International Journal of Data Warehousing and Mining
Volume 18 • Issue 2

Efficient Open Domain Question Answering With Delayed Attention in Transformer-Based Models

Wissam Siblini, Worldline, France*

Mohamed Challal, Worldline, France

https://orcid.org/0000-0003-0558-8651

Charlotte Pasqual, Worldline, France

ABSTRACT

Open domain question answering (ODQA) on a large-scale corpus of documents (e.g., Wikipedia) is a key challenge in computer science. Although transformer-based language models such as Bert have shown an ability to outperform humans to extract answers from small pre-selected passages of text, they suffer from their high complexity if the search space is much larger. The most common way to deal with this problem is to add a preliminary information retrieval step to strongly filter the corpus and keep only the relevant passages. In this article, the authors consider a more direct and complementary solution that consists of restricting the attention mechanism in transformer-based models to allow a more efficient management of computations. The resulting variants are competitive with the original models on the extractive task and allow, in the ODQA setting, a significant acceleration of predictions and sometimes even an improvement in the quality of response.

KEYWORDS

Bert, Deep Learning, Information Retrieval, Knowledge Management, Natural Language Processing, Question Answering, Scalability, Speed, Squad, Transformer

INTRODUCTION

The last few years have given rise to many disruptive innovations in the field of Natural Language Processing (NLP) and allowed a significant improvement of evaluation metrics on public benchmarks (Wolf et al., 2019). In particular, the proposal of a novel architecture called the *Transformer* (Vaswani et al., 2017) and an adaptation into the versatile easy-to-use language model Bert (Devlin, Chang, Lee, & Toutanova, 2019) have led to a series of publications generating a continual enthusiasm. Recent transformer-based models such as RoBerta (Liu et al., 2019), XLNet (Z. Yang et al., 2019), Albert (Lan et al., 2019) managed to outperform humans on difficult benchmarks for general language comprehension assessment. This exploit led to the democratization of their use in many applications. We here focus on automatic question answering where we search for the answer of a user question in a large set of text documents (e.g. the entire English Wikipedia with millions of articles). Language models have been proven efficient on a sub-task called extractive Question Answering (eQA), sometimes also referred to as Reading Comprehension (RC), on the reference dataset SQuAD (Rajpurkar, Zhang, Lopyrev, & Liang, 2016): given a question-document pair, the goal is to find the

DOI: 10.4018/IJDWM.298005

*Corresponding Author

answer within the document. But on our target task, referred to as Open Domain Question Answering (ODQA), the problem is more complex because for each question the search space is much larger. Since Transformer-based readers already require a non-negligible time to process a single question-paragraph pair, they cannot manage millions in real-time. The most common solution is to combine eQA with Information Retrieval (IR) (Manning, Schütze, & Raghavan, 2008) to first select p relevant documents and only apply the costly reading comprehension model on them. Such a combination has proven itself in BertSerini (W. Yang, Xie, et al., 2019) where the widely known Lucene with BM25 (Białecki, Muir, Ingersoll, & Imagination, 2012; Robertson et al.,1995) for the IR part was combined with Bert for the eQA part.

In this paper, we propose to tackle the time issue from a more direct and complementary angle which consists in using partial attention in the eQA model so that many computations can be saved or only done once as a preprocessing. More precisely, our contributions are the following: **(1)** We use a Delaying Interaction Layers mechanism (DIL) on transformer-based models that consists in applying the attention between subparts (segments) of the input sequence only in the last blocks of the architecture. We implement this mechanism for both Bert and Albert and refer to the variants as DilBert and DilAlbert. **(2)** We study their behavior in the standard eQA setting and show that they are both competitive with the base models. **(3)** We analyze the impact of delayed interaction on the models complexity and then empirically confirm that in the ODQA setting, it allows to speed up computations by an order of magnitude on either GPU or CPU. **(4)** Finally, we evaluate the models on the reference ODQA dataset OpenSQuAD (Chen, Fisch,Weston, & Bordes, 2017) by combining them with Answerini as W. Yang, Xie, et al. (2019). Although DilBert (resp. DilAlbert) performs slightly worse than Bert (resp. Albert) when faced to a single relevant passage (eQA), it can outperform it in the ODQA setting when having to select the right answer within several paragraphs.

Additionally, our code is made available with the paper[1] to (i) allow the reproduction of all the paper results and (ii) encourage new proposals by offering an ODQA pipeline similar to BertSerini and scripts to test it interactively on Wikipedia or evaluate it on OpenSQuAD.

Background

For a long time, researchers have been interested in Automatic Question Answering (Woods & WA, 1977) to build intelligent search engines and browse large-scale unstructured documents databases such as Wikipedia (Chen et al., 2017; J. Lee, Yun, Kim, Ko, & Kang, 2018; Ryu, Jang, & Kim, 2014; S. Wang et al., 2017; W. Yang, Xie, et al., 2019). This kind of system is often designed with several layers that successively select a smaller but more precise piece of text. It generally starts with a one-stage or a multiple-stage *Information Retrieval* step (Ad-hoc retrieval) that identifies relevant documents for the question at hand. Then, an algorithm designed for *eQA* is applied to identify the answer spans within the selected passages.

Ad-hoc Retrieval

Given a user query, searching for relevant items within a large set of documents generally consists in: (i) applying an encoding model to all documents, (ii) encoding the query as well, (iii) applying a ranker that produces a relevance score for each query-document pair based on their encodings, and finally (iv) sorting the documents accordingly (Manning et al., 2008). For the encoding part, proposals go from vocabulary and frequency based strategies such as bag-of-words or TF-IDF (Baeza-Yates, Ribeiro-Neto, et al., 1999), to word2vec embeddings averaging (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) to even strongly contextualized embeddings with Bert (Devlin et al., 2019; MacAvaney, Yates, Cohan, & Goharian, 2019; W. Yang, Zhang, & Lin, 2019). For the relevance score part, the most popular choice is a fixed similarity/distance measure (cosine or Euclidean) but researchers have explored others strategies such as siamese networks (Das, Yenala, Chinnakotla, & Shrivastava, 2016), histograms (Guo, Fan, Ai, & Croft, 2016), convolutional networks (Dai, Xiong, Callan, & Liu, 2018), etc. In the recent literature (MacAvaney et al., 2019), several combinations between contextualized embedding techniques (Elmo and Bert) and rankers (DRMM, KNRM, PACRR) have been tried. By fine-tuning BERT for IR and exploiting the representation of its [CLS] token, the authors obtained state-of-the-art results on several reference IR benchmarks.

But there are also old and proven IR baselines like BM25 (Robertson et al., 1995), a weighted similarity function based on TF-IDF statistics (frequencies of the question words in the documents, and their inverse frequencies in the overall corpus), which until today remains very competitive with neural methods (Lin, 2019). BM stands for Best Matching and the regular BM25 is extremely fast and benefits from decades of fine engineering, for instance in its implementation in Lucene (Białecki et al., 2012; P. Yang, Fang, & Lin, 2017, 2018).

Extractive Question Answering

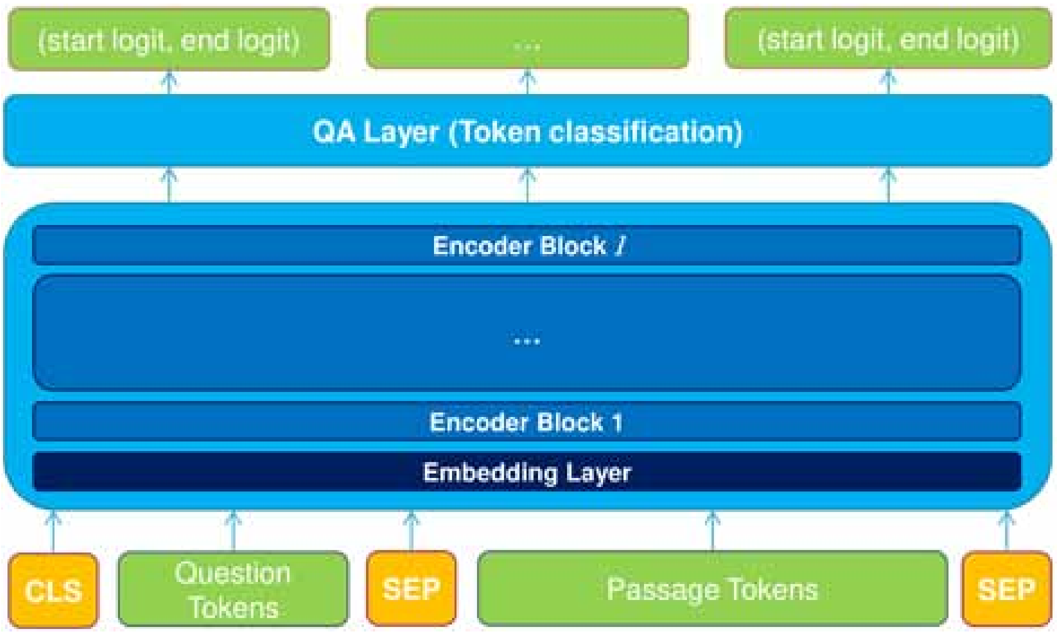
The extractive Question Answering task consists in identifying a question's answer as a text span within a rather small passage. The most popular dataset for this task is the Stanford Question Answering Dataset (SQuAD) (Rajpurkar, Zhang, Lopyrev, & Liang, 2016) which consists in more than a hundred thousand questions, each paired with a Wikipedia article paragraph. The state of art algorithms for this task are transformer-based language models such as Bert (Devlin, Chang, Lee, & Toutanova, 2019) or Albert (Lan et al., 2019). They are usually composed of an input embedding layer, followed by a succession of l encoder blocks (which implement, in particular, a self-attention mechanism) and finally an output layer.

Self-attention and Bert Bert is built from the encoder part of the Transformer architecture (Vaswani et al., 2017). Since its proposal, the Transformer has quickly taken over the previously mainstream recurrent neural models in NLP. It is an encoder-decoder model that works essentially with self-attention and attention rather than recurrent units. The advantage is that the hidden states in a layer can be computed in parallel and not sequentially. Moreover, each state depends on a dynamically chosen set of states from the previous layer (through attention scores), allowing to deal efficiently with very long sequences. In order to explain how Bert works, let us describe in detail the encoder part of the Transformer. It is designed to take as input a sequence of items (e.g. words). Each item has an index (with respect to a global vocabulary) and a position that the model transforms into an initial global vector (embedding layer). The sequence of initial embeddings then goes successively through several identical blocks (encoder blocks) that implement two steps: a self-attention step and a step with a feed-forward layer applied element-wise on the sequence. Each step is followed by a residual connection. The self-attention mechanism consists of computing, for each item of the sequence, a new representation that is a weighted sum of the item itself (projected with a learned "value" matrix V) and the other items of the sequence (also projected with the "value" matrix V). The weight of each item is called the attention score and is computed from a similarity between the considered item (projected with a learned "query" matrix Q) and the others (projected with a learned "key" matrix K). To be even more precise, the self-attention is multi-headed in the Transformer. It means that multiple self-attention layers are applied to the sequence in parallel and the results are concatenated. For each head, the "query", "key" and "value" matrices are different. More details with explanatory figures can be found here: https://jalammar.github.io/illustrated-transformer/.

Bert uses the Transformer's encoder architecture and considers, as input, a sequence of words/tokens that are the concatenation of two subsequences (called segments) with special words as separators between them. Its embedding layer embeds the tokens' ids, their positions, and, in addition, their segment ids (0 or 1). This layer is followed by the sequence of encoder blocks. Bert can be used as a general language model, which means that it has been pre-trained by its authors on general self-supervised tasks with a large amount of text, to be later adapted and transferred to several specific NLP tasks from text classification to question answering. The pre-training tasks are "masked language" (trying to predict masked words from an input sentence) and "next sentence prediction" (deciding if the sentences from the two segments are two subsequent sentences from a global text or not). In the following paragraph, we explain how Bert can be used for the extractive question answering task after pre-training.

Bert for extractive question answering (Figure 1) On the eQA task, Bert takes as input a text sequence that is the concatenation of the question (first segment) and the associated passage (second segment). The whole sequence is tokenized and a special [CLS] (resp. [SEP]) token is added at the beginning (resp. between the question and the passage and at the end). Then, the associated features (token ids, segments, positions...) are fed to the model. The embedding layer produces a first sequence of embeddings which goes through the l encoder blocks to finally turn into a sequence of contextualized vectors. Then, each final vector of the passage's tokens go through a feed-forward classification layer that predicts two probabilities, to be the beginning and the end of the answer span.

Figure 1. Architecture of Bert-like models (l blocks) applied to eQA. Start/end logits inform on the probability for a token subsequence to be the answer.



Open Domain Question Answering

Open Domain Question answering solutions generally combine a retriever that solves the Ad-hoc Retrieval task by selecting relevant documents and a reader that solves the extractive Question Answering task on the selected passages. An emblematic example is DrQA (Chen, Fisch,Weston, & Bordes, 2017), a bot developed at Facebook which is able to answer to questions by searching in the entire English Wikipedia in real-time. It consists in a TF-IDF + cosine retriever that selects 5 relevant documents followed by a multi-layer RNN reader (Chen, Bolton, & Manning, 2016). Later, other proposals were able to surpass its quality of answer with a paragraph reranker (J. Lee et al., 2018; S. Wang et al., 2017) or with a "minimal" retriever preselecting small but relevant portions of text (Min, Zhong, Socher, & Xiong, 2018). In the meantime, Bert was released and W. Yang, Xie, et al.(2019) proposed a first successful usage for ODQA, by simply combining answerini/Lucene with a Bert base fine-tuned on SQuAD, which achieved the new state-of-the art performance on the reference benchmark OpenSQuAD (Chen, Fisch, Weston, & Bordes, 2017). Since then, a few proposals (Feldman & El-Yaniv, 2019; K. Lee, Chang, & Toutanova, 2019; Ren, Cheng, & Su, 2020; Z. Wang, Ng, Ma, Nallapati, & Xiang, 2019) have been focusing on better selection of passages and a better pooling of answers from them.

In all of these proposals, the scaling issue is always tackled from the retriever's perspective. Yet, the ODQA setting offers many opportunities to address it from the reader's side, which is the main motivation behind this work.

Main FOCUS OF the ARTICLE

Delaying Interaction Layers

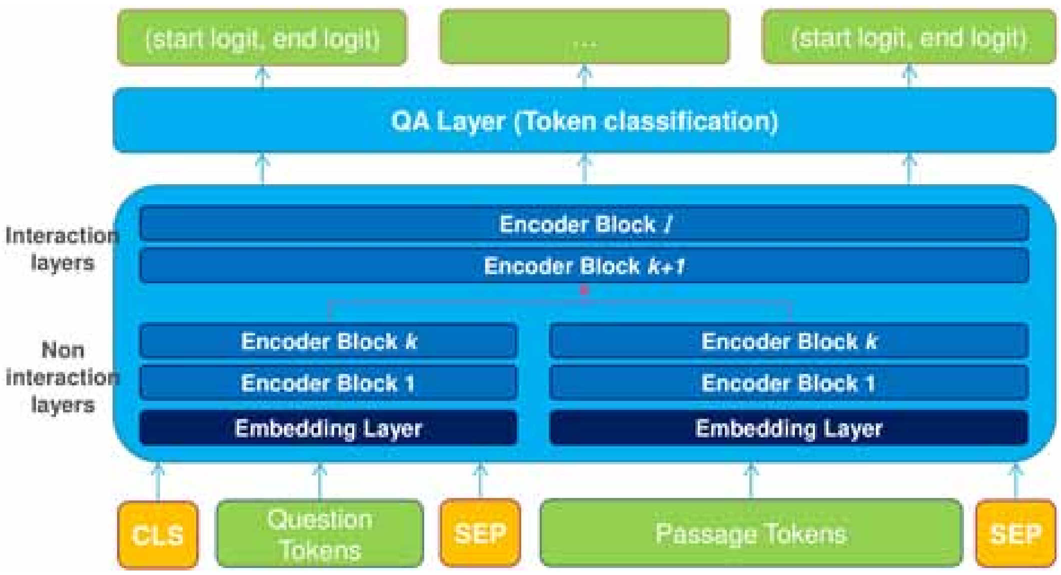
In the ODQA setting, the set of documents is rather static. If q questions are asked and there are p documents to search the answer within, transformer-based readers will consider all the $q \times p$ possible pairs, split them into fixed-length examples and make predictions on them. Not only this is very costly but all computations are done interactively. More precisely, in the encoder blocks, the attention mechanism makes the representation of the documents tokens dependent on the question tokens, so there is no possibility to pre-compute intermediate representations of the database's documents without knowing the question beforehand. To better face this setting, we consider a generic modification of the architecture of existing models to allow reducing computations and make preprocessing possible. The key is to "delay" the interaction between question and paragraph so that a part of the computations can be done independently. Also, we need to make changes (i) that are not specific to a given model but that can be applied to a maximum of transformer-based language models and (ii) that do not introduce new parameters so that we can benefit from existing pre-trained weights.

The principle of the mechanism is shown schematically in Figure 2. We consider an entry split between the two segments (after the first separator [SEP]). Let us denote by $s_q \in \mathbb{R}^{n_q \times d_e}$ (resp $s_p \in \mathbb{R}^{n_p \times d_e}$) the question segment (resp. the paragraph segment) after the embedding layer, with n_q being the size (number of tokens) of the question comprising the token [CLS] and the first token [SEP], n_p the size of the passage including the last token [SEP], and d_e the dimension of the embedding space (often 768 or 512). Let us also denote by E_j the j-th encoder block, parameterized by a set of weights θ_j. We apply the k first encoder blocks independently on the two segments to obtain $s_q^{'} = E_k \circ \ldots \circ E_1 \left(s_q\right)$ and $s_p^{'} = E_k \circ .. \circ E_1 \left(s_p\right)$. We then concatenate them into $s' = concat\left(s_q^{'}, s_p^{'}\right)$ before applying the $l-k$ last encoder blocks to get $s'' = E_l \circ .. \circ E_{k+1}\left(s'\right)$ which ultimately goes through the output layer ("QA Layer"). We refer to the first k blocks as the non interaction blocks and the last $l-k$ as the interaction blocks. The impact of the hyperparameter k is studied in the experimental section. Note that the weights θ_k, ..., θ_l involved in the computation of $s_q^{'}$ are the same as those involved in the computation of $s_p^{'}$ and in the computation of $E_k \circ \ldots \circ E_1 \left(concat\left(s_q, s_p\right)\right)$ in the original transformer model. We initialize these weights with the pre-trained models. Then, we can make the weights applied to the question evolve independently of those applied to the paragraph but we choose to share them instead as it keeps the number of weights equal to the original model. We implement the described mechanism in Python on the basis of the original models implementation from the *transformers* python library, version 2.7.0 (Wolf et al., 2019).

Delayed interaction reduces the complexity of the first k blocks. More precisely, self-attention requires a number of computations proportional to n_s^2 where $n_s = n_q + n_p$ is the input sequence length. Therefore, the implemented mechanism reduces the complexity from $O\left(n_q^2 + n_p^2 + 2n_p \times n_q\right)$

to $O\left(n_q^2+n_p^2\right)$. Although it already makes a difference, it can be negligible in practice, because the paragraph is much longer than the question, and because the reduced intra-block operations are in fact highly parallelizable. The difference that matters takes place in the ODQA setting. Consider that a single encoder block has a forward complexity of $C \times n_s^2$. Therefore, the whole encoder has a complexity of $l \times C \times n_s^2$ to process a single example, and $l \times C \times q \times p \times n_s^2$ to process a set of q questions and p paragraphs (for the sake of simplicity, assume that paragraphs are cut in such a way that question-paragraph pairs fit in the model input size n_s). With delayed interaction, computations in the first k blocks for each question (resp. paragraph) do not have to be redone for each paragraph (resp. question). Therefore, the complexity becomes $k \times C \times \left(q \times n_q^2 + p \times n_p^2\right) + \left(l-k\right) \times C \times q \times p \times n_s^2$. Thus with p and q in the order of 10^2, the quadratic term $p \times q$ largely dominates and the speedup tends to the ratio $\frac{l-k}{l}$ between the number of interaction blocks and the total number of blocks in the original model. Note that the independent paragraph processing (of complexity $k \times C \times p \times n_p^2$) can be done at initialization in the ODQA setting, instead of interactively when users ask questions.

Figure 2. Delayed interaction in Bert-like eQA models: the first k (resp. the last $l-k$ blocks are applied to question and paragraph separately (resp. to the whole).



Experimental Study

In our paper, the goal is to ultimately assess the interest of delayed interaction in Open Domain Question Answering by studying its performance on the reference dataset OpenSQuAD as W. Yang, Xie, et al.(2019). We start by analyzing the associated eQA sub-task (SQuAD v1.1, Rajpurkar et al. (2016)), in particular to know how the predictive performance is impacted with respect to the hyperparameter k. Then, on the ODQA task, we measure the speedup and then the predictive performance.

We consider two language models as baselines. The first one is Bert since it is the most used[2] and also the one with the largest set of different pre-trained weights. The second one, to test our framework in a challenging way, is Albert because it has the specificity of using the same weights in all of its l encoder blocks (the weights θ_j are the same for all j, $j \in [\![1,k]\!]$ **and** $j \in [\![k+1,l]\!]$) and we want to know if this leads to an unexpected behavior or not. We refer to the variants as DilBert (resp. DilAlbert) which stands for "Delaying Interaction Layers" in Bert (resp. Albert).

Extractive QA

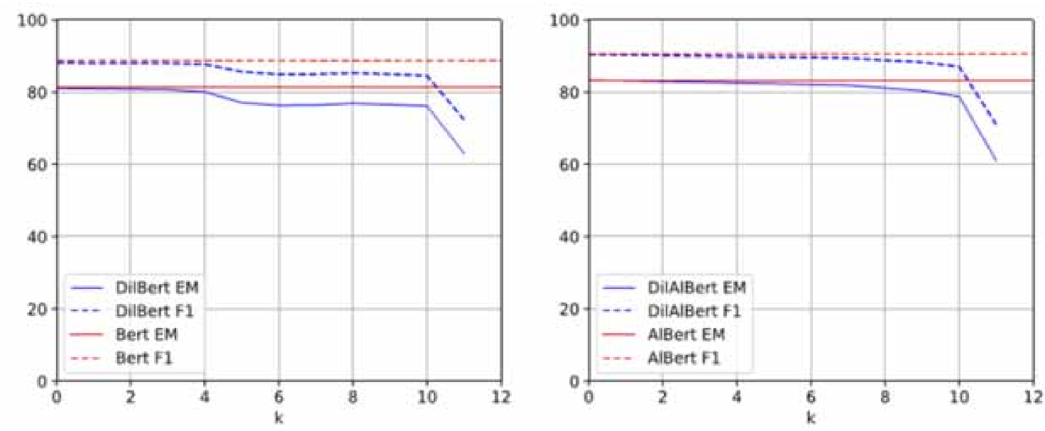
For the eQA task, we train Bert, DilBert, Albert and DilAlbert on the SQuaD v1.1 train set. We consider the base version of each model and we initialize them with the English pre-trained weights: *bert-based-uncased* for Bert and DilBert and *albert-base-v2* for Albert and DilAlbert.

The training is carried with the *run_squad.py* script from the *transformer* library using the default hyperparameters: 2 epochs, an input sequence length n_s of 384, a batch size of 12, and a learning rate of 3e-5. The models are evaluated on the SQuAD v1.1 dev set using the official metrics for the task (Exact Match and F1-score). For DilBert and DilAlbert, we make the hyperparameter k vary from 0 to $l-1$ and analyze its impacts on the performance (Figure 3).

As a sanity check, we can observe that without delayed interaction ($k=0$), our implementation of DilBert (resp. DilAlbert) provides the same results as Bert (resp. Albert). Then, as k increases, the performance remains very competitive and it only decreases significantly for $k=11=l-1$.

For $k=l$, there is no interaction anymore so start logits and end logits associated to paragraph tokens do not depend on the question and the performance dramatically drops to an exact match of around 13 (not reported in the Figure 3). For information, results of Dil variants with k non interaction blocks are superior to a variant of the original models from which we would have completely removed k blocks: for example, a Bert in which only 2 blocks are kept, trained according to the same protocol, obtains an exact match of only 26.4 and an F1 score of 36.2. Experiments are all carried on SQuAD v1.1 here because OpenSQuAD is based on it. Nevertheless, the Dil variants also apply to SQuAD v2.0. For instance, Albert, DilAlber$t_{k=6}$ and DilAlber$t_{k=10}$ respectively obtain an F1-Score of 81.4, 80.3 and 72.4 on its dev set.

Figure 3. Evolution of DilBert's (left) and DilAlbert's (right) Exact Match (EM) and F1-score (F1) on the SQuAD v1.1 dev set with respect to the number of non interaction blocks k. The performance of Bert (left) and Albert (right) are also displayed as a horizontal line.



There is a slight difference between the behaviors of DilBert and DilAlbert. Whereas the latter seem to have a smooth evolution with respect to k, the former has two plateaus from $k = 0$ to $k = 4$ and from $k = 5$ to $k = 10$ and a decrease of performance in between. With inspiration from Clark, Khandelwal, Levy, and Manning (2019), our hypothesis is that since each block in Bert has its own set of weights, it could be that the pre-trained attention heads in the 6^{th} layer have a component useful for the targeted eQA task.

Speedup

To analyze the speedup entailed by delayed interaction, we consider a simple open domain question answering setting where $q{=}100$ questions are asked and the RC model (e.g. Bert) searches for the answers in a database of $p{=}100$ passages. We measure the total time required by Bert, Albert, DilBert and DilAlbert to process all question-paragraph pairs (Table 1). For the Dil variants, we detail the time required in the non interaction blocks to do the independent processing on questions and paragraphs and in the interaction blocks to do the dependent processing on question-paragraph pairs. We consider two values for the number of non interaction blocks: $k = 10$ because it allows preserving most of the eQA performance and $k = 11$, the largest possible value. Experiments are carried on a bull server with a Nvidia Tesla V100 GPU and a 5-cores/10-threads Intel Xeon Gold 6132 (2.6-3.7 GHz) CPU.

Table 1. Time consumption to search for the answer of 100 questions in 100 passages with Bert, Albert, DilBert and DilAlbert. "NI Q" (resp. "NI P") refers to the time required to process the q questions (resp the p passages) in the Non Interaction blocks, and "I Q-P" refers to time required to process the $p \times q$ pairs in the Interaction blocks. We also report in parentheses a time consumption normalized with respect to the number of blocks (l k or $l-k$ and inputs (p q or $p\times$ q)

		Bert	DilBert$_{k=10}$	DilBert$_{k=11}$	AlBert	DilAlBert$_{k=10}$	DilAlBert$_{k=11}$
	NI Q		1.0 (1e-3)	0.9 (8.2e-4)		1.2 (1.2e-3)	1.4 (1.3e-3)
GPU	**NI P**		0.8 (8e-4)	1.1 (1e-3)		1.0 (1.0e-3)	1.1 (1.0e-3)
	I Q-P	117.9 (9.8e-4)	17.0 (8.5e-4)	9.9 (9.9e-4)	142.2 (1.2e-3)	22.4 (1.1e-3)	13.1 (1.3e-3)
	Total	117.9	18.8	11.9	142.2	24.6	15.6
	Speedup	x1	x6.3	x9.9	x1	x5.8	x9.1
	NI Q		4.2 (4.2e-3)	4.8 (4.4e-3)		4.1 (4.1e-3)	4.8 (4.3e-3)
10-threads	**NI P**		15.4 (1.5e-2)	16.7(1.5e-2)		16.7 (1.7e-2)	17.6 (1.6e-2)
CPU	**I Q-P**	2602.6 (2.2e-2)	323.3(1.6e-2)	159.6(1.6e-2)	2597.6 (2.2e-2)	346.8 (1.7e-2)	176.3 (1.7e-2)
	Total	2602.6	342.2	181.1	2597.6	367.7	198.7
	Speedup	x1	x7.6	x14.4	x1	x7,1	x13.1

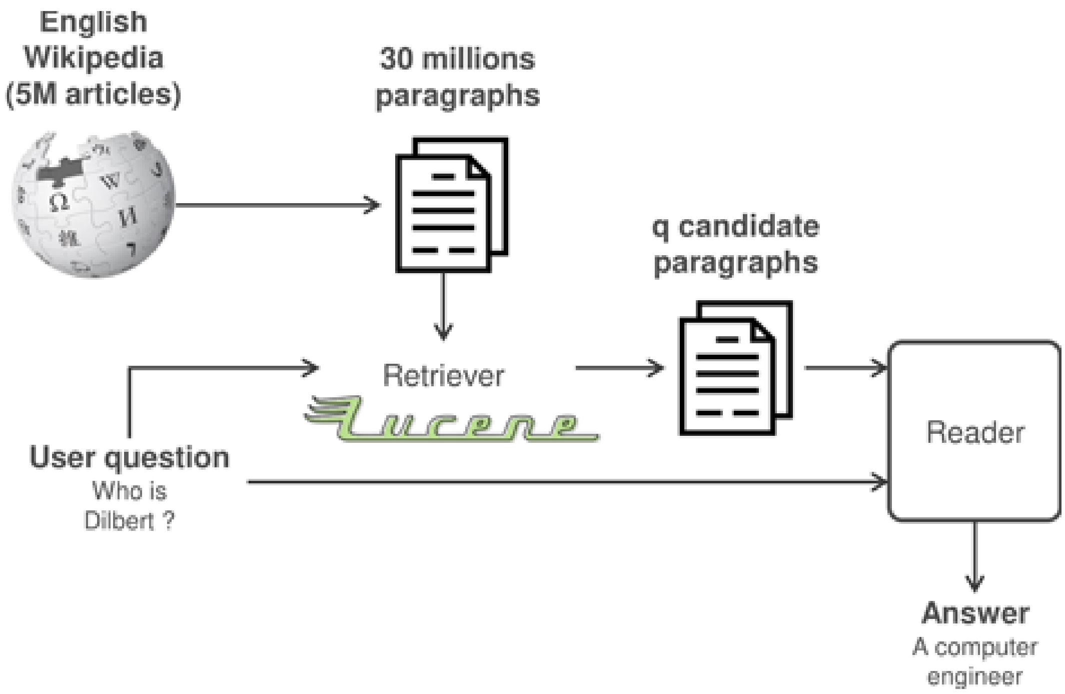
The empirical results confirm the theoretical intuitions of the complexity analysis. First, since intra-block calculations are parallelizable and a GPU has a great power of parallelization, we can see on GPU results that the processing time per block and per input sequence is approximately always the same (approximately 1e-3) whereas it depends more on the length of the sequence on CPU (about 4e-3 for the question with a length of $n_q \approx 16$ and 1.3e-2 for the question-paragraph concatenation with a length of $n\_s{=}384$). Second, in the ODQA setting, where we look for the answer to several questions (q) in a static set of paragraphs (p), the processing in interaction blocks takes most of the time because it is proportional to the number of pairs $p \times q$, which is much greater than p and q. Thus, since original models have $l = 12$ interaction blocks and Dil variants only have $l - k$ (1 or 2) interaction blocks, the speedup factor is close to $\frac{l}{l-k}$ (≈ 6 or ≈ 12 .

Open Domain QA

Apart from speedup, it is necessary to check the impact of delayed interaction on the quality of answers in Open Domain QA. We focus here, as Chen et al. (2017), on OpenSQuAD. The objective for algorithms is to answer to questions using the entire English Wikipedia[3]. More precisely, the questions are the same as the 10570 questions of SQuAD v1.1 dev set but algorithms are not provided with the associated paragraphs. Instead, they have to search the answer in a larger set of 5,075,182 articles.

Since our study specifically focuses on how partial attention can improve the reader Bert/Albert, we do not dwell on the choice of the retriever. Instead, we choose a proven ODQA method of the literature whose reader is based on a Transformer architecture, and replace it with our variants. The baseline BertSerini (W. Yang, Xie, et al., 2019) is a good candidate framework as it uses a robust retriever and Bert as a reader. Its implementation is not available so we propose our own (Figure 4).

Figure 4. General architecture of the question answering pipeline used in our experiments



We first apply a preprocessing step to properly index the Wikipedia dump with Lucene using the pyserini library, version 0.9.4.0 (P. Yang et al., 2017, 2018).

As Wikipedia articles are rather long and sometimes cover multiple topics, they are difficult to process as such for both the retriever and the reader (J. Lee et al., 2018; S. Wang et al., 2017; W. Yang, Xie, et al., 2019), so we start by splitting them into paragraphs. We consider two strategies: (1) Using a double newline as delimiter. We then discard all items with less than 30 characters leading to a number of paragraphs around 29.1 million as in (W. Yang, Xie, et al., 2019). (2) Using a fixed length of 100 words with a sliding window of 50 words as in (Z. Wang et al., 2019). We obtain close end-to-end results with both techniques but retain the slightly better second strategy for the final pipeline.

The splitted paragraphs are then processed with the pyserini script *index*[4] which outputs all the necessary inverted indices for retrieval. Then, we instantiate pyserini's retriever that relies on the inverted indices and that uses BM25 (pyserini's default parameters) as a relevance score s_{bm25}. For each question, it returns the p paragraphs with the top s_{bm25}. We choose $p = 29$ and $p = 100$ as in BertSerini. This retrieval step takes less than 0.15s on CPU with the hardware configuration described in the previous section. Then, the reader is applied to each paragraph and produces a start logit score s_s and an end logit score s_e for each token. To aggregate the results into a final prediction, we consider two variants: one that excludes the retriever score and selects the text span with the highest reader score $s_r = \frac{s_s + s_e}{2}$ across all paragraphs, and one that includes it by replacing s_r with $\mu s_r + \left(1 - \mu\right) s_{bm25}$ where μ is a hyperparameter between 0 and 1. A cross validation on a subset of the SQuAD v1.1 train set questions shows that, from 0.1 to 0.9 in steps of 0.1, the value of $\mu = 0.5$ leads to an optimal end-to-end performance for all considered readers (Bert, DilBer$t_{k=10}$, Albert and DilAlber$t_{k=10}$ fine-tuned on SQuAD v1.1). We add the multilingual version of Bert (Devlin et al., 2019; Pires, Schlinger, & Garrette, 2019) to open perspectives in other languages.

The overall pipeline is evaluated with the exact match (EM) and the F1-score (F1) between its answer predicted from Wikipedia and the expected answer. Obviously, the task here is more difficult than eQA since the paragraph containing the answer is not provided. Not only the IR part will select p paragraphs that might not contain the answer, but additional difficulties emerge (detailed in the Discussion).

To measure the impact of the retriever part, we compute the percentage of questions for which the expected answer at least appears in the p selected paragraphs (column **R** in Table 2). Since the paragraph splitting methodology and the version of pyserini were not detailed by W. Yang, Xie, et al.(2019) we were only able to obtain the same IR performance as Bertserini for $p=29$ but we obtained a lower **R** for $p = 100$.

The final end-to-end ODQA results exhibit interesting properties (Table 2). Although slightly worse on the eQA task (e.g. Figure 3), Dil variants almost always outperform original models for ODQA when the IR score is not used, and sometimes by a significant margin (e.g. when there is a high number of candidate paragraphs p). When exploiting the IR score, the conclusion are more mixed.

In fact, the original models (Bert, mBert, Albert) sometimes produce noisy predictions with high scores for passages with little relevance (i.e. a low score s_{bm25}) (Xie et al., 2020; W. Yang, Xie, et al., 2019). Delayed interaction acts like a regularization similar to dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), since the resulting architecture is the same as the original one, except that some connections are dropped in the first k blocks. Therefore, the variants are more subject to generalization. When using the retriever score in addition to the reader's predictions, all pipelines focus more on the relevant passages, which helps improving the overall performance. In this case, the original models manage to catch up with dil variants.

Table 2. End-to-end performance (EM, F1) of our ODQA pipeline with different readers. Results when excluding (w/o) or including (w/) the IR score are detailed. The number of retrieved passage is noted as p. EM is the Exact Match and R is the retriever score.

Model	EM		F1		R
	w/o	w/	w/o	w/	
DrQA (Chen et al., 2017)	27.1	-	-	-	77.8
R^3 (Wang et al., 2017)	-	29.1	-	37.5	-
Par. R. (Lee et al., 2018)	-	28.5	-	-	83.1
MINIMAL(Min et al., 2018)	-	34.7	-	42.5	64.0
Yang et al. (2019a):					
Bertserini (p= 29)	-	36.6	-	44.0	75.0
Bertserini (p= 100)	-	38.6	-	46.1	85.8
Our results:					
Bert (p= 29)	37.4	43.5	44.1	50.9	74.6
DilBert (p= 29)	38.8	42.0	46.7	50.4	74.6
Bert (p= 100)	31.8	44.3	38.0	51.8	82.4
DilBert (p= 100)	36.5	43.2	43.8	51.6	82.4
mBert (p= 29)	34.4	41.9	40.6	49.1	74.6
DilmBert (p= 29)	38.5	42.8	45.8	50.5	74.6
mBert (p= 100)	28.3	42.3	34.1	49.3	82.4
DilmBert (p= 100)	34.7	43.8	41.7	51.6	82.4
Albert (p= 29)	40.6	44.1	47.0	51.3	74.6
DilAlbert (p= 29)	39.7	43.6	47.7	51.8	74.6
Albert (p= 100)	36.2	44.9	42.1	52.1	82.4
DilAlbert (p= 100)	36.8	44.8	43.8	52.9	82.4

The results with multilingual weights are slightly worse than with English weights, but especially for the original model. In fact, the DilmBert variant is always better than mBert. As for the results with Albert, they are close to those of Bert but a little better in general.

The best performance (EM: 44.3, F1: 51.8) obtained in this study with a Bert base-english-uncased, simply fine-tuned for the eQA task on SQuAD v1.1 and combined with BM25, is better than Bertserini's, whereas the latter uses the same basic blocks and has a higher IR (R) score. This improvement is mainly due to engineering (documents splitting, version of Lucene), and is consistent with observations made in the literature. For example Xie et al. (2020) also report higher results for a BertSerini-like pipeline (EM: 41.8, F1: 49.5, R: 86.3).

To draw more robust conclusions on the comparison between Bert and Dilbert, we additionnaly provide, in Table 3, their performance on three other datasets frequently used for ODQA (TriviaQA, Natural Questions and WebQuestions). We consider the same trained weights for Bert and DilBert, the same document source (the 2016 Wikipedia dump) and the same retriever (BM25) as in the experiment with OpenSQuAD. The results are shown for the hypermeters p = 100 and $\mu = 0.5$. Except for TriviaQA when using the IR score, DilBert still perform slightly better than Bert.

Discussion

Complexity is a major industrial and societal concern that affects efficiency of systems in production and their carbon footprint (Thakur & Chaurasia, 2016). By approaching the ODQA problem with both the use of IR and modifications of the reader, we can greatly reduce computations. Dil variants, which can benefit from pre-trained weights already available, reduce the number of interaction blocks from 12 to 2 (resp. 1) with $k = 10$ (resp 11) reducing the cost by 85% (resp. 92%) in Bert/Albert,

Table 3. End-to-end performance (EM, F1) of our ODQA pipeline with Bert and Dilbert, for p=100, on three additional datasets. Results when excluding (w/o) or including (w/) the IR score are detailed.

	WebQuestions					Natural Questions					TriviaQA				
Model	**EM**		**F1**		**R**	**EM**		**F1**		**R**	**EM**		**F1**		**R**
	w/o	**w/**	**w/o**	**w/**		**w/o**	**w/**	**w/o**	**w/**		**w/o**	**w/**	**w/o**	**w/**	
Bert	9.8	10.6	18.2	19.7	75.0	9.7	12.6	15.1	19.0	62.0	31.9	**36.3**	38.9	**46.7**	84.2
DilBert	**10.1**	**10.8**	**19.5**	**20.3**	75.0	**11.2**	**13.2**	**17.3**	**20.2**	62.0	**32.2**	33.9	**40.1**	44.6	84.2

which soon will be standards in search engines. To give a point of comparison, the famous alternative of Bert known as DistilBert (Sanh,Debut, Chaumond, & Wolf, 2019) reduces the cost by 50% (essentially by going from 12 blocks to 6 blocks). On SQuAD v1, DistilBert gets roughly the same EM/F1 and speed as DilBert with $k = 6$. But DilBert can go further (its EM/F1 for $k = 10$ is as good as for $k = 6$ but speed is improved). Moreover, since the Dil mechanism is generic, we applied it to Albert, and DilAlbert with $k = 10$ not only gets a better EM/F1 than DistilBert, but also has about 4x fewer parameters and is about 3x faster in ODQA. Instead of making a comparison, a better idea would probably be to apply the Dil mechanism to Distilbert as nothing should prevent it. And nothing should either prevent the application of delayed interaction to the retriever part of the ODQA pipeline for cases where this one is also based on a Transformer (Z. Wang et al., 2019).

In addition to acceleration, Dil variants are in fact competitive, in ODQA, in terms of response quality. They can even surpass the original models when searching for answers in a wide variety of passages. Since the SQuAD v1.1 train set is based on a few hundred of Wikipedia articles, a tuned language models may show good eQA performance on its dev set due to over-specialization on certain topics (Xie et al., 2020). The regularization induced by restrictions in Dil certainly reduces the performance for the most relevant passage (Figure 3) but also unexpected behaviors on irrelevant passages (Table 2). More explicit solutions to improve generalization and multi-passage selection (e.g. global normalization or distant supervision) have recently been explored (Z. Wang et al., 2019; Xie et al., 2020).

The best performance in ODQA (in particular compared to eQA) still seems a little weak today (see the best exact match in Table 2). This can be explained by the additional difficulty of retrieval on Wikipedia but also by limits in the evaluation. Indeed, besides the retriever's bottleneck which results in a 15%-18% decrease, there are questions in the SQuAD v1.1 dev set that are no longer suitable in open domain (see Table 4).

Table 4. Examples of questions expected answers and predicted answers by one of the ODQA pipeline on OpenSQuAD. The answers here are all correct, but the metric does not reflect it.

Question	Expected Answer	Predicted Answer	EM	F1
When did Zwilling and Karistadt become active at Wittenberg?	June 1521	mid-1521	0	0
The Los Angeles Angels of Anaheim are from which sport?	MLB	Major League Baseball	0	0
How many fumbles did Von Miller force in Super Bowl 50?	2	two	0	0
What is the AFC short for?	American Football Conference	Asian Football Confederation	0	0.33
How many graduate students does Harvard have?	14000	15000	0	0
What position did Newton play during Super Bowl 50?	quarterback	QB	0	0

To end the discussion, we would like to mention a recently published approach called DeFormer (Cao, Trivedi, Balasubramanian, & Balasubramanian, 2020). It was developed independently of our work and we were not aware of it at the time we carried out the study. It is very close to the dil mechanism exploited in this paper except that Cao et al. (2020) apply an additional optimization step to make their variants closer to the original models. We obtain on SQuAD, with Bert, results similar to them. Despite the obvious connections, our study brings additional valuable elements, for instance: **(1)** We focus on the implication of a partial attention mechanism in ODQA where the acceleration factors are very interesting. We also bring new results showing how the variants can actually be competitive with the original models on OpenSQuAD. **(2)** We apply the mechanism to mBert and to Albert. The latter is a bit special because weights are shared between blocks. Our study shows that the mechanism is still compatible and that weights can even be shared between blocks without interaction and blocks with interaction. In view of the anteriority of DeFormer, we may in the future refer to the approach as DilAlbert or DeFormer-Albert.

Conclusion

Delayed interaction allows to accelerate the very famous Transformer by about an order of magnitude and does not need any additional pre-training. We can establish a link between our work, focused specifically on the impacts for ODQA, and a more general research direction consisting in speeding up deep language models using partial attention (Zaheer et al., 2020). There remains paths to explore in order to improve our work. One of them is to enhance learning by following ideas from Z. Wang et al. (2019); Xie et al. (2020), i.e. carrying a multi-passage fine-tuning. Another natural direction is to address the issue of memory management. Specifically, storing pre-computed representations of documents can become expensive for large-scale databases. It would be interesting to evaluate the impact of compressing them with sparsification techniques (Sun, Guo, Lan, Xu, & Cheng, 2016; Zhao et al, 2020) or binarization (Tissier, Gravier, & Habrard, 2019).

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.

REFERENCES

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval* (Vol. 463). ACM Press.

Białecki, A., Muir, R., Ingersoll, G., & Imagination, L. (2012). Apache lucene 4. In Sigir 2012 workshop on open source information retrieval (p. 17). ACM.

Cao, Q., Trivedi, H., Balasubramanian, A., & Balasubramanian, N. (2020). Deformer: Decomposing pre-trained transformers for faster question answering. doi:10.18653/v1/2020.acl-main.411

Chen, D., Bolton, J., & Manning, C. D. (2016). A thorough examination of the cnn/daily mailreading comprehension task. In *Proceedings of the 54th annual meeting of the association for computational linguistics (*volume 1*: Long papers)* (pp. 2358–2367). doi:10.18653/v1/P16-1223

Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading wikipedia to answeropen-domain questions. In *Proceedings of the 55th annual meeting of the association for computational linguistics (*volume 1*: Long papers)* (pp. 1870–1879). doi:10.18653/v1/P17-1171

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does bert look at? An analysis of bert's attention. In *Proceedings of the 2019 acl workshop blackboxnlp:Analyzing and interpreting neural networks for nlp* (pp. 276–286). doi:10.18653/v1/W19-4828

Dai, Z., Xiong, C., Callan, J., & Liu, Z. (2018). Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh acm international conference on web search and data mining* (pp. 126–134). doi:10.1145/3159652.3159659

Das, A., Yenala, H., Chinnakotla, M., & Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th annual meeting of theassociation for computational linguistics (*volume 1*: Long papers)* (pp. 378–387). doi:10.18653/v1/P16-1036

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (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)* (pp.4171–4186). Academic Press.

Guo, J., Fan, Y., Ai, Q., & Croft, W. B. (2016). A deep relevance matching model for ad-hocretrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management* (pp. 55–64). ACM.

Feldman, Y., & El-Yaniv, R. (2019). Multi-hop paragraph retrieval for open-domain question answering. doi:10.18653/v1/P19-1222

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). *Albert: A lite bert for self-supervised learning of language representations.* arXiv *preprintarXiv*:1909.11942.

Lee, J., Yun, S., Kim, H., Ko, M., & Kang, J. (2018). Ranking paragraphs for improving answer recall in open-domain question answering. doi:10.18653/v1/D18-1053

Lee, K., Chang, M.-W., & Toutanova, K. (2019). Latent retrieval for weakly supervised open domain question answering. doi:10.18653/v1/P19-1612

Lin, J. (2019). The neural hype and comparisons against weak baselines. In ACM SIGIR Forum (Vol. 52, pp. 40–51). doi:10.1145/3308774.3308781

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). *Roberta: A robustly optimized bert pretraining approach.* arXiv preprint arXiv:1907.11692.

MacAvaney, S., Yates, A., Cohan, A., & Goharian, N. (2019). Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 1101–1104). ACM.

Manning, C. D., Schütze, H., & Raghavan, P. (2008). *Introduction to information retrieval.* Cambridge University Press. doi:10.1017/CBO9780511809071

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111–3119). Academic Press.

Min, S., Zhong, V., Socher, R., & Xiong, C. (2018). Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th annual meeting of the association for computational linguistics (*volume 1*: Long papers)* (pp. 1725–1735). doi:10.18653/v1/P18-1160

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual bert? In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4996–5001). doi:10.18653/v1/P19-1493

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2383–2392). doi:10.18653/v1/D16-1264

Ren, Q., Cheng, X., & Su, S. (2020, April). Multi-Task Learning with Generative Adversarial Training for Multi-Passage Machine Reading Comprehension. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 8705–8712. doi:10.1609/aaai.v34i05.6396

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. *Nist Special Publication Sp*, *109*, 109.

Ryu, P.-M., Jang, M.-G., & Kim, H.-K. (2014). Open domain question answering using wikipedia-based knowledge model. *Information Processing & Management*, *50*(5), 683–692. doi:10.1016/j.ipm.2014.04.007

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.* arXiv preprint arXiv:1910.01108.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Sun, F., Guo, J., Lan, Y., Xu, J., & Cheng, X. (2016). Sparse word embeddings using l1-regularized online learning. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 2915–2921). Academic Press.

Thakur, S., & Chaurasia, A. (2016). Towards green cloud computing: Impact of carbon footprint on environment. In *6th international conference on cloud system and big data engineering (confluence)* (pp. 209–213). Academic Press.

Tissier, J., Gravier, C., & Habrard, A. (2019). Near-lossless binarization of word embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 7104–7111. doi:10.1609/aaai.v33i01.33017104

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention is all you need.* arXiv preprint arXiv:1706.03762.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., . . . Jiang, J. (2017). *R-3: Reinforced reader-ranker for open-domain question answering.* arXiv preprint arXiv:1709.00023.

Wang, Z., Ng, P., Ma, X., Nallapati, R., & Xiang, B. (2019). Multi-passage bert: A globally normalized bert model for open-domain question answering. doi:10.18653/v1/D19-1599

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (2019). *HuggingFace's Transformers: State-of-the-art natural language processing.* arXiv preprint arXiv:1910.03771.

Woods, W. A. (1977). *Lunar rocks in natural English: Explorations in natural language question answering*. Academic Press.

Xie, Y., Yang, W., Tan, L., Xiong, K., Yuan, N. J., Huai, B., & Lin, J. et al. (2020, April). Distant supervision for multi-stage fine-tuning in retrieval-based question answering. In *Proceedings of The Web Conference 2020* (pp. 2934-2940). doi:10.1145/3366423.3380060

Yang, P., Fang, H., & Lin, J. (2017, August). Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1253-1256). doi:10.1145/3077136.3080721

Yang, P., Fang, H., & Lin, J. (2018). Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, *10*(4), 1–20. doi:10.1145/3239571

Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., . . . Lin, J. (2019). *End-to-end open-domain question answering with bertserini.* arXiv preprint arXiv:1902.01718.

Yang, W., Zhang, H., & Lin, J. (2019). *Simple applications of bert for ad hoc document retrieval.* arXiv preprint arXiv:1903.10972.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems (pp. 5753–5763). Academic Press.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., & Ontanon, S. (2020). Big bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 33.

Zhao, C., Hua, T., Shen, Y., Lou, Q., & Jin, H. (2021) Automatic Mixed-Precision Quantization Search of BERT. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence Main Track*, 3427-3433. doi:10.24963/ijcai.2021/472

ENDNOTES

1 https://github.com/wissam-sib/dilbert
2 https://huggingface.co/models
3 For comparability reasons, we consider a dump from December 2016 as proposed by Chen et al. (2017).
4 https://github.com/castorini/pyserini

Wissam Siblini, after obtaining a Master of Engineering Degree from Ecole Centrale de Nantes (France) and a Master of Science Degree from the University Claude Bernard in Lyon (France), did a PhD thesis at the University of Nantes in partnership with Orange Labs in Lannion. His research focused on extreme multi-label classification and dimensionality reduction. Since he completed his PhD in 2018, he has been doing applied research at Wordline on Fraud Detection, Natural Language Processing, and Computer Vision.

Mohamed Challal, after obtaining a Master of Engineering Degree from Insa Lyon (France), started working at Societe Generale, Corporate and Investment Banking, as a Research And Development Engineer. His research mainly focusses on Natural Language Processing.

Charlotte Pasqual is a member of the Worldline Expert Community and of the Emerging Solutions and Market Insights team at Worldline Labs. She wrote her Masters' thesis in 2006 at the Fraunhofer Institute about motion detection using background estimation models and obtained the Master of Engineering Degree at ENSIIE. She joined the Worldline R&D team in 2010, where her main interests are natural interfaces and seamless experiences. She is currently leading a research program about natural language processing applied to customer journeys.