Question Answering on Wikipedia

Marco Aspromonte
marco.aspromonte@studio.unibo.it

Francesco Romito
francesco.romito2@studio.unibo.it

January 2022

Abstract

The aim of this project is to develop an online question answering model on the **Wikipedia knowledge base**. We develop it using a pipeline consisting mainly in a keyword extractor, a web scraping phase and a finally the question answering step which has been done using transformers, whose high performances in the field of natural language processing are well-known. In particular, we focused on **distil-BERT**, because from our main project **Question Answering on SQuAD dataset**, we evaluated it as a good compromise in terms of time, computational effort and results. The model is trained on **Squad2.0** dataset, because we consider also not answerable questions which are not included in the 1.0 version. In this case, in order to validate the model, we have to check how much accurate are the phases of the pipeline, in particular the keyword extracted used for the Google research, the Wikipedia section's extraction and the reliability of the correct answers. For the validation phase we decided to test our models on a split of **DB-Pedia** Dataset in order to **evaluate the accuracy of the keyword extractors**, then we evaluate the **correctness of the answer** with **Natural-questions** Dataset.
The Distil-BERT trained weigths are downloadable here from MediaFire.

Contents

1 Datasets

1.1 SQuAD2.0

From the official page of the dataset, the *Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.* It is composed by 150k questions which 50k are not answerable. We chose this dataset for the training phase of the pipeline, which consists in a distilBERT transformer model. TODO inserire numero training elements

1.2 SimpleDBpediaQA

The **SimpleDBpediaQA** is a benchmark dataset for simple question answering over knowledge graphs that was created by mapping SIMPLEQUESTIONS entities and predicates from Freebase to DBpedia. We chose this dataset to test the performances of the keyword extractors of our pipeline. It is divided in train, test and validation split. Therefore for our testing purposes it was enough to use just the train split. It contains 30186 rows composed by a column with a query and a column with its subject. It was necessary not only the usual text-normalization phase, but also to remove all the rows in which the subject of the question was not completely included in the question column, given that our keyword extractors are just able to extract a keyword from a span of text. After this we end up with a dataset of 23841 Query-Subject pairs.

	Query	Subject
0	what movie is produced by warner bros?	warner bros
1	who is a musician born in detroit?	detroit
2	which city did the artist ryna originate in?	ryna
3	who produced the film rough house rosie?	rough house rosie
4	what is the language in which mera shikar was ...	mera shikar

Figure 1: 5 rows of SimpleDBPEDIA

1.3 Natural Questions

Natural Questions is a question answering dataset in which each question consist of real anonymized, aggregated queries issued to the Google search engine. An annotator is presented with a question along with a Wikipedia page from the top 5 search results, and annotates a long answer (typically a paragraph) and a short answer (one or more entities) if present on the page, or marks null if no long/short answer is present. In order to test the entire pipeline, from this dataset we extracted the first 500 answers with a short answer, ending up with a set of 500 question-short answer pairs.

	Question	Answer
0	what purpose did seasonal monsoon winds have o...	enabled European empire expansion into the Ame...
1	what does hp mean in war and order?	hit points or health points
2	what happens to the rbc in acute hemolytic rea...	rapid destruction of the donor red blood cells...
3	what is the most current adobe flash player ve...	28.0.0.137
4	what is the first step in the evolution of the...	photoreceptor proteins that sense light

Figure 2: 5 rows of Natural Questions

2 The pipeline

The entire project aims to answer an open question searching on Wikipedia and is composed by 3 main parts:

1. **Keyword extraction**: once a question is submitted, it is passed first to a keyword extractor that will extract a span of text from the question of maximum 3 words. To do this we chose to include a probabilistic algorithm called **YAKE!**, and a BERT-based neural network called **KeyBERT**, comparing and mixing their results, in order to improve the efficiency model.

2. **Web scraping and articles selection**: once we extract the keywords of a question, we use them to pick a set of Wikipedia pages from which we retrieve a selection of relevant paragraphs, in which there is probably the written question.

3. **DistilBERT question answering**: once we have a set of paragraphs (contexts), we can provide each of them in pair with the question to the DistilBERT transformer, so we can obtain a set of eligible responses with their probabilitiy and, through an evaluation phase, we can elict the most likely answer.

2.1 Keyword extraction

2.1.1 YAKE!

YAKE! is an algorithm proposed in 2018 that aims to extract keywords given a span of text. First it cleans the document, splits it into single words and for each of them a score basing on **5 features** is assigned:

- **Casing**: reflects the casing aspect of a word.
- **Word Positional**: values more those words occurring at the beginning of a document based on the assumption that relevant keywords often tend to concentrate more at the beginning of a document.
- **Word Frequency**: indicates the frequency of the word, scoring more those words that occur more often.
- **Word Relatedness to Context**: computes the number of different terms that occur to the left (w.r.t. right) side of the candidate word. The more the number of different terms that co-occur with the candidate word on both sides, the more meaningless the candidate word is likely to be.
- **Word DifSentence**: quantifies how often a candidate word appears within different sentences.

All these features are heuristically combined into a single score which is assigned to each term. Then the algorithm considers a sliding window of 3-grams so the candidate keywords will be a set of all continuous 1, 2 and 3 grams that can be generated from the given document. Each candidates has a score that is the combination of the score of each word.

2.1.2 KeyBERT

KeyBERT is a simple implementation which uses BERT-embeddings to assign a set of features to all the 1, 2 and 3-grams of a text. Then this vector embeddings are used to find which is the most similar to the text itself, using a cosine similarity as metric. In particular, we used the model called **all-mpnet-base-v2**, which maps sentences and paragraphs to a 768 dimensional dense vector space. We initialize the parameters of such model as follows:

- **keyphrase_ngram_range=(1, 3)**, to set the length of the resulting keywords, in this case each keyword is composed by at most 3-grams
- **stop_words='english'**, the stopwords to remove from the document
- **top_n=5**, return the top 5 keywords

For example, if we consider the question: **When was adopted the Declaration of Independence?**, the keyword extractor returns **'adopted declaration independence'**, used for the research.

2.2 Web scraping and articles selection

At this point of the pipeline we have the question and one keyword related to it. The next step is about searching the keyword followed by the word 'Wikipedia' on the english Google search engine, filtering those Wikipedia pages which are among the first 5 results of the research. This is done into the function **generate_wikipedia_namepages()**, that returns a list of wikipedia urls. Through the Wikipedia API we can obtain the Wikipedia pages in a more handly way than the HTML raw, so we can simplify the text cleaning phase. For each page, we obtain a unique string containing all the paragraphs which are preceded by the string label 'Section', so we can easily split them. Then we store them in a dictionary that contains as keys the names of the Wikipedia pages, as values the paragraphs. This is done in the function **get_articles_from_namepage()**. This dictionary contains at most 5 keys but each of them could contain too much paragraphs, the most of them totally unrelated to the topic of the question. Moreover an excessive number of paragraph to feed the network will slow down dramatically the prediction phase. To avoid this we call the function **context_similarity()**, which takes as parameters the question and the dictionary of collected articles. First it converts the question and each paragraph into matrices of TF-IDF features. In this way we can roughly classify the documents and

we can compare each pair of question-context using the cosine similarity as metric to evaluate how much they are related. At the end, for each key we keep no more than the top 5 articles in the ranking so we will have at most 25 candidate ones. Here we have an histogram representing the distribution of such candidates with respect to the normalized confidence score associated to each of them.

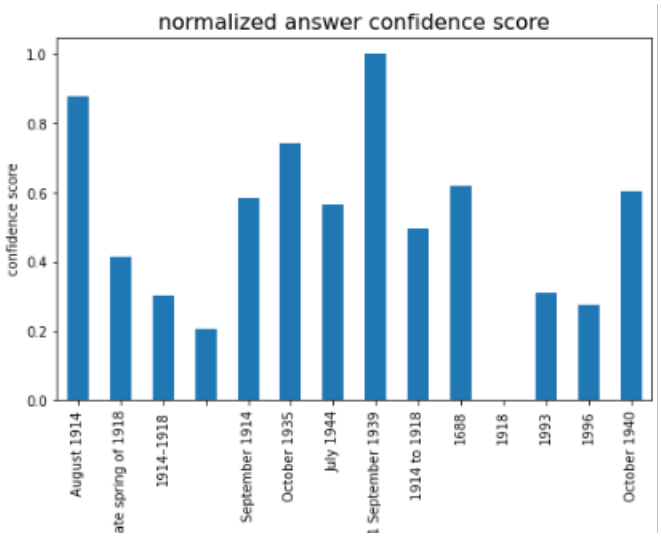

Figure 3: Normalized distribution of candidate answers wrt the confidence score

To speed up the answering process we have implemented a further option that filters out the computed candidates that are too much distant from the most similar one; this distance is tuned with a **treshold** (for test reason, we set it to 60%). If at the end of this process we have no collected articles the algorithm will stop ending up with no answer, otherwise they will be processed to feed the question answering neural network. So the resulting histogram , if we have too much distant confidence scores with respect to the maximum one, will be more pruned:

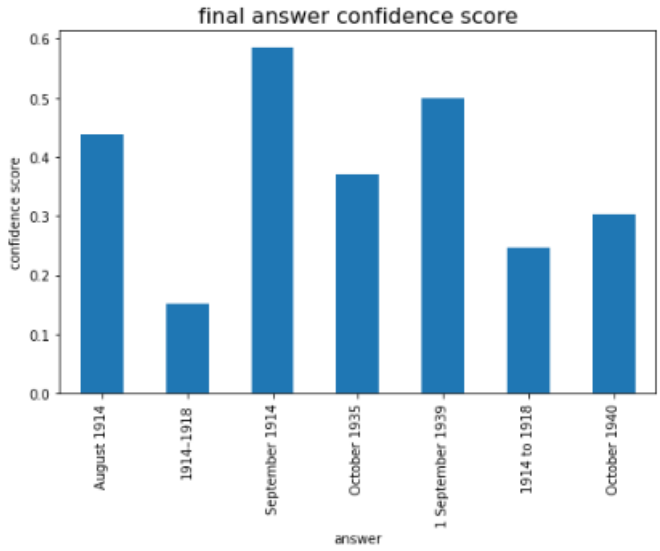

Figure 4: Normalized distribution after pruning with treshold set to 60%

2.3 DistilBERT question answering

In this phase we will deploy a faster and lighter version of BERT from huggingface transformers called DistilBERT. We have trained this model for two epochs with the training split of the SqUAD v2.0 (weigths download link in the abstract), so this model is able to give in case no answer to a given pair of question-contexts. Therefore, before we feed the data to the neural network, text must be tokenized, where each word is converted into integer. In particular, we have used the function tokenize_dataset, in which we tokenize the input sentence, we add CLS and the separator tokens, we pad or truncate the sentence to the maximum length allowed, in this case 384, and we manage the overlapping with the doc_stride parameter set to 128.

This model is iteratively fed with all the collected question-context pairs, so we obtain an answer associated to its probability for each of them. If there are no answer at all the algorithm will conclude with a failure, otherwise the set of answers will be subjected to a weighted voting process to elict the most likely answer.
The voting process is implemented in the function **voting_by_probability()**. Here we consider all the confidence scores associated to each answer and we normalize them trough the Min-Max normalization; then we multiply this normalized score to the frequency of such answer. Then we classify as 'Correct' the answer with the highest score and we use it as final result of the entire process.

3 Results

In this section we will discuss about results and the validation of our model. The metrics used to validate the model are the **precision, recall and f1-score**. We have computed such metrics both for the **keyword extractors** , and for **computed answers**. For the first validation step we used DBpedia as ground truth, for the second one we used Natural Questions.
Due to the complexity of the open-questioning task it was narrowness to use only a metric that does not takes into account the mining of a phrase, given that the same concept can be written in several ways. That is why we have also used a similarity function based on semantic to evaluate the correctness of a given answer.

3.1 Evaluation on Keyword Extractors

We will discuss about the quality of the two used keyword extractors, YAKE! and KeyBert, so for each of them we will post the results of the metrics, in order to infer which is the most powerful between them in terms of accuracy and computational effort.

- **YAKE!**:
  - average f1: 0.55
  - average precision: 0.98
  - average recall: 0.39
- **KeyBERT**:
  - average f1: 0.59
  - average precision: 0.97
  - average recall: 0.43

Analyzing these results it is clear that KeyBERT outperforms the YAKE! algorithm, anyway the difference is not so hard, also taking into account that KeyBERT uses the embeddings of a neural network to extract a keyword, so involving the semantic of a phrase, while YAKE! is only a probabilistic algorithm. This is reflected also in the time needed to extract a keyword, in fact

3.2 Evaluation on answers

We applied the metrics used for keyword extractors also for the computed answers with respect to the ground truth ones, ending up with the following results:

- **YAKE!**:
  - average f1: 0.10
  - average precision:0.12
  - average recall: 0.11
- **KeyBERT**:
  - average f1: 0.16
  - average precision: 0.18
  - average recall: 0.16
- **Mixed**:
  - average f1: 0.14
  - average precision: 0.17
  - average recall: 0.150

From the previous apparently worrying results, the model seems to not be able to give so much correct anwers. But in a context in which we have to evaluate how much is correct or wrong a given answer, it can be useful if we do not consider only the distance between strings as criteria, but it is more adequate if we deploy a **semantic distance** between them. This is reached encoding the embeddings both for ground truth answers

and for computed answers, using the embeddings of a BERT architecture, basically deploying the same tecnique used in the KeyBERT keyword extractor. Then, the answers are classified in 'Wrong' or 'Correct' trough the **semantic_similarity()** function, which combines the cosine similarity between the embedding matrices of the computed answer and the ground thruth and a treshold parameter to tune; for the posted results below it is set to 0.65. Therefore, if the treshold is too small, we may return too much false positives while using this value, generally the algorithm classifies correctly a given-true answer pair and that few false positives are pretty balanced with the false negatives:

- **YAKE!**
  Correct answers: 106
  Wrong answers: 394
  No Answer: 194
- **KEYBERT**
  Correct answers: 143
  Wrong answers: 344
  No Answer: 135
- **MIXED**
  Correct answers: 135
  Wrong answers: 365
  No Answer: 125

From the previous results, we can assert that using the KeyBert keyword extractor we can reach better results given that we have the highest number of correct ones.
On the other hand the keyword extractor that affects the worst the pipeline is the YAKE! one. It collects the highest number of 'Wrong' and 'NO ANSWER' and, consequentially, the lowest of correct ones.
Finally, the mixed approach shows an interesting behaviour given that the amount of correct and wrong answers is in the middle with respect to YAKE! and KeyBERT, while regarding the no answers this is the model with the best performances. Anyway this enhances the performances of the YAKE! pipeline but underperforms the KeyBERT one, meaning that this model answers the most but is more uncertain about the answer or, in any case, the answers retrieved with YAKE! often affects negatively the overall results.
Here we have some examples:

```
------------------------------------------------------------
Question : what's the dog's name on tom and jerry?

Ground truth answer : Spike
YAKE! answer: Charlie Adler
KeyBert answer : Spike
Mixed answer : Spike
------------------------------------------------------------
Question: which team won the Champions League in 2010?

Ground truth answer : Inter Milan
YAKE! answer: Inter Milan
KeyBert answer : Unirea
Mixed answer : Inter Milan
------------------------------------------------------------
Question : what is the main mineral in lithium batteries?

Ground truth answer : lithium
YAKE! answer: iron phosphate
KeyBert answer : lithium hydroxide
Mixed answer : iron phosphate
------------------------------------------------------------
```

Looking at the results, in the first two cases we can see how mixed approach is influenced positively both from YAKE! and KeyBERT, while in the third case it is underperformed by YAKE!. However , in the most of the cases the correct answer is retrieved by the KeyBERT model, even if there are some exceptions.

3.3 Error Analysis

Sometimes we have seen that the ground truth answer is not so confident as the mixed one, from which we have noticed that a crucial negative aspect of our results is that Natural Questions dataset is adaptable to our purposes, but it has a lot of errors. In particular, there are too much cases in which the ground truth answer is **wrong**, **logically incorrect** or **incomplete**, for example:

```
---------------------------------------------------------------
Question:  what are three different types of hotel properties?
Ground truth answer: Hotel barge
---------------------------------------------------------------
Question: what are the band members names of the rolling stones?
Ground truth answer: Mick Jagger
---------------------------------------------------------------
Question: what are the top five wine producing states?
Ground truth answer: California
---------------------------------------------------------------
```

Although, the correctness of extracted keywords and consequently the confidence of the answer is also influenced by the quality of the question, so the more the question is **well-posed**, the more the models are able to answer correctly, for example:

```
---------------------------------------------------------------
Generic Question : Where Pietro Mennea was born?
Provided Incorrect Answer : NO ANSWER FOUND :(


Specific Question: In what city was Pietro Mennea born?
Provided Correct Answer : Barletta
---------------------------------------------------------------
```

As a last case, the correctness of the answer is also influenced on how much the question is **specific** and **detailed**, for example:

```
---------------------------------------------------------------
Generic Question : What is the real name of Eminem?
Provided Incorrect Answer : Slim Shady


Specific Question : What is the baptism name of Eminem?
Provided Correct Answer : Marshall Bruce Mathers III
---------------------------------------------------------------
```

4 Conclusions

In general the pipeline is able to extract good answers, even if it is very sensible to the quality of questions. Therefore, if we could perform a better fine-tuning the model would be more robust and powerful, but we need a better-built dataset, that involves less wrong samples. In addictions, our pipeline extracts answers only from Wikipedia domain, so the probability of a correct answer is related to its knowledge base, for example on **historical** topics such probability is surely higher, while on other ones, as **topical topics**, it is probably lower.

References

[1] Kate Pearce, Tiffany Zhan, Aneesh Komanduri, Justin Zhan *A Comparative Study of Transformer-Based Language Models on Extractive Question Answering* (2021), Tensorflow, `https://arxiv.org/pdf/2110.03142.pdf`, consulted on January 2022.

[2] A distilled version of BERT: smaller, faster, cheaper and lighter `https://arxiv.org/abs/1910.01108`

[3] DBpedia project for content extraction `http://wikidata.dbpedia.org/develop/datasets`

[4] `https://github.com/valentinaboriano/Natural-Language-Processing`