Open-domain question answering with Electra

Cedric Sadeu and **Mamoudou Sacko** and **Oumayma Messoussi**
Computer and Software Engineering Department
Polytechnique Montreal
Montreal, Canada
{cedric-2.sadeu, mamoudou.sacko, oumayma.messoussi}@polymtl.ca

Abstract

This article introduces the work of our group for the INF8460 class project. The studied project is an open-domain question answering pipeline able to abstain from answering questions with no answer. Our final pipeline, first, computes TF-IDF vector representations of the questions and passages. Then, for each question, the passages were ranked according to the cosine similarity score and the top-25 were passed through to the next step. Next, the answer extraction phase utilized the ELECTRA model pretrainted on SQuAD2.0 and fine-tuned on the custom competition dataset to generate candidate answers. Finally, a threshold was applied to the answer scores to determine non-answerable questions. This article also highlights the different techniques and approaches studied for each step of the project, thus giving way to future improvements and alternative implementations.

1 Introduction

Question answering is a very active area of research in natural language processing. The goal of such systems is to process a set of corpus passages and return an answer to a user query, both in natural language. Such pipelines can be applied to dialog systems.

Question : What causes precipitation to fall?

Passage : In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

Answer : gravity

Figure 1: A sample question and answer.

To highlight the importance of this task, a variety of challenge tracks are proposed to drive and encourage continuous research and improvement of this task, such as the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) which uses passages extracted from Wikipedia documents.

The breakthroughs in performance in the recent years due to advances in deep learning have made it possible to surpass human performance in many tasks including question answering. Large volumes of annotated data like SQuAD allow us to effeciently train these models and adapt them to specific domains.

2 Related work

Previous works for question answering only focused on answerable questions, meaning that all questions were assumed to have an answer in the given passage. Examples of these earlier approaches are (Kadlec et al., 2016) and (Wang et al., 2017). More recently, with the emergence of datasets like SQuAD2.0 and the significant advances in deep learning, this task became more representative of the real-world by including non-answerable questions. Thus, newer models are more robust and able to detect these questions to avoid extracting an answer or associate it to a reduced confidence score. Kumar et. al. introduce a dynamic memory network (Kumar et al., 2015) using an iterative attention mechanism, therefore the attention is focused on the input and previous iteration result. Zhang et. al. propose a retro-reader architecture (Zhang et al., 2020) which performs a better verification for an answerable questions.

3 Overview

3.1 Question and passage representation

Task definition

This first step focuses on transforming raw text into *vector representations*. Let $S = \{w_0..w_n\}$ be a sequence of words forming a question or a passage.

The goal is to map S into a vector space, resulting in a representation $V = [v_0..v_k]$, v_i being float values, and $k <= n$. This highlights semantic similarities between sequences because their associated vectors will be close.

Techniques

Below is a brief introduction to the representation techniques we tested:

1. Term Frequency - Inverse Document Frequency, typically abbreviated as **TF-IDF**, is a common scheme where term frequencies are weighted by the inverse document frequency, first proposed by (Jones, 1972), to reduce the impact of very frequent terms that do not bear a lot of useful information. The formula often used is:

$$tf\text{-}idf(t, d) = tf(t, d) * idf(t)$$

where $tf(t, d)$ is the frequency of token t in document d, and $idf(t) = log(n/df(t)) + 1$, n being the total number of document in the corpus, and $df(t)$ is the number of document where the token t appears.

2. GloVe stands for Global Vectors for Word Representation, is an unsupervised model that produces dense vector representations. As introduced by (Pennington et al., 2014), the main advantage of this model is utilizing the global statistics from the co-occurrence matrix. Thus, it avoids local context windows such used in approaches like skip-gram (Mikolov et al., 2013) and perform better than global matrix factorization methods such as LSA (Deerwester et al., 1990) on word analogy tasks.

3. BERT is a model that became the new reference in many natural language processing tasks. The model as presented in (Devlin et al., 2019) greatly benefits from the transformers (Vaswani et al., 2017) architecture, more precisely the encoders part, and also employs an attention mechanism allowing it to look at an entire document as a word's context and highlight the most related words, unlike context window-based approaches. BERT also offers a variety of pre-trained models on vast corpora making it a very good option for fine-tuning on specific datasets or for generic word representation, often by averaging the embeddings of the last layer or creating a certain combination of the outputs of different hidden layers.

3.2 Passage retrieval and ranking

Task definition

In most question answering systems, this step is employed to select the set of $top-k$ passages most plausible to contain the answer to a given query q. Both the question and the passages in the corpus will be in vector format and a certain semantic similarity computation model is used.

Techniques

Below is a brief introduction to the ranking approaches we studied and tested:

1. Cosine similarity is a very common approach for passage ranking. Given the vector representations of a query q and train passages $\{p_0, ..., p_N\}$, N being the total number of passages, cosine similarity is computed for every pain (q, p_i) and then the output is sorted in descending order. Thus, the $top - k$ passages for each query are selected.

2. Siamese networks

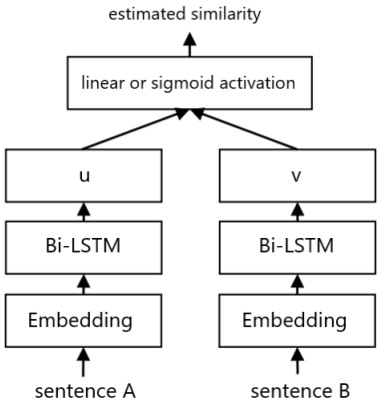

Figure 2: An example of siamese architecture for similarity estimation.

Siamese neural network are networks that contain two identical branches or sub-networks sharing weights and parameters as shown in figure 2. The inputs of the two branches are propagated through the branches with the goal to compare their corresponding feature vectors to detect similarities. This class of neural network is often used in computer vision to compare a pair of images.

3. Neural network regression is another fairly straightforward approach. Using a vanilla feed-forward neural network to estimate the similarity score between in pair of text embeddings concatenated. For our project, we juxtapose vector representations of a pair (query, passage). Like most deep learning models, the key to improving the performance of such regression is heavily related to the type and size of the training dataset.

3.3 Answer extraction

Task definition

The last step of our question answering pipeline in the answer extraction model. Let $top - k = p_0, .., p_k$ be the list of ranked passages that are most similar to a test query q. The model extracts a candidate answer for q from each passage p_i thus yielding a list of candidate answers. Finally, the candidate answers are ranked to select the most likely answer.

Techniques

We briefly describe the models we tested:

1. BERT for question answering is one of the many varieties of models based primarily on BERT, benefiting from its pre-trained version, to address a more specific task. As shown in figure 3, the model generates probabilities for each passage token being a start or end token, then the tokens forming a span with the highest probabilities is selected, and a score may also be computed combining the corresponding probabilities.

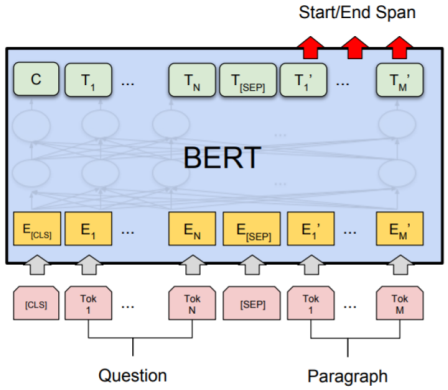

Figure 3: Adaptation of BERT for question answering.

1. ELECTRA is one of the most recent models that achieved SOTA performance on the SQuAD 2.0 benchmark. As presented in (Clark et al., 2020), it is a self-supervised masked language modeling model that is originally trained to differentiate between 'real' and 'fake' inputs. The fake inputs are obtained from a generator network ans used to replace some input tokens instead of the masking approach used in BERT, thus outperforming it.

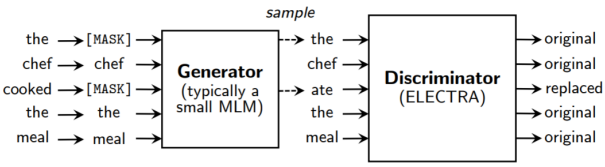

Figure 4: ELECTRA training approach.

4 Experiments

4.1 Datasets and metrics

Custom Kaggle data specific to the competition was provided. Overall, it includes 100,000 corpus passages and three subsets of format $(id, query, passage\_id, answer)$: a train set with 106,175 queries, a validation set and a test set with 10,000 queries each (no associated answers provided for test data).

For training the regression FFNN in step 2, we used the dev subset of the MS-MARCO dataset (Bajaj et al., 2018) from the TREC 2019 Deep Learning Track (Craswell et al., 2020), which is constructed from Bing search results. In total, the dataset contains around 3.2 million documents. So we limited our experiments to the dev set of 519,300 records in the format (query_id, doc_id, rank, score). Thus, we can train a model to predict the similarity score between a question and a passage.

To evaluate the passage ranking module of our pipeline, we use top_10 and top_N precisions, which indicate if the ground truth passage is included in the top_k.

4.2 Implementation

For our experiments, we used Google Colab for an interactive and collaborative environment for the team. We mainly used Pytorch, NLTK and HuggingFace as libraries to build the pipeline.

For the first step in our work, we have tried three types of representations after having pre-processed the corpus. Pre-processing includes tokenization, stop word removal and lemmatization. The first vector model was the TF-IDF representation of 15,000 features with unigram only, then taking into account n-grams of size 1 to 3 representation. Then, based on pre-trained GloVe embeddings, we were able to obtain the embeddings of our documents by averaging the embeddings of the words found in the doc, resulting in vectors of size 300. We also proceeded in the same way with the bert embeddings.

The second step was to find a better way to order our passages. As a first approach we calculated the similarity between the questions and our passages with the cosine metric. This process was carried out for all three above-mentioned representations. The TF-IDF representation was the most successful given the limited resources available in Colab. We also proceeded to train a siamese model and

Representation	Metric	Top-N	Precision 10	Precision N
TF-IDF bigramme (1,3)	cosinus similarity	25	0.475	0.572
TF-IDF unigrame	cosinus similarity	25	0.580	0.680

Table 1: Best models for passage ranking for the validation set.

a FFNN from scratch for the similarity estimation using the MS-MARCO dataset. Also, the size of the documents in MS-MARCO was considerably larger, therefore they were truncated and not efficiently represented in step one.

The last step was to extract the answers from our top passages. For this, we used several models derived from BERT. We first started with the *bert-large-uncased-whole-word-masking-finetuned-squad* base model from the transformers package which was pre-trained on SQuAD1.0. To improve this module, as shown in table 2, we experimented with Distibert and ELECTRA using different pre-trained weights and training configurations. We performed fine-tuning on SQuAD2.0 and our custom data.

4.3 Results

Our results derived from our choices of representations, ranking and response extraction models are summarized in tables 1 and 3. These results allowed us to find the best combination of modules to build the most efficient pipeline.

The performance metrics were obtained by applying the correction script available on Moodle.

3	2178	itunes
4	35540	<No Answer>

Figure 5: Sample answers extracted for validation set questions.

4.4 Discussion

1. Question and passage representation

After multiple experiments, we found that TF-IDF was the best fit representation for our project for various reasons: due to the limited computational resources with Colab, it was not possible to get the vector representations of all passages in our corpus with BERT. For BERT and GloVe, averaging the vectors from tokens yields a very poor performance. Therefore, with TF-IDF, we were able to control the dimensions of our vectors to best suit our computational needs. Ideally, we would use the full vocabulary to represent our documents but a size of 15,000 was a good trade-off between performance and RAM.

2. Passage ranking

For this next step, both the Siamese network and the FFNN that we trained on MS-MARCP were unsuccessful due to the fact that the distribution of the dataset differs from our Kaggle data since they are not sampled from the same source. Consequently, we used the cosine similarity as the main measure to rank the passage. We experimented with a few sample questions from the validation set to arrive at a decision for the value of top_N and we observed that $N = 25$ often contains the correct passage id.

3. Answer extraction

Our baseline was the BERT large model pre-trained on SQuAD1.0 against which we will compare the other tested approaches. From there, we proceeded to train a Distilbert model on the SQuAD2.0 dataset. This allowed for a performance improvement of about 5% with the EM metric. Next, we extracted answers using two varieties of the ELECTRA model: first, the pre-trained ELECRTRA model on SQuAD2.0 which improved the performance significantly arriving at 15.6% EM and 20.3% F1 score. Next, we fine-tuned ELECTRA on our custom corpus which defied our expectations and resulted in a slight performance drop of 0.6% EM. Unfortunately, due to limited resources, the training and fine-tuning process took several hours thus we were not able to test different hyperparameter values to improve the last model.

5 Conclusion and future work

Our work proposed an open-domain question answering pipeline using TF-IDF representations, cosine similarity for passage ranking and ELECTRA model trained and fine-tuned on SQuAD2.0 and the competition data for answer extraction. This placed our team in fourth place on the competition leaderboard with 37.6% EM on 70% of the test data. The main challenges faced duringt this project were limited computational resources since most NLP tasks require handling of large corpora and lengthily training. The limitations of our approach are the use of a threshold on candidate answer scores to determine non-answerable questions. Therefore, a potential improvement can be maid

Model	Tokenizer	Epoch	Batch	Optimizer
distibert* fine-tuned SQuAD2.0	DistilBertTokenizerFast	6	4	Adam
distibert* fine-tuned SQuAD2.0 + custom data	DistilBertTokenizerFast	3	4	Adam
electra** fine-tuned custom data	DistilBertTokenizer	2	8	Adam

Table 2: Training model configurations for answer extraction.
(distibert*: distibert-base-cased, electra**: deepset/electra-base-squad2)

Model	Exact Match	F1
bert-large-uncased-whole-word-masking-finetuned-squad	0.058	0.144
distibert* fine-tuned SQuAD2.0	0.102	0.178
electra** fine-tuned custom data	0.156	0.203
electra**	0.162	0.248

Table 3: Results from full pipeline with trained answer extraction models on the validation set.
(distibert*: distibert-base-cased, electra**: deepset/electra-base-squad2)

with better vector representations benefiting from the entire training vocabulary and more fine-tuning for the answer extraction module.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *CoRR*, abs/1603.01547.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada. Association for Computational Linguistics.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension.