SI 650 / EECS 549: Final Project

Kevin Lee (kjunwonl@umich.edu)
Shivika Bisen (sbisen@umich.edu)
Yashaswini Joshi (yjoshi@umich.edu)

1 Introduction

In the growing world of data, the world has seen a surplus of tools that are able to grab the most relevant information pertinent to a user's query. Google has been dominant as a search engine and accelerates the learning process for many people around the world. The reliance of such tools has transformed many industries in how they operate their day to day and what information their tools use to forecast information. As the top industries develop these tools and enhance their retrieval models, many industries still need to catch up on this development of technology. A specific industry in mind is the non-profit sector. The information that many UN agencies and NGOs push out are often in disarray and unorganized. This makes the task of retrieving relevant information for a specific topic near impossible unless the specific location for the report was known or a standardized search engine was implemented for this purpose. This is the problem that our retrieval project wants to focus on. We want to alleviate this pain by creating a search engine to accurately retrieve relevant information from a query for UN and NGO reports.

Our model uses a variety of pdfs from 3 agencies relevant to popular topics within the nonprofit sector. After extracting the content of these reports along with the urls that directly lead to the pdf, we will be creating an algorithm to correctly rank the relevancy of the report topic to the query of user input. Ultimately, this will take the user to the most related pdf to their interested field and alleviate the entropy this domain typically faces. This can lead to many advances in the nonprofit world by allowing these workers to get the necessary information for research and accelerate plans for implementation. The application of this tool can benefit committee meetings and final decisions on how work plans can be implemented in helping the communities they often work with.

There is a UN digital library for UN documents but there is no consolidated search engine for UN or NGO program reports. Our solution is innovative in the following ways. Firstly, our dataset has equity in representation of UN, Non Profit and Humanitarian aid organizations. For instance, given a query google search showed top relevant links mostly biased towards UN reports and had little or no NGO reports. Moreover it

retrieved media articles. Our dataset ensures reliable source/report from the organization. Secondly, our aim is to help NGO program managers and Policy makers to have a search engine to design the policies. Google search fails to do that because it is biased towards annual reports as top relevant. In our data corpus we have Program reports and not just Annual reports. Program reports are more informative on details on the designing of the program while Annual reports have marketing inclination. Finally, our search engine provides a BERT based summary for each retrieved report.

We ended up using a baseline model that utilizes word2vec and cosine similarity. We found that our model of BM25L retrieved more accurate results than the word2vec. We tried many models like a deep neural network that utilizes LSTM and RoBERTa models using Haystack. We found that these were computationally expensive as well as inaccurate.

2. Data

The data collected for this project consist of pdf reports from three different UN partner websites: International Red Cross Federation, International Water Association, and UNICEF. From these three agencies, we collected reports under UN Clusters that are often explored in the nonprofit sector:

- Health
- Logistics
- Nutrition
- Protection
- Shelter
- Water, Sanitation and Hygiene
- Camp Coordination & Management
- Early Recovery, Education
- Emergency Telecom's
- Food Security
- Humanitarian & Emergency Relief

We used a tool called Expertrec to crawl through the websites of these agencies and entered in these topics in order to retrieve information relevant to the query. We isolated the results to only pdfs and downloaded pdfs that were in English and non-duplicates. We also made note of the links for each of these pdfs by putting them in a separate file. Lastly, we looked through each of the pdfs and annotated them in terms of relevance to each topic. Some of these reports encapsulate more than just one of the 11 topics and

therefore we annotated the topic that each report may represent. In total, we have 217 pdfs from the three organizations all of which are annotated for the 12 topics.

Because our search engine requires the content of the pdf to be processed, we did a lot of preprocessing in this step. The first thing we did was extract the entire content of each pdf using Apache TIKA while parsing the metadata for each of the pdfs. We then created a dataframe that garnered a unique id we created, the title of each pdf, the content of each pdf, and the link associated with the pdf.

Afterwards, we cleaned the content extracted from the Apache TIKA by taking out special characters that were read in from TIKA. Because the content extraction for pdfs mining tools are still not perfect, we took out an absurd amount of newline characters as well as unencoded characters. This was important to extract the most relevant information out of each report. We then used the pre-trained text summarizing model BERT to create a summary from each pdf content in order to build our corpus that we would use for the ranking weight scheme algorithm. To work on our ground truth we created a query list containing 30 queries. For example, “School sanitation hygiene program”, “Audit of the Consolidated Financial Statements” and “disaster relief report” are all example queries we created to help evaluate our model’s results.

3. Related Work

In order to provide necessary context to the proposed solution of automatic lecture summarization, it is worth investigating previous research, identifying the pros and cons of each approach. In information retrieval (IR), queries and documents are typically represented by term vectors where each term is a content word and weighted by tf-idf, i.e. the product of the term frequency and the inverse document frequency, or other weighting schemes [1]. The similarity of a query and a document is then determined as a dot product or cosine similarity. A drawback of this approach, however, is that it may fail to detect relevant documents where no or only few words from a query are found. The semantic analysis methods such as LSA (latent semantic analysis) and LDA (latent Dirichlet allocation) have been proposed to address the issue, but their performance is not superior compared to common IR approaches. ‘Bridging the gap: Incorporating a semantic similarity measure for effectively mapping PubMed queries to documents‘ presented a query-document similarity measure motivated by the Word Mover’s Distance. Later for ranking they used BM25 [2]. This worked for them as IR scheme often fails to find relevant documents when synonymous or polysemous words are used in a dataset, e.g. a document including only “neoplasm” cannot be found when the word “cancer” is used in a query.

One promising recent innovation is models that exploit massive pre-training [3], leading to BERT [4] as the most popular example today. Researchers have applied BERT to a broad range of NLP tasks with impressive gains: most relevant to our document ranking task, these include BERTserini [5] for question answering and Nogueira and Cho (2019) for passage reranking. ‘Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval’ applied BERT to ad hoc document retrieval on news articles [6]. They had to address two challenges: relevance judgments in existing test collections are typically provided only at the document level, and documents often exceed the length that BERT was designed to handle. We also came across this challenge of pdfs exceeding the length that BERT could handle. They integrated sentence-level evidence for document ranking to address this challenge, where we split the pdfs and then aggregated the summary for ranking.

There are two ways of summarizing extractive and abstractive. Automatic extractive summarization researchers have attempted to solve the problem of time required for manual summarization. However, most of the approaches leave room for improvement as they utilize dated natural language processing models. Leveraging the most current deep learning NLP model called BERT, there is a steady improvement on dated approaches such as TextRank in the quality of summaries, combining context with the most important sentences [7].

Previous research on text extraction from pdf files have evaluated software text extraction toolkit and Apache TIKA gave good results [8]

In vector space model, Orkphol Et al constructed sentence vectors from a popular word-embedding approach, Word2vec, which is capable of effectively capturing semantic and syntactic similarities better than the LSA approach when dealing with a huge corpus of text. Cosine similarity was used to compute between those two sentence vectors for word sense disambiguation [10].

Lv Et al propose that when the documents are very long, BM25 fails! [9] That Okapi BM25 retrieval function tends to overly penalize very long documents. To address this problem BM25L "shifts" the term frequency normalization formula to boost scores of very long documents. with the same computation cost. BM25L is more effective and robust than the standard BM25. Since 80% of the reports in the data corpus were long (70-500 pages), we decided to implement BM25L.

4. Methodology

We retrieved our pdf data from three UN agencies (International Water Association (IWA), Unicef, and International Federation of Red Cross and Red Crescent Societies (IFRC)) using a web crawling tool called Expertrec to get at least 200 pdfs from these organizations. We made sure to only grab pdfs that were pertinent to 12 topics listed in the data section. Another measure we used was to only grab pdfs written in English and non-duplicates while getting the url associated with each pdf. As explained before, we annotated each of the pdfs to see if each pdf encapsulates one of the 12 topics. We understood that each pdf can relate to many of the topics, so we created this annotation to make it easier for more accurate retrieval later on. We also created a list of queries to use as a test case to see if our model would accurately retrieve pdfs close in topic relation to the query. All of our queries are something that someone working in the nonprofit sector may use when researching.

The next step was to use Apache TIKA to extract metadata and content from each of the pdfs. The content extraction included many special characters that are not easily encoded along with a vast amount of newline characters. We cleaned the data by removing all of those special characters and replacing newlines characters with spaces instead. This was to ensure that all the content was still there without changing any of the words. This method would also remove any chance of a newline character or unencoded character being included in the weighting scheme. From there, we created a dataframe using pandas and added the title, date, content, a unique identifier that we created and the url link associated with the pdf. We then exported this to a csv file to run a pre trained model BERT summarizer on the content data. Due to the large amount of content in each of the pdfs, we ran the BERT summarizer in batches and appended those to the csv. We made sure that each line in the BERT model was at least 60 words and not longer than 300 to save us some memory space when running our model.

Our team also went through every single one of our pdfs and manually annotated if this particular document fell into one of the 12 categories. Because a document can be related to more than just one of the 12 topics we wanted to categorize, it was important to annotate each topic that particular document was relevant to with a 1 or 0. We put this in a separate csv file that was ordered by document id.Using our new csv with the summaries, we created a final dataframe that merged together all our other features (title, id, content, URL, summary) with the annotation dataframe on unique id. This became our model's final data output.

The next step was to implement our baseline model. We decided to use word2vec to understand the similarity between a user query and the document that might want to be

retrieved. The first thing we did was to lower every single word in the summary and create a corpus from each document summary. We also lowered the user query for modularity. We then removed english stopwords from nltk's corpus as well as any punctuation so the model can only look at word similarities. We put this cleaned version of our summary into another column called "cleaned_summary". We then used gensim's word2vec model to create a word embeddings function for each word in the summary of our documents. We also created a cosine similarity function. Using these two functions, we created a ranking function that would look at the similarity from our query as well as each of our documents. Our team's ranking function returns the top 10 highest similarity document that retrieves the summary, title and URL associated with that document in our dataframe.

Cosine Similarity (A,B) = $\frac{A.B}{||A||\ x\ ||B||} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\ \sqrt{\sum_{i=1}^{n} B_i^2}}$

Our team also implemented a model of BM25L. We noticed that many models seem to have a bias towards lower length documents which may only return single topic documents. For the purposes of our project idea, we want to create a model that returned the most accurate document to the user query. So, using the same text preprocessing steps, we removed the stop words as well as any punctuation from our summary. We then put this processed text into another column called "cleaned summary". We then grabbed each of the cleaned summaries from our data frame and ran it through rank_bm25's BM25L function which uses the default hyperparameters (k = 1.5, b = 0.75, delta = 0.5) to create BM25L word weightings. This created the ranking for each document to query where we picked the top 10 and returned the summary, title and URL associated with that document. For our test case to run our base model against a few of our queries. We found some accurate results just by looking at the pdfs it returns.

BM25L = $\Sigma_{t\varepsilon q}\ log\ (\frac{N+1}{df_t+0.5}) * \frac{(k_1+1)*(c_{td}+\delta)}{k_1+c_{td}+\delta}$

BM25L builds on the observation that BM25 penalizes longer documents too much compared to shorter ones. The IDF component differs, to avoid negative values. The TF component is reformulated as $\frac{((k_1+1)\ .\ c_{td})}{(k_1+c_{td})}$ with $c_{td} = \frac{tf_{td}}{(1-b+b\ .\ (\frac{L_d}{L_{avg}}))}$. The component is further modified by adding a constant δ to it, boosting the score for longer documents [11] report using $\delta = 0.5$ for highest effectiveness

5. Evaluation and Results

To work on our ground truth we created a query list containing 30 queries. For example, School sanitation hygiene program, Audit of the Consolidated Financial Statements etc. We got the top 10 relevancy by running the BM25L algorithm. The parameter we used k = 1.5, b = 0.75, delta = 0.5.

We annotated the true relevance for the query and the documents retrieved and we then calculated precision, recall and F1 score based on annotated relevancy and predicted relevancy.

Precision = No. of relevant documents retrieved / No. of total documents retrieved
Recall = No. of relevant documents retrieved / No. of total relevant documents
F1 Score = 2 * Precision * Recall / Precision + Recall

We chose NDCG score to evaluate our model because it is one of the widely used TREC meric. It is designed for situations of non-binary notions of relevance. We annotated 30 queries and 5 predicted relevant documents for each queries. (150 in total) and similarly annotated for the baseline model as well to compare the results. Our ground truth annotation was based on relevancy score:
0: Not relevant
1: Somewhat relevant
2: Very Relevant

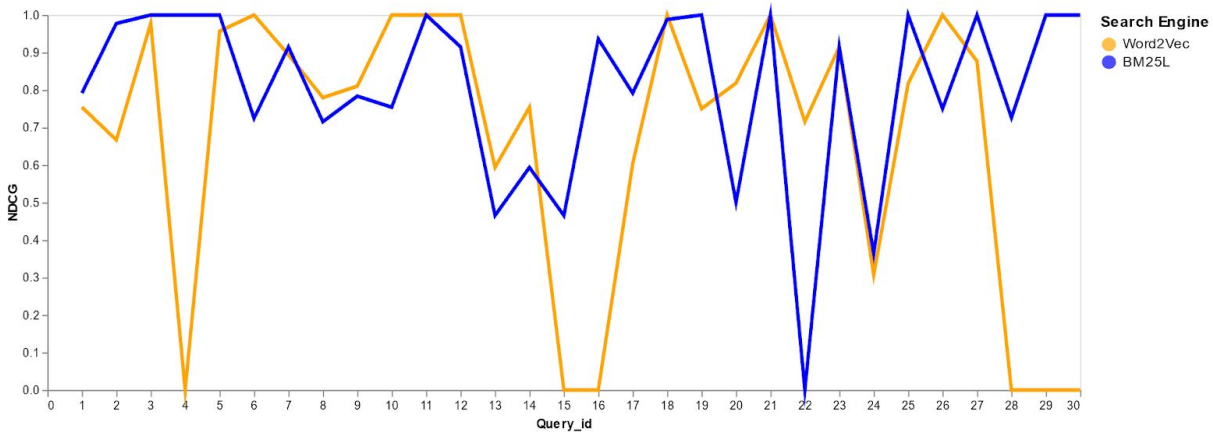

Figure 1: NDCG scores for 30 queries

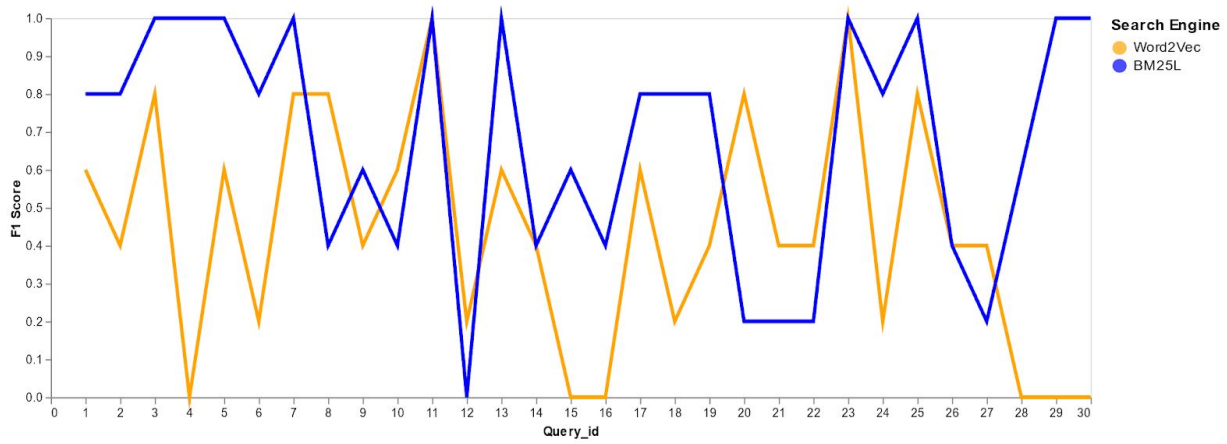

Figure 2: F1 scores for 30 queries

Overall BM25L performed better than the baseline model in terms of NDCG and F1 Score. We got F1-score as 0.36 and NDCG score as 0.66 for the baseline model. BM25L F1 score is 0.66 and NDCG score 0.80.

Results:

Link to the GIF: https://media.giphy.com/media/a5TRHUC3CBNawuNyhu/giphy.gif

6. Discussion

Overall, we saw that compared to our baseline model, BM25L models did well for broad queries (e.g “IWA reports”) and also did well for UN Cluster specific queries like, “Emergency Telecom”. However, it did not do well for queries requiring contextual understanding. For instance, “case study on blood donation drive” did not accurately retrieve relevant documents. We hypothesize that our baseline model (based on word2vec cosine similarity), performed better because it was able to understand lengthy queries requiring contextual understanding. However, the word2vec based baseline model is computationally expensive and takes a much longer time to retrieve reports over our BM25L model. Overall BM25L performed better than the baseline model in terms of NDCG as 0.80 and F1 score 0.66. Since NDCG score gives fine grained performance, we believe our model did well on UN and NGO reports.

7. Conclusion

The tool of quick search for the non-profit sector is still not readily available. With many organizations and agencies pushing out reports and news for different subsects, all this information is not available in one centralized location. Working in this industry requires a lot of individual effort in searching for the right information. Our model helps alleviate that problem in some ways by accurately retrieving relevant documents to a user's query. However, with our limited resources for documents our model can only give a finite amount of relevant documents to a user’s query. If this project continues to grow and build a large enough database of pdf reports from all NGO and UN-agencies, this model can be the most helpful tool for frustrated workers.

8. Other Things We Tried

Pdf preprocessing (pyPDF2, pdfplumber, pdfminer, AWS textract):

We tried several different types of pdf text extraction methods. Each time we implemented on a subset of pdfs to see which tool extracted the most content from the data. We noticed that pyPDF2 worked minimally and could not make sense of tables or captions for images. Pdfminer performed about the same as did AWS textract. Pdfplumber on the other hand performed comparably to Apache TIKA but did not have a good documentation for aggregating all the contents of the pdf extraction. Also, the meta data parsing performed subpar compared to TIKA.

Haystack:

Since we are dealing with documents/ reports we experimented with the Haystack library that is popular in semantic document search and retrieves more relevant documents. It has components: Document Store, Retriever, Reader and Finder. We used Elasticsearch as Document Store, BM25 as Retriever. Reader is a powerful neural model that reads through texts in detail to find an answer. We used RoBERTa via FARM or Transformers on SQuAD2.0 as a Reader. Finder binds Reader and Retriver. This mode did not give us relevant retrievals. For instance:

```
question = "water sanitation"
number_of_answers_to_fetch = 2

prediction = finder.get_answers(question=question, top_k_retriever=10, top_k_reader=number_of_answers_to_fetch)
print(f"Question: {prediction['question']}")
print("\n")
for i in range(number_of_answers_to_fetch):
    print(f"#{i+1}")
    print(f"Answer: {prediction['answers'][i]['answer']}")
    print(f"Research Paper: {prediction['answers'][i]['meta']['name']}")
    print(f"Context: {prediction['answers'][i]['context']}")
    print('\n\n')
```

```
12/10/2020 20:35:31 - WARNING - haystack.finder -   DEPRECATION WARNINGS:
            1. The 'Finder' class will be deprecated in the next Haystack release in
            favour of a new `Pipeline` class that supports building custom search pipelines using Haystack components
            including Retriever, Readers, and Generators.
            For more details, please refer to the issue: https://github.com/deepset-ai/haystack/issues/544
            2. The `question` parameter in search requests & results is renamed to `query`.
12/10/2020 20:35:31 - INFO - elasticsearch -   POST http://localhost:9200/document/_search [status:200 request:0.124s]
12/10/2020 20:35:32 - INFO - haystack.finder -   Got 10 candidates from retriever
12/10/2020 20:35:32 - INFO - haystack.finder -   Reader is looking for detailed answer in 57902 chars ...
Inferencing Samples: 100%|██████████| 1/1 [00:09<00:00,  9.35s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:08<00:00,  8.21s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:06<00:00,  6.71s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:06<00:00,  6.82s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:01<00:00,  1.53s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:01<00:00,  1.53s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:03<00:00,  3.05s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:03<00:00,  3.04s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:04<00:00,  4.47s/ Batches]
Inferencing Samples: 100%|██████████| 1/1 [00:04<00:00,  4.55s/ Batches]Question: water sanitation


#1
Answer: however
Research Paper: Uganda Gender and Sanitation and Hygiene _GWA
Context: n views siting type structure access collected realize objective structures described put place mobilization implementation national guidelines sanitation sp
```

Figure 3: Training Haystack

The reason could be that we don't have a large number of documents to train a deep learning model (RoBERTa).

RoBERTa on UN Cluster classification:

We also tried classification of reports to different UN clusters like Health, Education etc. To achieve this we had annotated reports with respective UN clusters. This annotation was treated as a label for RoBERTa model. The annotations were binary .We used SimpleTransformers library to implement the classification. With epoch 3 (optimal to avoid overfitting) we got a binary F1 score of 0 for 8 out of 11 UN Clusters. Clearly this method did not work as it would need more data.

LIME and Logistic regression for UN Cluster classification:

In order to classify reports into UN clusters we implemented Logistic Regression and accessed it using LIME (Local Interpretable Model-Agnostic Explanations). This method was computationally intensive to run on all the reports for all UN clusters. It did well for some clusters like Health and WASH but not for Logistics and Camp coordination. The main aim for the classification task was not fully achieved. More annotated data would be required.

Deep learning using LSTM:

One of the Deep Learning models we tried was LSTM. We passed the pre-processed summary of the pdf as an input to the model and created a word2vec representation of these vocabularies (numbers). For doing this, we initialise a dictionary of variable vocabulary() which stores each word as a key and a number as a value respectively. Another variable called inverse_vocabulary() which is a list that holds the value or number for each unique word. It was initialised as an <unk> token since we want to zero pad the words with a number zero. We did not want to assign any word a number 0. Hence, initialised with a <unk> token which holds the value zero.

Next, we created the embedding matrix for my vocabulary. For which we used the gensim library and google's pre-trained word2vec model. Google's pre-trained word2vec model gives a 300 dimensional vector for each word which will be fed to the Embedding layer of my model. Since, we will pad my sentences with a zero, we initialise the embedding matrix's zeroth element as zero. The size of the embedding matrix will be (Size of Vocabulary + 1 (for zero) X 300 (embedding dim)).

Since we use Keras Functional API, we defined three functions namely embedding(), middle() and predict(). The embedding function took the embedding matrix as an input and was Trainable as True when the network was getting trained. The embedding output is feeded into the middle function module as input on which the max pooling, lstm and dense layers are applied. The dense layer outputs a 128 feature maps which are then passed to the predict function which computes an L1 distance on these feature maps and then using a Dense layer with one neuron outputs a prediction of 0 or 1. With this we achieved the accuracy of 27%, which is below the baseline model we have chosen. We chose to focus the majority of our efforts on the classic IR algorithms as the input data size we have is not good enough for deep learning models.

9. What You Would Have Done Differently or Next

As the next step, we would like to collect more documents to implement Deep learning models and UN cluster classification. For all the models we used the summary of the pdfs as an input feature. Potential future work could include working on feature engineering and deriving more metadata like Author name. We would also like to Improve the UI of the search engine with additional features like highlighting the key words of the query in the summary and titles, listing similar documents, etc.

References

1. Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24.5 (1988): 513-523.

2. Kim, Sun, et al. "Bridging the gap: incorporating a semantic similarity measure for effectively mapping PubMed queries to documents." Journal of biomedical informatics 75 (2017): 122-127.

3. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana.

4. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota.

5. Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019b. End-to-end open-domain question answering with BERTserini. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 72–77, Minneapolis, Minnesota.

6. Wei Yang, Haotian Zhang, and Jimmy Lin. 2019c. Simple applications of BERT for ad hoc document retrieval. arXiv:1903.10972.

7. Miller, Derek. "Leveraging BERT for extractive text summarization on lectures." arXiv preprint arXiv:1906.04165 (2019).

8. Allison, Timothy B., and Paul M. Herc. MITRE CORP MCLEAN VA, 2015.

9. Lv, Yuanhua, and ChengXiang Zhai. "When documents are very long, BM25 fails!." Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. 2011

10. Orkphol, Korawit, and Wu Yang. "Word sense disambiguation using cosine similarity collaborates with Word2vec and WordNet." Future Internet 11.5 (2019): 114.

11. Kamphuis, Chris et al. “Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants.” *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II* vol. 12036 28–34. 24 Mar. 2020, doi:10.1007/978-3-030-45442-5_4