network model, which would potentially have increased the prediction capabilities for Task 1. Instead, we examined the use a pre-trained model, an unsupervised learning approach, and summarization techniques. In additions, we also participated in the study discussed in section 2 [3] where we extracted the entities and POS (Part of speech) from the conversation to extract the information need. Here in contrast, we used pre-trained model with a machine learning approach to extract the information need. Both the studies concluded that it is the complex process to extract the information need from conversation process and may require huge amount of data to verify the use of current state of art such as deep learning. Certain studies in task [2] showed the potential of using deep learning depending upon the size of data available.

7. Conclusions and Further Work

In this report, we described the methods used for our participation in the FIRE 2020 RCD tasks. Our results show that our approches WERE among the average of submissions. In further work we plan to investigate the behaviour of our approaches and identify how they might be improved. If more data becomes available, we plan to explore the use of deep learning techniques to understand the conversation's context better. FOR Task 2, we would like to examine the

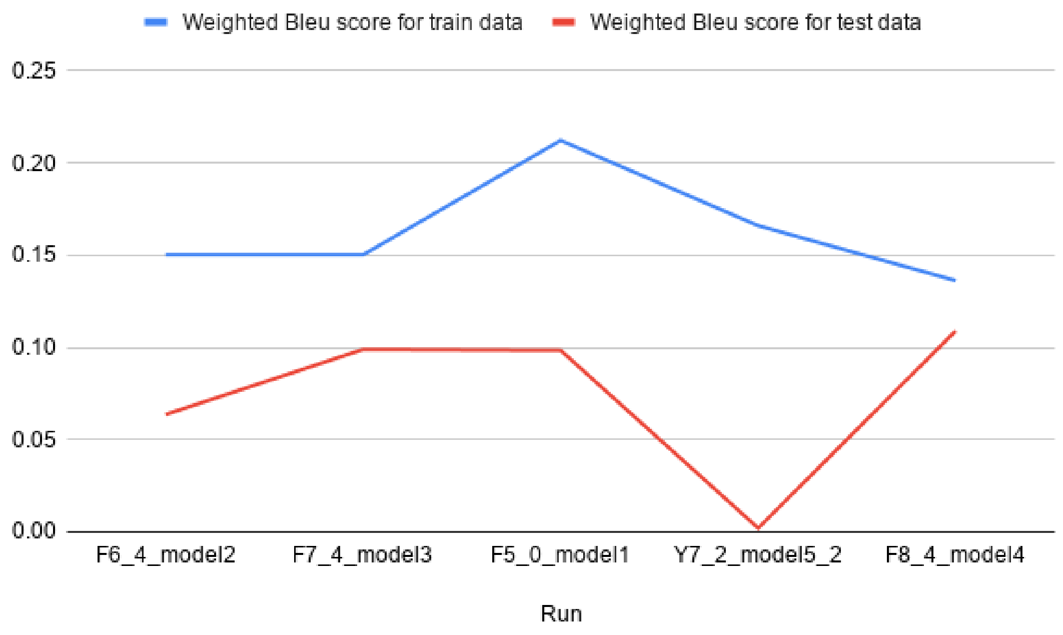


Figure 11: Task 1 bleu score comparison

application OF query re-formulation or query expansion techniques to improve information retrieval effectiveness [25, 26] or the use of neural network approaches to information retrieval [27].

8. Acknowledgments

This work was supported by Science Foundation Ireland as part of the ADAPT Centre (Grant 13/RC/2106) at Dublin City University.

References

[1] D. Ganguly, D. Pal, M. Verma, P. Sen, Overview of rcd-2020, the fire-2020 track on retrieval from conversational dialogues, in: Forum for Information Retrieval Evaluation (FIRE 2020), 2020.

[2] J. Dalton, C. Xiong, J. Callan, Trec cast 2019: The conversational assistance track overview, arXiv preprint arXiv:2003.13624 (2020).

[3] P. Arora, A. Kaushik, G. J. F. Jones, DCU at the TREC 2019 conversational assistance track, in: E. M. Voorhees, A. Ellis (Eds.), Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019, volume

1250 of *NIST Special Publication*, National Institute of Standards and Technology (NIST), 2019. URL: https://trec.nist.gov/pubs/trec28/papers/ADAPT-DCU.C.pdf.

[4] S. Reddy, D. Chen, C. D. Manning, Coqa: A conversational question answering challenge, Transactions of the Association for Computational Linguistics 7 (2019) 249–266.

[5] T. Chen, B. Van Durme, Discriminative information retrieval for question answering sentence selection, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 719–725. URL: https://www.aclweb.org/anthology/E17-2114.

[6] A. See, P. J. Liu, C. D. Manning, Get to the point: Summarization with pointer-generator networks, 2017. arXiv:1704.04368.

[7] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, L. Zettlemoyer, Quac: Question answering in context, arXiv preprint arXiv:1808.07036 (2018).

[8] M. Seo, A. Kembhavi, A. Farhadi, H. Hajishirzi, Bidirectional attention flow for machine comprehension, 2018. arXiv:1611.01603.

[9] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, Overview of the trec 2019 deep learning track, arXiv preprint arXiv:2003.07820 (2020).

[10] P. Rajpurkar, R. Jia, P. Liang, Know what you don't know: Unanswerable questions for squad, arXiv preprint arXiv:1806.03822 (2018).

[11] P. Jaccard, Étude comparative de la distribution florale dans une portion des alpes et des jura, Bull Soc Vaudoise Sci Nat 37 (1901) 547–579.

[12] D. Miller, Leveraging bert for extractive text summarization on lectures, 2019. arXiv:1906.04165.

[13] A. Kaushik, V. Bhat Ramachandra, G. J. F. Jones, An interface for agent supported conversational search, in: Proceedings of the 2020 Conference on Human Information Interaction and Retrieval, CHIIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 452–456. URL: https://doi.org/10.1145/3343413.3377942. doi:10.1145/3343413.3377942.

[14] H. P. Luhn, The automatic creation of literature abstracts, IBM Journal of research and development 2 (1958) 159–165.

[15] A. Haghighi, L. Vanderwende, Exploring content models for multi-document summarization, in: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Boulder, Colorado, 2009, pp. 362–370. URL: https://www.aclweb.org/anthology/N09-1041.

[16] P. Nathan, Pytextrank, a python implementation of textrank for phrase extraction and summarization of text documents, https://github.com/DerwenAI/pytextrank/, 2016.

[17] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, A. Jatowt, Yake! collection-independent automatic keyword extractor, in: European Conference on Information Retrieval, Springer, 2018, pp. 806–810.

[18] F. Boudin, pke: an open source python-based keyphrase extraction toolkit, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, Osaka, Japan, 2016, pp. 69–73. URL: http://aclweb.org/anthology/C16-2015.

[19] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, A. Jatowt, Yake! keyword

extraction from single documents using multiple local features, Information Sciences 509 (2020) 257–289.

[20] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of jaccard coefficient for keywords similarity, 2013.

[21] H. P. Luhn, The automatic creation of literature abstracts, IBM Journal of Research and Development 2 (1958) 159–165. doi:10.1147/rd.22.0159.

[22] C. Macdonald, N. Tonellotto, Declarative experimentation in information retrieval using pyterrier, 2020.

[23] M. Dreyer, D. Marcu, HyTER: Meaning-equivalent semantics for translation evaluation, in: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Montréal, Canada, 2012, pp. 162–171. URL: https://www.aclweb.org/anthology/N12-1017.

[24] A. Kaushik, Dialogue-based information retrieval, in: European Conference on Information Retrieval, Springer, 2019, pp. 364–368.

[25] R. Nogueira, K. Cho, Task-oriented query reformulation with reinforcement learning, 2017. arXiv:1704.04572.

[26] Z. Zheng, K. Hui, B. He, X. Han, L. Sun, A. Yates, Bert-qe: Contextualized query expansion for document re-ranking, arXiv preprint arXiv:2009.07258 (2020).

[27] Z. A. Yilmaz, S. Wang, W. Yang, H. Zhang, J. Lin, Applying bert to document retrieval with birch, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, 2019, pp. 19–24.