Osservazioni

- I retangoli che vedete ai bordi servono per vedere quanto sono grandi i margini scrivibili della pagina, si possono togliere commentando la riga `\usepackage{showframe}`
- L'abstract adesso conta 267 parole. Di solito gli abstract non sono piu' lunghi di 300
- Il nome del nostro strumento di QA sta in una macro, per usarlo nel report scrivere `\nomefico`
- Il link al repo github sta in una macro, per usarlo scrivere `\github`
- Il link alla pagina web per QA (ancora da definire) sta in una macro, chiamata `\app`
- le immagini sono solo una bozza, vorrei che fossero approvate al 100% prima di realizzarle in latex
- Ho truccato il json di emepio (che e' in questa cartella in formato `.jsonl`), perche' aveva 36 long answer candidates e non veniva bene l'immagine del json. Se si decide di cambiare esempio, cambiarlo coerentemente

NOMEFICODASCEGLIERE
A BERT-BASED, OPEN DOMAIN
QUESTION ANSWERING SYSTEM

nomeFicoDaScegliere: A BERT-based, open domain question answering system

Gabriele Barreca, Mario Bonsembiante and Gemma Martini

University of Pisa

Abstract

Question answering (QA) systems can be seen as information retrieval systems which aim is to respond to queries, stated in natural language, by returning short answers or long sentences. The "so-called" *open domain QA task* adds the challenge of understanding if the answer to the selected question may or may not be found in a given paragraph, which content has been buried within large text corpora, such as Wikipedia.

Building such systems for practical applications has historically been quite challenging and involved. The spectrum of possible answers given a question and a paragraph, moves from the "simple" *yes/no answers* to the longer and more articulated *long answers*, to then get to a trade-off between expressive power and succinctness, the "so-called" *short answers*, which aim to enclose the answer in a single and possibly short sentence.

In this paper, we present a BERT-based implementation that solves an open domain QA task, providing all the three categories of answers listed above, with particular attention on the most widely studied kind, i.e. short answers. We achieve pretty good results, although not as good as the state-of-the-art, that was not the purpose of this work.

As expected and already stated in previous work, we conclude that predicting long answers per se is pretty unreliable, while much better results are achieved if the short answer is predicted and then enlarged with the whole paragraph it lies in, from the original text.

Contents

1 Introduction

Aggiungere introduzione

aggiungere una frase in cui si dice che cosa si trova in quali sezioni

2 The architecture

Before digging into the details of the machine learning core of our BERT-based QA system, let us define the outline of the responsive QA tool we developed.

In Figure 1 there is a pictorial representation of how the user interacts with the system, how it processes the information (server side) and how it prompts the results.

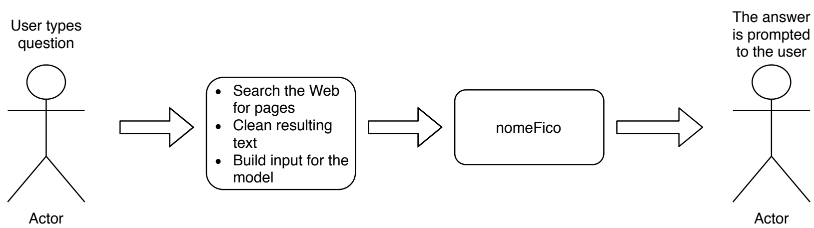


Figure 1: The full functioning of **nomeFicoDaScegliere**, combined with an effective user interface.

Aggiungere descrizione con immagini del workflow dell'applicazione, con un esempio che funziona, magari creando una sottosezione.

2.1 nomeFicoDaScegliere

We are now ready to discuss the implementation of **nomeFicoDaScegliere**.

We decided to tackle the open domain QA task by creating a stack of two neural networks, forming a two-layer architecture, as shown in Figure 2.

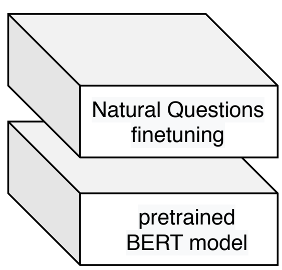


Figure 2: Sketch of the architecture of **nomeFicoDaScegliere**.

The first layer is built using BERT's [1] checkpoints [1] from Hugging Face, while the second layer is a neural network that uses BERT's embeddings and the Natural Questions (NQ) [4] dataset [2] with the aim of obtaining the answer to the question.

In the following paragraphs, the reader can find a more detailed explanation of the two layers.

2.1.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) [1] has been introduced by Google in 2018 and it has been defined as the biggest leap forward in the past five years.

Let us dig into details a bit more and explain how the BERT layer works.

BERT is an embedder, i.e. a neural network that translates words (more generally sentences and whole paragraphs) into sets of 512-dimensional vectors. One of the main advantages introduced by this technology is the way BERT splits words into tokens (tokenization process): it allows to split words into smaller pieces, before mapping them into vectors. This choice has the advantage of allowing the encoding of words that are not present in the dictionary and to represent all the words that share a prefix using a root vector and different representations for suffixes.

siamo sicuri sul 512?

The second and more crucial advantage introduced by BERT is the bidirectional self attention mechanism, that takes into account not only all the words that precede the target word, but also the rightmost part, giving to every word a context.

Fare digressione sull'utilizzo di BERT o BERT-large o ALBERT

As a recap, BERT (or BERT-based models) provides word-piece tokenization, a masked language model and the "next sentence" prediction, but it needs to be fine-tuned for a specific task, such as QA.

2.1.2 Fine-tuning

In this work the authors decided to fine tune the model on Natural Questions dataset.

Each training pattern is a `json` object, it is stored in a line and it has the following structure (see Figure 3):

- `document_text`: the HTML (cleaned of some tags) of the paragraph that may contain the answer;

[1] In practice, we run experiments using also ALBERT [2] and BERT Large. In the future also RoBERTa's [3] checkpoints will be used.

[2] Some qualities of NQ are the following: (1) the questions were formulated by people out of genuine curiosity or out of need for an answer to complete another task, (2) the questions were formulated by people before they had seen the document that might contain the answer, (3) the documents in which the answer is to be found are much longer than the documents used in some of the existing question answering challenges.

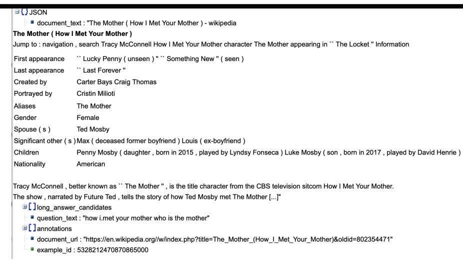
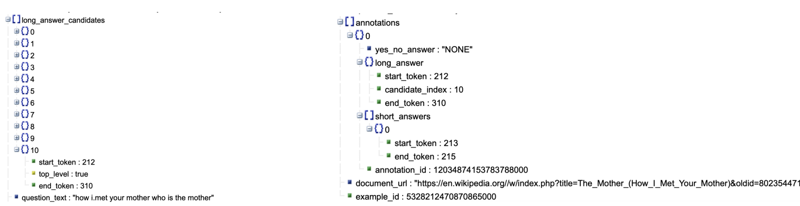


Figure 3: Broad structure of an input pattern.

- ◇ `long_answer candidates`: contains the original question and a list of start and end positions of candidates for the answer (an example in Figure 4a);
- ◇ `annotations`: contains three sub-objects, that represent if the question allows a "yes-no" answer, the information about the short answer and the information about the long answer respectively (as shown in Figure 4b). It is possible that a question does not allow to be answered looking at the paragraph given as input. In that case the fields in the `long_answer` object have value -1 and the list `short_answers` is empty.



(a) `long_answer candidates`. **(b)** `annotations`.

Figure 4: More details about the fields of an input pattern.

Cercare di capire che cosa rappresenta il campo `top_level` di `long_answer candidates`

Ho capito che cosa rappresenta `candidate_index`, e' l'indice della long answer vincente nell'elenco, vedi json di esempio

The NQ training set has on average, input patterns with size of 10MB and the whole training set is stored in a `.jsonl` file of size $\approx$ 17 GB. It goes without saying that loading both BERT's checkpoints and such file into RAM is not possible, so we managed to overcome this problem by splitting the file into chunks with size smaller than 100MB.

Another crucial characteristic of this dataset is that is strongly unbalanced towards the questions that are unanswerable:

> *In total, annotators identify a long answer for 49% of the examples, and short answer spans or a yes/no answer for 36% of the examples. We con-*

sider the choice of whether or not to answer a question a core part of the question answering task, and do not discard the remaining 51% that have no answer labeled.[4]

This issue of unbalanced data needs some more reasoning: as already discussed in Section 2.1.1, the BERT-based embedder maps the input text into vectors. It also performs another operation, namely it splits the paragraph in the so-called "crops", containing 512 tokens. It is easy to conclude that most of these crops do not contain the answer to the original question, hence the umbalancing becomes even more severe ($\approx 95\%$ of the crops do not contain the answer).

The authors overcame this issue by selecting only the 3% of the so-called "impossible questions"[3] and by designing an effective yet simple loss function.

Rifare figura

On a batch of examples, we computed the loss as follows:

$$\begin{cases} \dfrac{\dfrac{\left(\operatorname{avg}_{\text{batch}} \text{start\_loss} + \operatorname{avg}_{\text{batch}} \text{end\_loss}\right)}{2} + \operatorname{avg}_{\text{batch}} \text{long\_loss}}{2} & \text{if answerable} \\ 0 & \text{otherwise} \end{cases}$$

where *_loss is the categorical cross-entropy between the position of the true answer and the position of the guess, as shown in Figure 5.

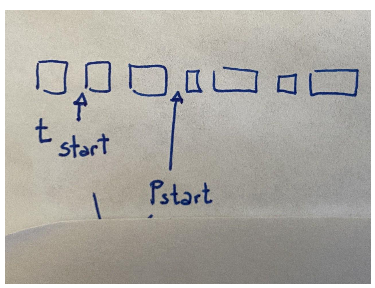


Figure 5: A pictorial representation of how the position of the first token of the answer may vary between target and predicted value.

Depending on the flag start position and end position of the answer in the given paragraph.

3 Experimental results

3.1 Hardware

[3]The code is available on GitHub, we set the value 0.03 to the parameter called `p_kepp_impossible` in `dataset_utils_version2.py`.

3.2 Hyper-parameters values

Parameter	Values
η ("learning rate")	$1e-4$, $1e-5$, $1e-6$
β_1	0.9
β_2	0.999
ε	$1e-7$
batch size	8 (ALBERT), 4 (BERT)

Table 1: Hyperparameters values.

4 Conclusions and future work

We describe **nomeFicoDaSceglière**as an interactive question answering platform that sees its application in various contexts, such as domotics or education. We performed experimental results and validations techniques that assess the goodness of this model, although we highlight some issues that could be solved more effectively in the future, such as the problem of the presence of a typo in the query (both human error in typing or machine speech-to-text misunderstanding)[4]. Moreover, we would like to stress that choosing to force the model to learn only the answerable question was only a first approach to solve the problem and that another technique worth deepening in the future may be using a binary classification layer to check if the answer is "plausible" (i.e. the meanings in the question are covered in the answer), as done by [5] and [6].

4.1 All references

- kbqa [7]
- coqa [8]
- collobert [9]
- vaswani [10]
- weston1 [11]
- weston2 [12]
- alberti 2019 [13]
- kwiatowski 2019 [4]
- chen [14]
- liu purple [15]
- liu yellow [16]

[4] One could try to use library already implemented in Python for auto-correction, such as `auto-correct` available on Pypy, or a more involved and more accurate dictionary-based auto-correct strategy.

◇ RoBERTa [3]

◇ ALBERT [2]

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[2] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 07 2019.

[4] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[5] Minghao Hu, Furu Wei, Yu xing Peng, Zhen Xian Huang, Nan Yang, and Ming Zhou. Read + verify: Machine reading comprehension with unanswerable questions. In *AAAI*, 2019.

[6] Seohyun Back, Sai Chetan Chinthakindi, Akhil Kedia, Haejun Lee, and Jaegul Choo. Neurquri: Neural question requirement inspector for answerability prediction in machine reading comprehension. In *International Conference on Learning Representations*, 2020.

[7] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. Kbqa: Learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10:565–576, 01 2017.

[8] Siva Reddy, Danqi Chen, and Christoper Manning. Coqa: A conversational question answering challenge. 08 2018.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is

all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[11] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann Lecun. Tracking the world state with recurrent entity networks. 12 2016.

[12] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. 03 2017.

[13] Chris Alberti, Kenton Lee, and Michael Collins. A bert baseline for the natural questions. 01 2019.

[14] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Bidirectional attentive memory networks for question answering over knowledge bases. pages 2913–2923, 01 2019.

[15] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. 01 2019.

[16] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. 04 2019.