# 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 models are used extensively to automate the process of finding information based on large databased of documents.

In this short paper we present our open-domain question answering LLM that uses the TriviaQA dataset of Joshi et al. (2017). With this dataset that contains question-answer-evidence triplets is, the LLM is trained using different methods for both the retriever and the reader. These methods are presented in the methodology sections and results and conclusions of this research are shown afterwards.

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.

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

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.