Document Retrieval Using Deep Learning

Sneha Choudhary
School of Data Science
University of Virginia
Charlottesville, VA
sc4xc@virginia.edu

Haritha Guttikonda
School of Data Science
University of Virginia
Charlottesville, VA
hg5mn@virginia.edu

Dibyendu Roy Chowdhury
School of Data Science
University of Virginia
Charlottesville, VA
drc2qw@virginia.edu

Gerard P. Learmonth
School of Data Science
University of Virginia
Charlottesville, VA
jl5c@virginia.edu

***Abstract*—Document Retrieval has seen significant advancements in the last few decades. Latest developments in Natural Language Processing have made it possible to incorporate context and complex lexical patterns to document representations. This opens new possibilities for developing advanced retrieval systems. Traditional approaches for indexing documents suggest averaging word and sentence encoding to form fixed-length document embeddings. However, the common bag-of-word approach fails to incorporate the semantic context, which can be critical for understanding document-query relevancy. We address this by leveraging Bidirectional Encoder Representations from Transformers (BERT) to create semantically rich document embeddings. BERT compensates the limitations of the Term Frequency Inverse Document Frequency (TF-IDF) by incorporating contextual embeddings. In this paper, we propose an ensemble of BERT and TF-IDF for a document retrieval system, where TF-IDF and BERT together score the documents against a query, to retrieve a final set of top K documents. We critically compare our model against the standard TF-IDF method and demonstrate a significant performance improvement on MS MARCO data (Microsoft-curated data of Bing queries).**

***Index Terms*—information retrieval, bert, tf-idf, query expansion**

I. Introduction

The last decade has witnessed a significant boost in data creation, generation and usage due to the pervasive penetration of the internet in our daily lives. We create 2.5 quintillion bytes of data everyday [1], of which machine-readable text is a significant portion. Availability of data and the need to access it have escalated the demand for better and efficient retrieval systems. A document retrieval system matches user queries to a set of documents and retrieves the most relevant documents. Earlier methods of retrieval used bag-of-words (BoW) assumptions and exact term-to-term matching. These retrieval methods are limited in their ability to model the context, semantics, and inter dependency of document terms. This has led to the development of distributed representation approaches. Distributed representations of terms embed semantics and relation between terms, but they are still context free. In document retrieval, it is critically important to generate document and query vectors that are representative of the context of the text in order to bridge the gap between queries and documents.

Recent advancements in Natural Language Processing (NLP) offer new opportunities to understand complex language patterns between queries and documents and therefore, build an efficient retrieval system. Bidirectional Encoder Representations from Transformers (BERT) [2], introduced by Devlin et al. demonstrate a new method of pre-training language representations that produce state-of-the-art results in various NLP tasks. In this paper, we adapt BERT by implementing an ensemble of Term Frequency - Inverse Document Frequency (TF-IDF) and BERT to overcome the shortcomings of TF-IDF. Both models run in parallel, and documents are ranked based on the weighted sum of each score.

We also compare our approach with a baseline model (TF-IDF) and demonstrate a significant improvement in the performance of our proposed retrieval system. We further analyzed the impact of different query lengths, embedded Named Entity Recognition (NER) tasks, and different weights for BERT and TF-IDF on our model.

II. Background

In neural Information Retrieval (IR), there are two types of architectures for relevance matching, namely Interaction Based and Representation Based.

A. Interaction Based

An interaction-based model builds a joint representation of interactions between query and document. It then uses a deep neural network to learn hierarchical matching patterns and produce a matching score. Existing interaction-based models such as ARC-II [3] and MatchPyramid [4] employ a Convolutional Neural Network to learn hierarchical matching patterns over the matching matrix. These models are essentially position-aware using convolutional units with a local "receptive field" and learning positional regularities in matching patterns. This technique was designed for image recognition tasks and has been shown [5] to work well on semantic matching problems due to the global matching requirement (i.e., all the positions are important). However, it may not be suitable for ad-hoc retrieval tasks, since such positional regularity may not exist [5]. Moreover, this approach requires similarity assessments for training the interaction of document-query pair over labels. The difficulty in obtaining large amounts of labelled data and availability of unlabelled data drove this research toward semi-supervised and unsupervised techniques.

978-1-7281-7145-6/20/$31.00 ©2020 IEEE

B. Representation Based

A representation-based model generates a distributed representation of a query and documents using a deep neural network, and then calculates the relevance score of each query-document pair using a similarity metric — usually the cosine similarity. We have restricted our discussion to representation-based models as they are unsupervised and do not require labelled data.

III. Related Work

Salton et al. extensively studied the use of term frequency and document frequency to rank the documents in a Vector Space Model (VSM), now known as TF-IDF [6]. Since then, several attempts have been made in [7], [8] to explain TF-IDF in a theoretical framework. Hiemstra formulated probabilistic justification for TF-IDF to better understand statistical ranking mechanisms in [7]. Robertson et al. [8] introduced the much-celebrated BM25 scoring function where term counts are modified as two Poisson distributions. Major drawbacks of these approaches are that they do not account for ordering of terms and they ignore semantics.

Bengio et al. introduced the concept of word embeddings by simultaneously learning a language model that predicts a word given its context and representation [9]. The use of these word embeddings became ubiquitous in information retrieval tasks. In 2013, Google released an efficient method for learning high-quality distributed vector representations called Word2Vec, built using Skip-Gram architecture [10]. This method was designed to capture semantic and syntactic relationships between words. A major limitation of Word2Vec is that it assumes fixed vocabulary length and cannot handle out-of-vocabulary terms.

Advancements in deep neural networks made it convenient to train complex models on larger data. This led to an active research in neural Information Retrieval. Deep Semantic Similarity Model (DSSM) by Huang et al. [11] trained query and document in two deep neural network models with a fully connected layer and cosine distance to measure similarity between them. Inspired by this [11], many variants of DSSM were introduced in [12], [13]. However, there was a paradigm shift to pre-trained architectures which make large-scale IR algorithms significantly efficient. Use of these pre-trained architectures were then extended to word embeddings and deep language representation models such as BERT [2] and ELMo [14] in recent years.

IV. Methodology

A. General Framework

Fig. 1 shows the general framework of our model. We used an ensemble of TF-IDF and BERT that works in parallel, giving two different relevance scores for every query-document pair, which then determines the final rank of the document.

In order to reduce the time of retrieval, document vectors are generated beforehand and stored locally. So, during retrieval, we only have to vectorize queries. Once a user query is raised, the first step is to calculate the respective TF-IDF and BERT vector representations of the query. Next, relevance score for each query-document pair over the entire corpus is calculated. Cosine similarity and dot product are the relevance scores used for BERT and TF-IDF models respectively. If q is a query vector and d is a document vector in a set of Q queries and D documents, relevance scores are calculated based on Algorithm 1.

Algorithm 1 Document Retrieval

for $q \in Q$ **do**
  for $d \in D$ **do**
    $tfidf\_score = q \cdot d$
    $bert\_score = \dfrac{q \cdot d}{|q| * |d|}$
    $score(q, d) \leftarrow SUM(tfidf\_score, bert\_score)$
  end for
  $document\_ranked \leftarrow sort(score)$
  $top\_k\_docs \leftarrow document\_ranked[: k]$
end for

A linear combination of these two scores is used to determine the cumulative score of each document based on which the documents are ranked. Top K documents are retrieved based on this rank.

B. TF-IDF Model

VSMs are widely used for text representation because of their simplicity and interpretability. In VSMs, weights are assigned to each term based on different weighting schemes. TF-IDF is the most common weighing scheme, particularly in IR. Intuitively, TF-IDF determines the importance of a given term in a particular document. The term frequency (TF) component refers to the number of times a term appears in a given document normalized by size of the document to prevent bias toward longer documents. Inverse document frequency (IDF) determines the number of documents in the corpus in which the given term occurs. It is a measure of how common or rare the term in the entire corpus. By using these two statistics in combination, a word is given more importance if its frequency is high in a document and loses its importance if it has high frequency in the entire corpus. By doing this, we assign higher weights to the keywords and lower weights to the common words such as articles and prepositions. If t, d represent terms and documents respectively, TF-IDF is calculated as:

$$TF(t,d) = \frac{Count\ of\ t\ in\ d}{Number\ of\ terms\ in\ d} \tag{1}$$

$$IDF(t,d) = log_e \frac{Number\ of\ documents}{Number\ of\ documents\ with\ t} \tag{2}$$

TF-IDF is the product of the above two statistics.

$$TF - IDF(t,d) = TF(t,d) * IDF(t,d) \tag{3}$$

Both the documents and queries were prepossessed using Python's NLTK library [15]. Preprocessessing steps included

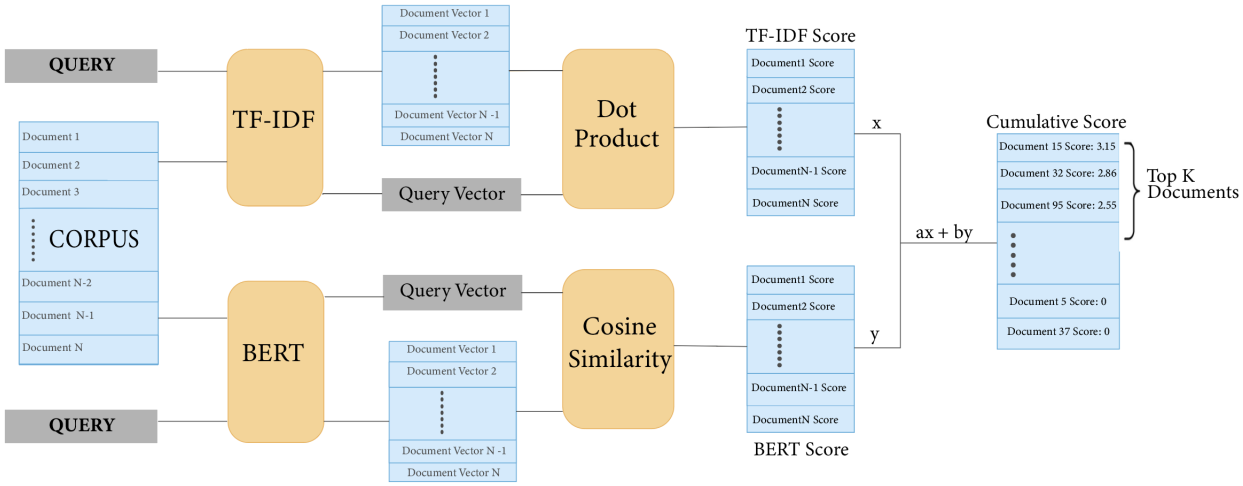

Fig. 1: General Framework

Tokenization and Stemming. Once documents were preprocessed, TF-IDF vectors were generated based on above formulas and stored locally. After every user query, the query text is preprocessed and the TF-IDF vector for each query was generated. The relevancy score for each document was calculated based on dot product of the query and document vectors.

C. BERT Model

BERT has shown promising results in various NLP tasks such as NER, next sentence prediction, and leaving behind models such as ELMo and LSTM [2]. Its multi-layer structure with a self-attention mechanism makes it possible to encode context into word embeddings. The BERT model is bidirectional, which is jointly conditioned on both left and right context across all the layers to develop deep bidirectional representations from unlabelled data.

BERT model can be used for two purposes: pre-training and fine-tuning. In the pre-training stage, the model is trained over unlabeled data, and in the fine-tuning stage, the pre-trained weights are used to train labeled data for various downstream tasks such as NER, classification etc.

In this paper, BERT is pre-trained to provide corpus-specific word embeddings. First, every sentence or document is preprocessed by BERT tokenizer specified in [2]. Each sentence is marked with a CLS token at the beginning and a SEP token at the end as shown in Fig. 2.

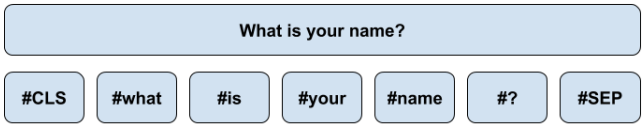

Fig. 2: BERT Tokenization

After tokenization, the model was pre-trained by unsupervised learning based on Masked Language Models using Next Sentence Prediction (NSP). From the pre-trained model, document vectors were extracted and stored in JSON format. During the run time, only a query was transformed into a vector using the same pre-trained BERT network. Next, the cosine similarity was computed between every query-document pair to give each document a score. This score can be called a ”semantic score” as it essentially captures the semantic relationships across documents [2]. With BERT, our main aim was to rank the documents in the corpus according to the semantic relevance.

V. Experimental Setup

To measure the effectiveness of our approach, we selected a subset of the MS MARCO data. MS MARCO is a large-scale data built for document re-ranking with over 1 million queries and 3.2 million documents [16]. All the queries have been sampled from anonymous Bing searches. For both pre-training and testing purposes, we extracted a subset of 3,000 queries and relevant top 100 documents per query, for a total of 28,500 documents.

The BERT model was pre-trained using a Masked Language Model and NSP across all layers for 20 steps, using a maximum sequence length of 128 tokens. Other hyperparameters such as batch size and learning rate were held identical to [2].

Document vectors were generated by taking the embedding corresponding to the CLS token from the last hidden layer of the 12-layered BERT architecture. As suggested in the original paper [2], this is a special classification token which represents the aggregate sequence representation.

A. Metrics

Since the MS MARCO data was labeled with relevancy ranking of each document per query, we chose two metrics - Mean Average Precision (MAP) and Normalized Discounted

Cumulative Gain (NDCG). A detailed description of each metric is explained below:

1) MAP: Precision is a measure of model correctness with respect to its overall predictions. To calculate precision of the model, precision values from all the queries are averaged giving the mean average precision value. Precision for K documents can be determined by

$$P@k = \frac{[relevant\ docs] \cap [retrieved\ docs]}{[retrieved\ docs]} \tag{4}$$

For evaluating this metric, we considered a document retrieved by the model as relevant if it is present within top 10, 5, and 3 documents in the MS MARCO data with respect to every query. By default, our model retrieves 10 documents per query but also evaluated MAP scores on smaller slices, since depending on the situation, a user may be only interested in a narrow selection of the retrieved documents.

2) NDCG: NDCG measures ranking quality of a retrieval system. It accounts for the ranks associated with each document while evaluating overall performance of the model. It follows three basic principles:

1. Cumulative Gain - Highly relevant documents are more useful than moderately relevant documents.

2. Discounting - Relevant documents should come up in earlier set of results.

3. Normalization - Overall score of the model should be irrelevant to the type of the query.

The formula used for this metric is:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{5}$$

$$where, DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \tag{6}$$

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{rel_i}{log_2(i+1)} \tag{7}$$

VI. Results

We tested two different initializations for the BERT model. The weights were generated using a next-sentence prediction task on the Wikipedia Corpus, Book Novel Corpus [2] and on MS MARCO data. Table I shows that BERT trained on MS MARCO data performed better than the general purpose pre-trained BERT [2].

Even so, the MS MARCO-tuned BERT did not outperform TF-IDF. Most likely, this is because many of the MS MARCO queries are direct and they do not heavily depend on the semantics of the document. For instance, queries such as "how long is the flight from Chicago to Cairo" and "how much fiber is in carrots" can be considered quite straightforward. The inter dependency of words and position of words do not heavily impact the results. Regardless of the ordering, tokens such as "long", "flight", "chicago" and "cairo" are enough to infer the question's intent and are likely to appear directly in any responsive document. Intersection of such keywords between the query and documents makes TF-IDF a fairly good approach on this data.

Additionally, BERT and TF-IDF use 128 and 632 encoding lengths respectively. It is highly likely that the BERT encodings used here are only long enough to function as 'document summaries' and not as the accurate descriptors of relevant topics and facts. BERT can either be used to extract language features from the textual data or fine-tune on specific tasks such as question answering, NER, classification etc. In our approach, we were only interested in extracting document vectors from BERT and not focused on fine-tuning approaches. However, when BERT is fine-tuned with labeled data for question answering, it is stand-alone sufficient to provide the best results [17].

The composite BERT and TF-IDF model did outperform the traditional TF-IDF method by approximately 5 percent. (Table I). This evidence suggests that BERT can recover semantic and contextual content of a document which BoW retrieval methods like TF-IDF discard. This is useful in edge cases where even though a document and a query might use similar words, the meaning of the two is drastically different. In such scenarios, it becomes important to understand each word as a part of a larger context. For instance, examples such as

"It's fun, it's not boring"
"It's not fun, it's boring"

denote two completely different sentiments. However, TF-IDF would treat the two queries the same and retrieve the same set of documents. Hence, it becomes important to understand how words compose the meaning of longer phrases. BERT helps in differentiating between the two and fetches different result sets per query. When BERT is incorporated with TF-IDF, the precision value increases from 80% to 85% as shown in Table I.

VII. Analysis

A. Encoding Mixing Weights

Taking the BERT encoding as a contextual clue to pick between similar TF-IDF vectors, we then investigated the appropriate level of context to incorporate in our model by evaluating the model across various linear weighting ratios for the TF-IDF and BERT score. The results associated with this experiment are displayed in Fig. 3. These weights can be fine-tuned to fit a different corpus. In our case, we observed highest MAP value when the ratio of TF-IDF and BERT score is 1.

TABLE I: Performance Metrics

	Metrics			
Model	**MAP@3**	**MAP@5**	**MAP@10**	**NDCG**
BERT [2]	0.40	0.41	0.40	0.43
BERT (MS MARCO)	0.53	0.53	0.50	0.54
TF-IDF (Baseline)	0.80	0.79	0.74	0.74
BERT + TF-IDF	**0.855**	**0.84**	**0.79**	**0.79**

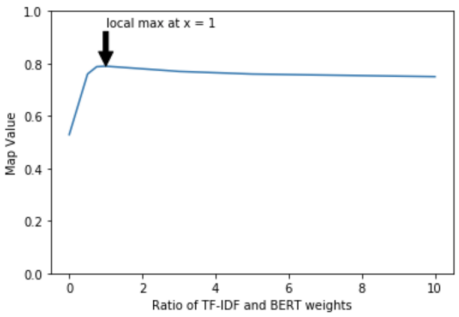

Fig. 3: Mixing weights of TF-IDF and BERT

B. Query Length

MAP01 is MAP value of TF-IDF model while MAP11 is MAP value of BERT + TF-IDF model. NDCG11 is the NDCG score of BERT + TF-IDF model. From Table II, we can observe that queries with length less than 5 tokens and greater than 16 tokens have better performance in the baseline (TF-IDF) model than the queries with medium length (6 to 15 tokens). By incorporating BERT, we can see significant improvement in the MAP values of the medium length queries.

C. Question Types

In order to investigate the impact of embedded NER tasks on model performance, we took the query types to be rough proxies for NER-heavy and more general retrieval tasks. Table III shows the performance breakdown for each query type. The MAP values for the queries which are NER-heavy (queries with 'what', 'when', 'where', 'which', 'who') are 6% less than the queries which are more general (queries with 'why', 'define', 'do', 'how'). Using combination of TF-IDF and BERT instead of just TF-IDF increased the performance of almost all the query types. We can also observe maximum improvement in the 'when' and 'who' questions.

VIII. Conclusion

We introduce an ensemble of BERT and TF-IDF approaches to develop an end-to-end document retrieval system. Using this parallel architecture, we tested on a subset of MS MARCO data, demonstrating a significant improvement over traditional retrieval models. NDCG and MAP values provide enough evidence to suggest that our approach showed superior results.

TABLE II: Query Length and Performance

Length	count	MAP01	MAP11	NDCG11
len $\leq$5	1473	0.77	0.80	0.76
6 $\leq$ len $\geq$ 10	1374	0.73	0.79	0.74
11 $\leq$ len $\geq$ 15	136	0.74	0.81	0.75
16 $\leq$ len $\geq$ 22	17	0.79	0.78	0.76

TABLE III: Performance by Query Type

Tag	count	MAP01	MAP11	NDCG11	NER
What	1245	0.74	0.78	0.73	NER Heavy
When	42	0.77	0.84	0.77	
Where	75	0.74	0.78	0.74	
Which	39	0.75	0.80	0.77	
Who	84	0.72	0.81	0.75	
Define	241	0.77	0.80	0.76	General
Do	82	0.73	0.74	0.75	
How	460	0.73	0.79	0.73	
Why	16	0.8	0.92	0.78	

Pretraining BERT on our own data improved the accuracy of the model. We analyzed the conceptual difference between BERT and TF-IDF approaches. To expand the scope of the approach followed in this paper, we are interested in extending this to multilingual documents. Also, further analysis on query expansion techniques can be accounted as a part of the future work.

IX. Acknowledgments

This work is sponsored by Logistic Management Institute (LMI). We extend our sincere gratitude to Michael Anderson of LMI for his consistent support and insights. We thank Prof. Gerard Learmonth of University of Virginia for his invaluable guidance over the year in crafting this research.

References

[1] Bernard Marr. "How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read". Forbes, May 2018.
[2] J Devlin, M W Chang, K Lee et al., Bert: Pre-training of deep bidirectional transformers for language understanding[J], 2018.
[3] B. Hu, Z. Lu, H. Li, Q. Chen, "Convolutional neural network architectures for matching natural language sentences", Proc. Advances Neural Information Processing Systems, pp. 2042-2050, 2014.
[4] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. "Text Matching as Image Recognition", In AAAI, 2016
[5] J. Guo, Y. Fan, Q. Ai, W. B. Croft, "A deep relevance matching model for ad-hoc retrieval", Proc. 25th ACM Int. Conf. Inf. Knowl. Manage., pp. 55-64, Oct. 2016.
[6] G. Salton, C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, vol. 24, no. 5, pp. 513523, 1988.
[7] D. Hiemstra, "A Probabilistic Justification for Using TF x IDF Term Weighting in Information Retrieval", Int'l J. Digital Libraries, vol. 3, no. 2, pp. 131-139, 2000.
[8] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3. In TREC '94: The third text retrieval conference, 1994.
[9] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, "A neural probabilistic language model", J. Mach. Learn. Res., vol. 3, pp. 1137-1155, Mar. 2003.
[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, "Distributed representations of words and phrases and their compositionality", Proc. Advances Neural Information Processing Systems, pp. 3111-3119, 2013.

[11] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, "Learning Deep Structured Semantic Models for Web Search using Clickthrough Data", Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2013.
[12] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, "Learning semantic representations using convolutional neural networks for Web search", Proc. 23rd Int. Conf. World Wide Web, pp. 373-374, 2014.
[13] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval", Proc. CIKM, 2014.
[14] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in NAACL-HLT, 2018, pp. 2227–2237.
[15] S. Bird, "NLTK: The natural language toolkit", Proc. Joint Conf. Int. Committee Comput. Linguist. Assoc. Comput. Linguist. Interact. Presentat. Sessions, pp. 69-72, 2006.
[16] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng,Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, Tong Wang, "MS MARCO: A human generated MAchine Reading COmprehension dataset", 2016.
[17] Jinhyuk Lee et al., Biobert: pre-trained biomedical language representation model for biomedical text mining, 2019, [online] Available: .