Valéria Pereira de Souza

Strategies to deal with the 512 token limitation in BERT-based ranking models: performance and latency impacts

Belo Horizonte, Minas Gerais

2021

Valéria Pereira de Souza

Strategies to deal with the 512 token limitation in BERT-based ranking models: performance and latency impacts

Research work as a partial requirement to obtain a bachelor's degree in Information Systems

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Marcos André Gonçalves

Belo Horizonte, Minas Gerais
2021

Sumário

1 INTRODUCTION 5
1.1 Objectives 6

2 BACKGROUND 7
2.1 Learning to Rank and Neural IR 7
2.1.1 Single and multistage pipelines 7
2.2 Long Document Representation Techniques 8
2.2.1 Rank and Fusion 9
2.2.1.1 Simple Applications of BERT for Ad Hoc Document Retrieval[1] 9
2.2.1.2 Applying BERT to Document Retrieval with Birch[2] 10
2.2.1.3 End-to-End Open-Domain Question Answering with BERTserini[3] 10
2.2.1.4 Exploring Classic and Neural Lexical Translation Models for Information Retrieval: Interpretability, Effectiveness, and Efficiency Benefits 10
2.2.2 Normalized Datasets 11
2.2.3 Document precomputation 11
2.2.3.1 TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval[4] 11
2.2.3.2 Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching[5] 11
2.2.4 Clipping 12
2.2.5 Summarization 12
2.3 State Of The Art in Neural IR 13

3 METHODOLOGY 14
3.1 Summarization Techniques 14
3.2 MS Marco Dataset 15
3.3 Bert model - Reranker 16
3.4 Experimental Setup 16

4 RESULTS 17

5 CONCLUSION 18

REFERÊNCIAS 19

I dedicate this work to my parents, that proved me it’s never too late.

Agradecimentos

Thank you my beloved friends Clarissa, Daniel, Ewerton, Fernanda, Fran, Gabriel, Giovanni, Matheus Henrizo, Mirian, Rafa and Yasmim. Without you this journey would not have been so joyfull. I would not have grown as much as I have. It is immensely inspiring to walk by your side.

Thank you, Igor, for being such a caring and supportive partner in such trial times as finishing an academic degree throughout a pandemic. You provided me with all the strength, love and pizza I needed to be my best. I love you.

1 Introduction

Search engines are ubiquitous in today's informational behaviour of computer users as document collections have became increasingly larger. Under the hood, search engines perform many complex tasks that aim to retrieve relevant information, in respect to an information need. Among different retrieval tasks that can be performed, the ad hoc text retrieval is when the information need is expressed in a query composed of a few keywords and it's matched with some, usually long, textual documents.

Search engines are a combination of stacked techniques that starts with the selection of a subset of candidate documents from the initial pool of documents. The candidates are delivered to subsequent algorithms that will rank this subset in terms of their relevance to fulfill the query[6].

A classic challenge in ad hoc retrieval is the vocabulary mismatch where query terms may not appear at all in a relevant document. Traditional candidate generation techniques make use of handcrafted features and exact term matching and so, are not able to bridge the gap between query and document vocabulary. Zhao and Callan[7] show that an average query term fails to appear in 30-40% of the documents that are relevant to the user query. According to Carpineto and Romano, query length has remained the same, about 2.3 terms, since ten year before the study. At the same time, the collections have grown larger making the effects of polysemy more severe.

Query expansion techniques that are used to address this issue heavily rely on handcrafted dictionaries and ontologies. If a user searches for a certain term and this query is expanded to it's synonyms, one has to built this correlations manually. As languages evolve through time, query expansion databases became obsolete and updating them is very expansive given the need of specialists.

Neural language models have led to breakthroughs in many tasks in the field of natural language understanding, including ad hoc information retrieval. Neural models that perform matching in a latent space tend to be more robust towards the vocabulary mismatch problem compared to lexical term-based matching models [8]. A new wave of breakthroughs came with contextualized language models, where embeddings are computed in a self-supervised manner. By computing deeply-contextualized semantic representations of query–document pairs, these language models help minimize the term mismatch problem[9].

These improvements pushed forward a ranking strategy that does not rely on a candidate generation phase, but attempts to rank directly from the document collection. This strategy aims to avoid the errors of the candidate generation phase to propagate into the ranking phase. This area of research is not new, but neural models are very computationally costly making this type of systems impracticable. Since modern language models have been presented, new strategies for one stage ranking have been proposed with way better latency than previous ones. This fact shifts some of the research light towards investigating deeper this type of solution, in the hope to achieve fast and precise retrieval.

BERT[10] - Bidirectional Encoder Representations from Transformers - is a pre-trained deeply contextualized language model presented in 2018 that has already inspired several publications that evolved the SOTA in the ranking arena. BERT-based models have to deal with a know limitation: because of it's architecture, they can only accept inputs 512 tokens long[**?**]. This limitation is due to its quadratically increasing memory and time consumption on the length of the input text[11]. When dealing with long texts, Bert-based models have to make use of some strategy to approach this problem. For the purpose of this work, long texts are ones that exceed the 512 token limitation.

1.1 Objectives

The purpose of this work is to experiment on different techniques of document representation found in the scientific literature, aiming to isolate and understand the impacts of each experimented technique on score and latency.

2 Background

2.1 Learning to Rank and Neural IR

Learning to Rank (LTR) is a class of techniques that apply supervised machine learning to solve ranking problems. They act at as the last step in a multi-stage ranking setup, where other algorithms perform the initial stages of candidate generation. They uses feature vectors that depend on handcrafted features. Candidate generation algorithms such as BM25[12] rely on term matching to filter an initial subset of candidate documents.

Neural IR (NIR) is an emerging field that applies specifically neural networks to ranking problems. Research publication in the area has been increasing, along with relevant workshops, tutorials and plenary talks[6]. That is because, unlike traditional learning to rank models, these architectures depend less on manual feature engineering and more on automatically detecting regularities in good matching patterns. The power of neural ranking models lies in the ability to learn from the raw text inputs for the ranking problem to avoid or minimize many limitations of hand-crafted features and exact term matching[13] such as the vocabulary mismatch.

Retrieval systems have to account for many complex and varied scenario posing challenges to the development of new solutions, so compromises have to be made. For example, beyond precise, IR systems must have low latency to serve results in timely manner. Traditional IR techniques are less data hungry, therefore less computationally costly. On the other hand neural solutions are better at addressing the problem of vocabulary mismatch. In terms of training, neural models learn from raw text, as opposed to traditional LTR, that rely on handcrafted features.

Another difference is regarding rare terms: latent representations are less robust to them because their term understanding depends on how often the models sees such term during training time. Exact matching models, like BM25, on the other hand, can precisely retrieve documents containing rare terms[6]. Learning latent representation of text is important for dealing with vocabulary mismatch, but exact matching is also important to deal with rare terms and intents.

2.1.1 Single and multistage pipelines

Information retrieval systems are expected to output a list of documents, ordered by relevance in respect to a given query. Due to the increasingly larger collections, current systems adopt a two steps approach. The first step, the candidate generation phase, is downsizing the collection and ranking the subset with an efficient, but not so effective model, tuned for high recall. The second step, the re-ranking phase, is to rank the subset with high precision.

Deep neural networks have successfully addressed the semantic matching problems, but it's power is limited to the result of the candidate generation phase, so why not using them from bottom up? Neural ranking models rely on dense vector representations, that are very costly to generate. Matching every document in a collection with the query is infeasible in current setups. Neural models have shown to perform better at ranking, but their computational cost makes this approach not viable for real time inference. However, simply forming the pipeline by appending a BERT reranker to an effective first-stage retriever does not guarantee an effective final ranking. [14]

Given that neural models show a lot of potential in addressing the vocabulary mismatch problem, researchers keep looking for ways to use such models to rank the whole collection without the need of a candidate generation phase.

2.2 Long Document Representation Techniques

In order to investigate the document representation techniques being used by BERT-based ranking strategies, 12 papers were selected for reviewing according to the following scope: published between 2019 and February, 2021 that addressed the retrieval tasks such as ad hoc retrieval, question answering and document recommendation. The preferred task was ad hoc retrieval but other tasks were considered when a novel approach was detected and presumed useful for ad hoc retrieval.

No distinction was made between ranking in one or two stages, since in both the limitation of tokens is an issue. The paper had to use BERT as a core model for experimentation or part of the proposed architecture.

A line of research attempts to simplify the structure of transformers, but currently few of them have been successfully applied to BERT [15] and so architectures that are built with the sole purpose of dealing with the input length problem were left out of the scope of this work.

The search for papers was made in several steps. The first phase was keyword search in the Arxiv repository and Google Scholar. Keywords used included "bert, information retrieval, ranking". For the Arxiv repository, a script was used to scrape and filter results. For the google Scholar, search was performed manually. Two papers were selected from the MS Marco dataset leaderboard. Not all entries in the leaderboard have accompanying papers or use BERT. The final stage was a snowballing search in the references of the previously found papers. Thirteen papers were found to fit the criteria and were reviewed for this work. In order to perform IR experiments for Question Answering with BERT, Crijns[16] selected three strategies to deal with BERT's input limitation. The simplest is concatenating query and document and clipping the content to 512 tokens and ignoring

Technique	Quantity
Rank and Fusion	5
Normalized datasets	2
Document precomputation	2
Clipping	2
No information	1
Total	12

Tabela 1 – Summary of techniques and quantities

any token after. In the Extraction-based Summarization technique, important words are selected from the document based on some metric. For example, the author uses TF-IDF to select, and concatenate to the query, the top words until the limit of 512.

Finally, the Rank and Score Fusion technique, is when multiples runs of the model are made to different or all passages of the document. Each run is assinged a score. The second stage can be ranking the passages or fusing scores in many different ways (averaging scores, for example) to determine which passages will be concatenated to the query.

In this work, five different methods were employed to deal with the input restrictions: clipping as a baseline, summarization and three different fusion techniques. The author found that all three fusion methods outperformed the clipping and summarization methods[16] with the Max Fusion having the highest Precision@1 and Precision@3 at the MS Marco dataset.

No other work was found that experimented with different techniques and reported performance results and none was found for the ad hoc retrieval task. As the result of the literature review, four categories of strategy were found in twelve papers. The classification of techniques followed Crijns's work whenever possible. When not existing, names were freely given. The summary of the numbers of papers for each identified category can be found in Table 1.

For each category, the techniques are presented separately for each paper.

2.2.1 Rank and Fusion

Rank and Fusion Techniques were the most used class of techniques showing up in five of the twelve papers reviewed. Different from Crijns'[16] work, none presented results for different fusion techniques.

2.2.1.1 Simple Applications of BERT for Ad Hoc Document Retrieval[1]

This paper by Yang et al, if the input to BERT is longer than 512 token, at he sentences are split into fixed sized chunks and inference is applied on chunks individually, later aggregating sentence scores to produce document scores. Across the 250 topics, each document averaged 43 sentences, with 27 tokens per sentence. It is not clear how this aggregation is made.

The author understand they sidestepped the issue of not having sentence-level relevance judgments, although there are some obvious distant supervision techniques to “project” relevance labels down to the sentence level that should be explored.

2.2.1.2 Applying BERT to Document Retrieval with Birch[2]

For this work, inference is applied over each sentence in a candidate document and sentence-level evidence is aggregated for ranking documents as follows:

$$Sf = a.Sdoc + (1 - a).\sum_{i=1}^{n} wi.Si$$

Where Sdoc is the original document score and Si is the i-th top-scoring sentence according to BERT; a and wi’s are parameters that need to be learned. In practice, only up to the three top-scoring sentences in each document are considered. The intuition behind this approach comes from Zhang et al[17], who found that the “best” sentence or paragraph in a document provides a good proxy for document relevance.

A sentence-level dataset is created on the fly from the initial ranking by splitting each document into its constituent sentences. Each sentence is fed into their BERT model to obtain a relevance score. These relevance scores are then aggregated with document scores to rank the candidate documents.

2.2.1.3 End-to-End Open-Domain Question Answering with BERTserini[3]

The architecture of BERTserini is comprised of two modules, the Anserini retriever and the BERT reader. The retriever is responsible for selecting segments of text that contain the answer, which is then passed to the reader to identify an answer span. At inference time, for retrieved articles, BERT reader is applied paragraph by paragraph. The reader selects the best text span and provides a score. The reader score is combined with the retriever score via linear interpolation:

2.2.1.4 Exploring Classic and Neural Lexical Translation Models for Information Retrieval: Interpretability, Effectiveness, and Efficiency Benefits

The rank and fusion applied by this paper is as follows:

1. Splits longer documents d into m chunks
2. generates m token sequences [CLS] q [SEP] di [SEP]; – processes each sequence with BERT to generate contextualized embeddings for regular tokens as well as for [CLS].
3. The outcome of this procedure is m [CLS]-vectors $clsi$ and n contextualized vectors $w1w2...wn$: one for each document token ti.

The authors don't make clear how they cobmine the contextualized vectors.

2.2.2 Normalized Datasets

When training ranking systems, longer documents tend to have higher scores, thus creating a systematic bias against shorter documents[18]. For that reason, some datasets normalize the input length and provide inputs shorter than BERT's limit. When using such datasets, the underlying assumption is that a relevant passage is a proxy for a relevant document.

MS Marco is a large scale family of datasets focused on machine reading comprehension, question answering, and passage ranking. For information retrieval, MS Marco offers full document ranking and passage ranking data. The passage ranking dataset was assembled from the document collection. For every full document, editors annotated the relevant passages in a very labourers process inviable for inference time.

CEDR: Contextualized Embeddings for Document Ranking[19] use the normalized datasets TREC Robust 2004 and WebTrack 2012–14. If an input is longer than 512, it is split as evenly as possible, what happened only with less than 1% of the inputs.

Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval[20] uses the MS MARCO and TREC-CAR datasets and do not explicit if they had to deal with inputs longer than 512.

2.2.3 Document precomputation

2.2.3.1 TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval[4]

Many models compute query and document embeddings together by concatenating them with the [SEP] token. In this paper, query and document are embedded separately, which allows for document precomputation. BERT embeddings are combinations of three components: token embeddings, segment embeddings and position embeddings. While, the input of a TwinBERT encoder only contains one single sentence and segment embeddings are unnecessary. Therefore, the input embeddings only consist of the sum of token embeddings and position.

2.2.3.2 Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching[5]

This paper introduces the SMITH model. For the pre-computation, the input document is first split into several sentence blocks, which may contain one or more sentences using their proposed greedy sentence filling method. When the input text becomes long, both relations between words in a sentence block and relations between sentence blocks within a document becomes important for content understanding. Therefore, the authors mask both randomly selected words and sentence blocks during model pre-training. The sum of the masked sentence block prediction loss and the masked word prediction loss is the final SMITH model pre-training loss.

2.2.4 Clipping

Clipping was found in two papers from 2019 and was not seen on more recent papers. This method sacrifices the possibility that the distant tokens "pay attention" to each other, which becomes the bottleneck for BERT to show its efficacy in complex tasks[11].

In the Multi-Stage Document Ranking with BERT[21] work, the authors truncate the query, candidates *di* and *dj* to 62, 223, and 223 tokens, respectively, so the entire sequence will have at most 512 tokens when concatenated with the [CLS] token and the three separator tokens. Using the above truncation limits, in the datasets used in this work none of the queries are truncated, and less than 1% of the documents are truncated.

In the Passage reranking with BERT[22] work, the query is truncated to have at most 64 tokens. The passage text is also truncated such that the concatenation of query, passage, and separator tokens have the maximum length of 512 tokens.

2.2.5 Summarization

There are two basic types of single document summarization: the extractive and abstractive approach. An extract summary is generated by selecting a few relevant sentences from the original document. Summary's length depends on the compression rate[23]. There a few ways to choose which sentences to extract.

In an abstractive summarization, the ideia is to capture high level concepts and ideas, rather than the original sentences. It requires extensive use of natural language understanding techniques [23] and for the potential added latency, could be considered as a form of document preprocessing.

Summarization was found only at Crijns's work. Whenever the input was too long, the context was summarized (last conversational iterations). It was done by sampling a number of important words, extracted from the context, based on their TF-IDF scores so that the context would fit within the input restrictions. The author reports that summarization improved the results over the clipping baseline, but underperformed in comparison to Score Fusion techniques. The author's hypothesis is that the TF-IDF extraction removes the word from its surrounding context, affecting Bert potential for understanding exactly this type of information.

2.3 State Of The Art in Neural IR

The Bidirectional Encoder Representations from Transformers (BERT) is a language model proposed by Google in 2018 and, since then, has made a significant impact in the fields of natural language processing and information retrieval. The debut of BERT in ranking was made in 2019 by Nogueira and Cho (2019)[22] beating the state-of-the-art IRNet model with an MRR@10 of 0.359 against 0.281 on the MSMarco dataset[18].

Since then, much research is being made in order to expand, address shortcomings and test BERT-based architectures for ranking in both multistage and single stage text retrieval. In the single stage setup, the gains of using BERT-based representation learning potential is to get past the sparse representations of candidate generation and the vocabulary mismatch problem.

The main limitation of this intent is the cost in computing the representation at serving time. ColBERT[15] introduces a late interaction architecture achieving competitive ranking performance two orders of magnitude faster than vanilla BERT model in a end-to-end ranking setup. Colbert was trained in the MSMarco Passage Ranking dataset, composed of paragraph-length passages that fit into the BERT input limitation of tokens.

In the MS Marco dataset, full ranking task, the Approximate Nearest Neighbor Negative Contrastive Learning (ANCE)[24] technique is largely used and holds the first two position in the leaderboard (MRR@100 eval). The BERT based submission "BERT-m1 base (v4) / enriched traditional"(feb/2021) held the seventh position as of this writing with a MRR@100 of 0.408.

3 Methodology

From the techniques presented in the background section, the rank and fusion was found to be the most promising in terms of scoring in the context of question answering and would be a good candidate to test on the context of document retrieving. These technique, however, has a very high impact on latency since each document is split in many different ones even at inference time.

Two aspects posed a problem to experiment with rank and score techniques: 1- the experiments made with the clipping technique were already above the available infrastructure limits, so the added latency would not allow for a feasible experimental setup and 2- the implementation details are not available. Given these obstacles, experimentation with rank and fusion are left to future studies. Document precomputation techniques would impose the same problems as rank and fusion.

We chose to experiment with the summarization technique. The dissertation that presented the technique used TF-IDF to select the most salient terms. But this technique removes words from the context, and context is what BERT is trying to learn. The author found that this technique did not perform well. Th prsetn work implements two different extractive summarization techniques, further detailed. The baseline was the TF-IDF summarization technique. Beyond TF-IDF, experiments with the clipping method were also performed for comparison. The clipping method, being the most popular, was used as a baseline for other experiments.

3.1 Summarization Techniques

Two extractive summarizations were tested: fixed extraction and LSA summarization. Crijin used TF-IDF in order to select the most representative terms, but found that removing words from its surrounding context could be diminishing Bert's capacity to read from context and for that reason, was not used in the present work. Instead, we chose techniques that could keep such context. Abstractive summarization was also not tested. The need for deep neural language understanding would characterize it as document precomputation; out of the scope of this work.

Fixed extraction consists in extracting a fixed number of equally spaced text blocks. Four blocks were extracted starting at the beginning of the document. This method was not found in any papers and its a heuristic that does not depend on knowledge of the content. The size of the block takes into account the remainder space left after url+title concatenation. The extraction is made before tokenization so the space left is calculated with a reduction of 25%, to give room to tokens.

LSA summarization is an unsupervised method that applies several learning techniques to identify salient sentences while avoiding semantic repetition. The implementation was based on the luisfredgs implementation, available at https://github.com/luisfredgs/LSA-Text-Summarization.

The first step in LSA summarization is assembling a sparse matrix of words per sentence. There a few ways to fill the cell. We used the simple frequency of words. The second step is applying singular value decomposition to reduce noise and help identify semantic similar sentences. The last part is selecting the sentences, which is a hiperparameter. For this work, the 50 most salient sentences were retrieved for the summary.

In both chosen techniques it is possible to guarantee the final length of the tokens vector and clipping anything above 512 is always used when needed.

3.2 MS Marco Dataset

The other technique, normalized datasets, was not considered feasible for a real inference setup, So the experiments were made in a non-normalized dataset: Document ranking MS Marco. This dataset was chosen for being one of the few that provides the whole document, instead of just passages and of those, is the most used in the papers reviewed. This dataset is composed of three main parts:

- 3.213.835 documents, composed of id, url, title and body
- 367.013 queries, composed of id and body
- QRels, or query relevance judgments. For each of the 367.013 queries, a single document is annotated as relevant

Alongside with msmarco files, a candidates datset was used: HDCT files, composed of 366.000 queries with 1.000 candidate documents for each. In the re-ranking setup, an initial algorithm (hdct in this case) is used to generate candidates for the next phase, the re-ranking. HDCT was used instead of the popular BM25 given that it was the one used with the selected architecture. We were aiming to isolate the document representation effects, so that was left untouched.

The HDCT dataset is divided in 183 partial files, each containing 2.000 queries and respectively candidate documents. When trained with all the queries, total ti, from preprocessing to final evaluation was over 140h using Google Colab Pro GPUs. Unfortunately, Google Colab Pro only allow for 24h of training with a cool down of another 24h if the limit is reached. To avoid reaching the limit and allow for more experimentation, the implementations were tested with at most 10 files. Experiments were run with 1, 5 and 10 files, to compare latency as input grows.

3.3 Bert model - Reranker

The architecture was selected from the MSmarco document ranking leader board from top down in terms of score. The candidate should be Bert-based, have an open source implementation of its code and preferably, a paper. The highest ranking architecture at the time was Ance, from Microsoft, which has an open source implementation. Unfortunately the technique used by Ance is to select negative samples from the entire dataset instead of just from a small batch. MSmarco dataset is huge and explodes in size once in memory. In all setups tested, the machine ran out of memory.

The second candidate was Prop, that appears several times in the leader board in different configurations. The open source code was not under maintenance and was missing key scripts to work properly. Attempts at contacting the author showed no results. Finally, the Reranker code was implemented with the Hugging Face library, making use of more familiar patterns and APIs. The model that appears on the leader board is published in the library's model zoo. The code is production grade and is ready to be used with GPU. Some additional setup was needed for it to be fully operational and to receive the interventions.

3.4 Experimental Setup

The preprocessing of the files consisted in tokenizing both the positive example and ten negative examples to be presented to the model in training time. In the clipping experiments, the fields url, title and body were concatenated and any token above 512 was clipped. The tokenization was done with bert-base-uncased tokenizer class as implemented in the Hugging Face library. All experiments were run with 5 different seeds to allow for confidence interval calculations.

The experiments were evaluated with MRR@100 score, like the msmarco leaderboard, and was calculated with the official Microsoft script. The infrastructure used was Google Colab Pro with Tesla T4 GPU, 27GB RAM and 200GB Google Drive storage.

The resulting code of this work is available at https://github.com/Valerieps/Preprocessing-techniques-with-Reranker

4 Results

Table 2 shows results comparing MRR@100 for all four techniques tested. TF-IDF and fixed summarization showed no statistical difference between the scores. LSA performed significantly better than both summarization techniques but showed no statistical difference to clipping. TF-IDF performed worse than clipping, a different result than what happened in the question answering scenario in Crijin's work.

Technique	MRR@100
Clipping	0,0915 ± 0,0054
TF-IDF	0,0356 ± 0,0118
Fixed	0,0531 ± 0,0059
LSA	0,0848 ± 0,0071

Tabela 2 – MRR@100 compared to TF-IDF

As for the inference times, shown in table 3, TF-IDF was the fastest of all four techniques tested. Fixed and LSA summarization and clipping didn't present statistical or noticeable differences in seconds per query at inference time.

A search at Google engine for five different queries returned results in 0,6575 ± 0,0984 second, so it seems that TF-IDF runtime is within what could be considered acceptable. The other techniques are above this threshold but could also be perceived as a satisfactory result by a human.

Technique	seconds/query
Clipping	0,9846± 0,0055
TF-IDF	0,6654 ± 0,0000
Fixed	0,9859 ± 0,0000
LSA	0,9866 ± 0,0000

Tabela 3 – Inference times compared to TF-IDF

5 Conclusion

TF-IDF, despite being the fastest technique, did not perform better as in the question answering scenario, but LSA summarization tied with clipping in terms of MRR@100 and inference latency. Given that clipping is an easy to implement technique it would be the case to choose this technique over the others in the document ranking task.

Techniques of abstractive summarization that leverage neural networks could potentially give better results than clipping. They are know to be expansive to train, but don't need the query and could be performed offline. In this case, inference latency would be competitive. Future experiments could test this family of techniques.

It is important to remember that due to extremely long training times, it was not possible to train with the whole dataset, so further experimentation also should make room for longer training times.

Referências

[1] YANG, W.; ZHANG, H.; LIN, J. Simple applications of bert for ad hoc document retrieval. *arXiv*, 2019. ISSN 23318422.

[2] YILMAZ, Z. A. et al. Applying BERT to document retrieval with birch. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Proceedings of System Demonstrations*, p. 19–24, 2020.

[3] YANG, W. et al. End-to-end open-domain question answering with BERTserini. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Demonstrations Session*, p. 72–77, 2019.

[4] LU, W.; JIAO, J.; ZHANG, R. TwinBERT: Distilling Knowledge to Twin-Structured Compressed BERT Models for Large-Scale Retrieval. *International Conference on Information and Knowledge Management, Proceedings*, p. 2645–2652, 2020.

[5] YANG, L. et al. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. *International Conference on Information and Knowledge Management, Proceedings*, p. 1725–1734, 2020.

[6] MITRA, B.; CRASWELL, N. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, v. 13, n. 1, p. 1–129, 2018. ISSN 15540677.

[7] ZHAO, L.; CALLAN, J. Term necessity prediction. In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. [S.l.: s.n.], 2010. p. 259–268.

[8] MITRA, B.; CRASWELL, N. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval*, v. 13, n. 1, p. 1–126, 2018. Disponível em: <https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/>.

[9] ZHAO, L.; CARBONELL, J.; YANG, Y. *Modeling and Solving Term Mismatch for Full-Text Retrieval*. 2012.

[10] DEVLIN, J. et al. 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). Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://www.aclweb.org/anthology/N19-1423>.

[11] DING, M. et al. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, v. 33, 2020.

[12] ROBERTSON, S.; ZARAGOZA, H. *The probabilistic relevance framework: BM25 and beyond*. [S.l.: s.n.], 2009. v. 3. 333–389 p. ISSN 15540669. ISBN 1500000019.

[13] GUO, J. et al. A Deep Look into neural ranking models for information retrieval. *Information Processing and Management*, Elsevier Ltd, 2019. ISSN 03064573. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0306457319302390>.

[14] GAO, L.; DAI, Z.; CALLAN, J. Rethink Training of BERT Rerankers in Multi-Stage Retrieval Pipeline. p. 1–8, 2021. Disponível em: <http://arxiv.org/abs/2101.08751>.

[15] KHATTAB, O.; ZAHARIA, M. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT*. 2020.

[16] CRIJNS, T.; VRIES, A. de. *Have a chat with BERT; passage re-ranking using conversational context*. Tese (Doutorado) — Master's thesis, Radboud University (Aug 2019), https://www. ru. nl/publish ..., 2019.

[17] ZHANG, H. et al. Evaluating sentence-level relevance feedback for high-recall information retrieval. *CoRR*, abs/1803.08988, 2018. Disponível em: <http://arxiv.org/abs/1803.08988>.

[18] LIN, J.; NOGUEIRA, R.; YATES, A. *Pretrained Transformers for Text Ranking: BERT and Beyond*. 2020.

[19] MACAVANEY, S. et al. CEDR. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, Jul 2019. Disponível em: <http://dx.doi.org/10.1145/3331184.3331317>.

[20] DAI, Z.; CALLAN, J. *Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval*. 2019.

[21] NOGUEIRA, R. et al. Multi-stage document ranking with BERT. *arXiv*, 2019. ISSN 23318422.

[22] NOGUEIRA, R.; CHO, K. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. Disponível em: <http://arxiv.org/abs/1901.04085>.

[23] GAMBHIR, M.; GUPTA, V. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, Springer Netherlands, v. 47, n. 1, p. 1–66, 2017. ISSN 15737462.

[24] XIONG, L. et al. *Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval*. 2020.

[25] NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, Taylor Francis, v. 23, n. 3, p. 153–163, 2004. Disponível em: <https://doi.org/10.1080/01449290410001669914>.

[26] BOYTSOV, L.; KOLTER, Z. *Exploring Classic and Neural Lexical Translation Models for Information Retrieval: Interpretability, Effectiveness, and Efficiency Benefits*. 2021.

[27] LIN, J.; NOGUEIRA, R.; YATES, A. *Pretrained Transformers for Text Ranking: BERT and Beyond*. 2020.