# BERT for information retrieval

Traditional retrieval models obtain ranking scores by searching for exact matches of words in both the query and the document. However, this limits the availability of positional and semantic information and may lead to 𝑣𝑜𝑐𝑎𝑏𝑢𝑙𝑎𝑟𝑦 𝑚𝑖𝑠𝑚𝑎𝑡𝑐ℎ [126]. By contrast, neural ranking models construct query-to-document relevance structures to learn feature representations automatically via word2vec [122] and GloVe [51]. There are two main classes of neural ranking models: representation-focused models and interaction-focused models.

Using TriviaQA for building an Open-domain Question Answering LLM

1. **Introduction**: The context of the problem to be solved will be presented.

Open-domain question answering (ODQA) models (Voorhees and Tice, 2000) are used extensively to automate the process of finding information in large databases of documents, and using this information to answer questions. A traditional and widely used framework for ODQA models is the *Retriever-Reader* (Chen et al., 2017), in which the ODQA model consists of two modules: An *information* *retriever (IR)* and a *reader*. The IR retrieves passages of information *(context)* from the document database, and from the question and these passages, the reader yields an answer. Other newly-suggested frameworks exist, such as the *Retriever-Only* (Lee et al., 2021b) in which the answer is extracted directly from the document database, and the *Generator-Only* (Roberts et al., 2020) in which the IR step is omitted and an LLM directly generates an answer to the question instead. This report, however, focuses on exploring the traditional *Retriever-Reader* framework.

Different methods exist for implementing the IR. Non-neural methods, such as TF-IDF (Chen et al., 2017) and BM25 (Mao et al., 2021), encode the question and the passages in sparse vectors and are limited to capturing syntactic and word-based similarities between the question and the passages (Qu et al., 2021). Recent neural network-based methods such as DPR (Karpukhin et al., 2020) however, encode the question and the passages in dense vectors and are able to capture deeper semantic-based similarities between the question and the passages.

Different methods also exist for implementing the reader, and can be divided into two categories: *Extractive* *readers* and *Generative readers*. Extractive readers locate and extract the answer from the given context passages (Karpukhin et al., 2020; Qu et al., 2021), while generative readers yield the answer by generating new text by means of token prediction (Raffel et al., 2020; Izacard and Grave., 2021).

In this short paper we present our retriever-reader-based open-domain question answering LLM that uses the TriviaQA dataset of Joshi et al. (2017). With this dataset that contains question-answer-evidence triplets, the LLM is trained using different methods for both the retriever and the reader. The dataset and the methods used are presented and explained more thoroughly in the methodology section, which is followed by the results and conclusions of this research.

Data

1. **Methodology**: The technical aspect, that is, the approach that was followed to carry out the project will be clearly described. You should collect important details about how the experimentation and analysis were designed.

**Data**

The dataset used for training the Question Answering Large Language Model is TriviaQA (Joshi et al., 2017). This dataset consists of 650K Question-Answer combinations with evidence from Wikipedia or the web. Trivia websites were used for finding the question-answer combinations after which documents were collected to prove the answer was correct. These documents were found by imputing the question in the Bing search machine and taking the 50 first URL’s listed for that question. Besides Wikipedia pages were collected that overlapped with the entity mentioned in the question. This distant supervision method means that the evidence documents were found separately from the question answer pairs. It turned out that 80% of the Wikipedia pages coupled to a question can also answer the question, whereas 75% of the webpages can.

* How did we split the data?
* How did we chunk the data? (maybe this is part of the methodology part)

Methodology

* Retriever
  + BERT
* Reader

1. **Presentation of the results**. In this section, the results obtained are presented. Result tables must be included correctly, using the specified evaluation metrics. It is also recommended to perform a **qualitative analysis**. The assessment and discussion of the results will be positively considered.

**Conclusions**. The conclusions of the work will be presented very briefly.

Compare to trivia paper: Neither approach comes close to human performance (23% and 40% vs. 80%), suggesting that TriviaQA is a challenging testbed that is worth significant future study.

You need to process only the test data/You are not updating any parameters so you don’t need the training data

We don’t have the actual test data

Dus met train en dev -> train omzetten in train en dev (7900 voor dev/validation dataset) en dev wordt onze test (7993) en test in apart document

Controlling generation LLM: niet te veel tokens laten genereren want anders te lange antwoorden en willen efficient

Eerst met weinig vragen beginnen

We retrieven augmented context

Geen tijd voor finetunen

Baseline: query question into LLM without context (want weet al veel),

daarna met docs erbij -> zelf bedenken welke augmentation en dat kan daarna ook gebruik worden voor beantwoorden model

Selecteren zelf hyperparameters dmv proberen, maar hoeft niet hele validation, 100 is al wel goed

First follow TUTORIAL

Bert is destructive, moeten ook generative doen

Destructive: train model om te zeggen of t in paragraaf staat of niet -> gaat wel veel sneller

Reader is generative

Split the test because it is a lot of computation time

Huggingface already trained models , wij Hoeven niet te trainen/finetunen

Siebe

**Retriever**

BM25 -> no preprocessing

BM25 with lemmatization

First question and wiki connected -> when question gets in, it knows which Wikipedia page and with embeddings it can create the best one

Then new question, it can see how similar to another question and which Wikipedia pages were related -> so it can retrieve the relevant info from those pages

Connection

Voor poster:

Explanation of method, evaluation:

Baseline

Leon’s

Siebe DPR but with not a lot of documents

The first question we should ask is *why should we represent text using vectors?* The straightforward answer is that for a computer to understand human-readable text, we need to convert our text into a machine-readable format.

*Sparse vectors are called sparse because vectors are sparsely populated with information. Typically we would be looking at thousands of zeros to find a few ones (our relevant information). Consequently, these vectors can contain many dimensions, often in the tens of thousands.*

Where sparse vectors represent text syntax, we could view dense vectors as *numerical representations of semantic meaning*. Typically, we are taking words and encoding them into very dense, high-dimensional vectors. The abstract meaning and relationship of words are numerically encoded.

**References**

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, Online. Association for Computational Linguistics.

Joshi, M., Choi, E., Weld, D., & Zettlemoyer, L. (2017). TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *Proceedings Of The 60th Annual Meeting Of The Association For Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/p17-1147

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for opendomain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, Online. Association for Computational Linguistics.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021b. Learning dense representations of phrases at scale. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 6634–6647, Online. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4089–4100, Online. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for opendomain question answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5835–5847, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140):1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5418–5426, Online. Association for Computational Linguistics.

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 question answering track. In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC’00), Athens, Greece. European Language Resources Association (ELRA).