Independent Research Study
-
Query based paragraphs classification

Vincent Chabot, Urjit Patel
Center for Data Science - New York University
{vec241, up276}@nyu.edu

Abstract

As part of the Automated Researcher project, our goal in this independent study was, given documents retrieved by querying a search engine, to build a framework capable of selecting from these documents the particular paragraphs that are specifically relevant to the initial query, that is to say that answer or contain part of the answer to the initial query. In order to do so, and given the lack of such explored task in the literature, we first built our own data set that would best replicate this situation. Then, we explored the literature in natural language sequence matching (NLSM) and implemented, adapted and improved a various number of promising models. Given the initial performances of our baseline models that have shown our task to be particularly challenging and difficult, we were able to greatly improve our results with more elaborated models.

1 Introduction and related work

1.1 The Automated Researcher Project

Our independent research study is part of the Automated Researcher project, led by Rodrigo Nogueira under the supervision of Professor Kyunghyun Cho. The objective of this project is to provide a comprehensive answer to non trivial questions, given a database of documents for which we suppose one or more documents are related to the question (be it document databases such as arXiv, Wikipedia or even the web). By non trivial question, we mean question that cannot be answered in one or few words, such as factual question like ”who was the 4th President of the United States?”. The ultimate objective would be to be able to provide comprehensive answers to questions as complex as ”What are the most promising direction to cure cancer and why?”.

In order to provide an answer to a query from a database of documents, the framework is the following: first, retrieve the documents from the database that are related to the query and that will allow to answer it. Then, answer the query by selecting only the parts of the documents that specifically contain elements of answer to the query (typically paragraphs) and finally synthesize these paragraphs into a comprehensive answer. The answer can be built either by extractive summarization (selecting the most relevant paragraphs while not including redundant ones) or abstractive summarization (given the selected paragraphs, produce an answer using a generative model).

Significant work has already been done till date on the first part, which requires building an efficient search engine that retrieves the relevant documents. In particular, Nogueira et al. (2017) [1] introduce a query reformulation system based on a neural network trained with reinforcement learning that rewrites a query to maximize the number of relevant documents returned. The later parts of the framework still need to be built.

1.2 Extracting from documents paragraphs that answer a specific question

Given a query and documents that are related to the query (typically documents that would be retrieved by the search engine in the first step), our objective in this research study consists in building a model able to filter out from documents (i.e. select) the paragraphs that are relevant to the query, that is to say that brings an answer or a part of the answer to the query.

1.3 Related Work

Natural language sentence matching (NLSM) is a task encountered in many natural language problems, from natural language inference (Bowman et al.,2015 [2]), in which the task consists in inferring the nature of a relationship between two sentences (entailment, neutrality or contradiction), to paraphrase identification (Yin et al., 2015 [3]). The key task in NLSM consist in assessing the semantic relatedness between the sequences of interest.

NLSM is also widely used in some question answering tasks, such as in the case of question-answer pairs classification, that is to say for which we want to determine if the answer is relevant to the query or not. Several frameworks have already been proposed for NLSM, such as frameworks based on Siamese architectures (Bromley et al., 1993 [4]), which are networks where both text sequences are encoded into a vector using the same neural network, and then the decision is made by comparing these two vector by either measuring the similarity between the two vectors obtained, or by concatenating the two obtained vectors and feeding them into a multilayer perceptron (Bowman et al., 2015 [2], Tan et al., 2015 [5]). Siamese architecture allow to share the weights and ease the training, but encoding both sequence without any interaction between the two and only doing the comparison phase once both sequences are encoded is an important limitation and might result in losing relevant information. Therefore, more advanced techniques, such as ones using soft-attention mechanism (Bahdanau et al., 2016 [6]), have been introduced (Hermann et al., 2015 [7]). A new framework called "compare-aggregate" was recently introduced by Wang et al. (2016 [8]), in which the comparison between two sequences is done by comparing smaller units such as words or contextual vectors, and then aggregating these comparison results to take a final decision (by CNN or LSTM).This framework, by capturing low level features interactions, results into significant improvements.

In our independent study, we propose to implement and adapt different models explored in the literature to evaluate their performance on a new data set of NLSM which is more relevant to the Automated Researcher project, that is to say matching queries with only the relevant paragraphs from all the paragraphs of a document that was obtained from a search engine using this query. Therefore, the greatest difficulty comes from the fact that all the paragraphs are somehow related to the topic of the query but only few paragraphs are relevant to answer the query.

2 Models

2.1 Problem definition

Our task is the following supervised, binary classification problem: given a query paragraph pair, for which we know if the paragraph is relevant to the query or not, classify the pair. Hence, we can represent each sample by a triple $(query, paragraph, y)$ where $query = (q_1, ..., q_{n_q})$, $paragraph = (p_1, ..., p_{n_p})$, and $y = 0$ or 1, where n_q and n_p are respectively the number of words in the query and in the paragraph,

For every model, we output a final score in R^2 for which we take the softmax to get a probability distribution. Then, the final predicted label 0 or 1 is just decided by taking the argmax of that score in $R^2 : y^* = argmax_{y \in \{0,1\}} Pr(y|query, paragraph)$.

2.2 Baselines

In order to assess the efficiency of our models, we build 3 baselines. For each baseline, we represent each question and paragraph by their Bag Of Words vector representation, denoted q and p, that is q is the average of the vector representations of the words in the query, and similarly for p. Words are

primarily embedded using GloVe pretrained vectors[1] from Pennington et al. (2014) [9]. Therefore, we have that $(q, p) \in R^d$, where d is the dimensionality of the word embeddings.

2.2.1 Baseline #1 : Bi-linear Product

The first baseline model is simply the bi-linear product between bag-of-words vectors q and p, that is to say $p^T W q$ where W is a weight matrix $W \in R^{d \times d}$. In practice, in order to obtain a final score in R^2, we define a tentsor $W^{[1:2]} \in R^{d \times d \times 2}$ and we compute the bilinear tensor product $p^T W^{[1:2]} q$, where each entry is computed by one slice i = 1 or 2 of the tensor: $s_i = p^T W^{[i]} q$. Therefore the final (unnormalized) scores (in R^2) are :

$$scores = p^T W^{[1:2]} q$$

2.2.2 Baseline #2 : Multilayer perceptron over concatenation of q and p

For our second baseline model, we concatenate the Bag Of Words vectors q and p, and pass it through a Multilayer perceptron, with 3 layers and ReLu as activation function. Therefore the final (unnormalized) scores are :

$$scores = MLP([q, p])$$

where [,] stands for concatenation and MLP for multilayer perceptron.

2.2.3 Baseline #3 : Multilayer perceptron over q and p square point wise difference and point wise multiplication

For our last baseline model, we define two operations following Wang et al. (2016) [8] :

$$Substraction(SUB) : s = (q - p) \odot (q - p)$$
$$Multiplication(MULT) : m = q \odot p$$

where $\odot$ is the element wise multiplication. Therefore, we have two operation that capture some distance notion, the SUB operation being closely related to the Euclidian distance while the $MULT$ operation is closely related to the cosine similarity.

Finally, we concatenate the resulting vectors s and m, and pass it through a Multilayer perceptron, with 3 layers and ReLu as activation function, so the final (unnormalized) scores are :

$$scores = MLP([s, m])$$

2.3 CNN or RNN + attention + concatenation or SUB-MULT concatenation + MLP models

Second, we define several models that all share a same core attention step using a vector representation of the query to obtain a vector representation of the paragraph. As mentioned in the introduction, the attention mechanism allows to take into consideration some knowledge of a given sequence (here the question) in the generation of the final vector representation of the other sequence (here of the paragraph) and this before the final comparison step of the two sequences. The general framework is the following :

1. For both query and paragraphs, we embed the words using GloVe pretrained vectors from Pennington et al. (2014) [9]. At this point, the query and the paragraph are represented by two matrices Q and P of dimension respectively $d \times n_q$ and $d \times n_p$, where n_q and n_p are respectively the number of words in the query and in the paragraph, and d is the embedding dimension.
2. We process Q and P through a preprocessing layer that can either be a) a convolution network or b) a BiLSTM and keeping all hidden states, to obtain two new matrices ($\bar{Q}$, $\bar{P}$) $\in$ ($\mathrm{R}^{l \times n_q}$, $R^{l \times n_p}$) where l is either the number of filters in a) or the dimension of the concatenated hidden states from forward and backward RNNs in b). The objective here is to obtain a new embedding vector for each word that captures some contextual information in addition to the word itself.

[1] https://nlp.stanford.edu/projects/glove/

3. From $\bar{Q}$, we obtain a vector representation of the query $q \in R^l$ by performing either max pooling if the preprocessing layer was a convolution, or by concatenating the last hidden states of the forward and the backward RNNs if the preprocessing layer used was a BiLSTMs.
4. From $\bar{P}$, we obtain a vector representation of the paragraph $p \in R^l$ by using q to perform a soft-attention mechanism over $\bar{P}$. That is to say, we use q and $\bar{P}$ to obtain n_p weights, and then p is simply the weighted sum of the columns of $\bar{P}$ using these weights. More precisely, if we call the column vectors of $\bar{P}$ as $\bar{p_i} \in R^l$, each vector $\bar{p_i}$ being the i^{th} column of $\bar{P}$ and representing the context vector of the i^{th} word in the paragraph, then p is computed as follow :

$$p = \sum_i^{n_p} \alpha_i \bar{p_i}$$

$$\alpha_i = \frac{\exp(q \cdot \bar{p_i})}{\sum_k^{n_p} \exp(q \cdot \bar{p_k})}$$

where $\cdot$ stands for the dot product.

5. Then, as for baseline models #2 and #3, we define two possible scoring architectures :
   either $scores = MLP([q, p])$
   or $scores = MLP([s, m])$ with $s = SUB(q, p)$ and $m = MULT(q, p)$.

We can use either RNN or CNN for preprocessing layer and also we have two scoring layers as well, we obtain 4 different models all based on the same central attention layer.

2.4 CNN or RNN + attention + compare + aggregate models

Following Wang et al. (2016) [8], we modify our previous models by both doing attention over $\bar{P}$ using the word context vectors from the query and not only the final vector representation of the query q as previously, and by adding a comparison and an aggregation layer as defined in Wang et al. (2016) [8]. The two first steps to obtain $\bar{Q}$ and $\bar{P}$ are the same as explained in the previous section. The framework is the following :

1. Same as 1. from section 2.3
2. Same as 2. from section 2.3
3. Attention : we use the column vectors of $\bar{Q}$, that we call $\bar{q_j} \in R^l$, each vector $\bar{q_j}$ being the j^{th} column of $\bar{Q}$ and representing the context vector of the j^{th} word in the query, to compute attention weights over the columns of $\bar{P}$. As a result, we obtain n_q vectors h_j, one for each column vector $\bar{q_j}$ in $\bar{Q}$, which are attention-weighted sums of the column vectors of $\bar{P}$:

$$h_j = \sum_i^{n_p} \alpha_i \bar{p_i}$$

$$\alpha_i = \frac{\exp(\bar{q_j}.\bar{p_i})}{\sum_k^{n_p} \exp(\bar{q_j}.\bar{p_k})}$$

4. Comparison : for each $(\bar{q_j}, h_j)$ pair, we compute $s_j = SUB(\bar{q_j}, h_j)$ and $m_j = SUB(\bar{q_j}, h_j)$, with $(s_j, m_j) \in (R^l, R^l)$, and then:

$$t_j = tanh(W[s_j, m_j] + b) \in R^{2l}$$

where W is a weight matrix in $R^{l \times 2l}$ and b a bias vector in R^l.

5. Aggregation : finally, we concatenate all the vectors t_j to form a matrix in $R^{l \times n_q}$ and process this matrix through a one layer CNN (convolution + ReLu + max pooling + fully connected layer) to perform the final classification.

Similarly as in section 2.3, because we use two possible preprocessing layer with either a convolution operation or a BiLSTM, we obtain two models on which we evaluate the performances.

3 Experiments

3.1 Data

We got access to the TREC CAR dataset[2], which is large collection of knowledge articles from Wikipedia that are divided into content passages and outlines. Across all articles, initially all extracted paragraphs are provided as one large passage corpus in CBOR-encoded archives (CBOR is similar to JSON, but it is a binary format that compresses better and avoids text file encoding issues) to preserve hierarchical structure and entity links. The data was divided in five parts or folds, each fold has paragraphs and section details of randomly selected set of Wikipedia articles. We used one fold, which contains more than 600,000 paragraphs from various Wikipedia articles. Each fold we had below files,

$*.cbor$: Full articles as derived from Wikipedia

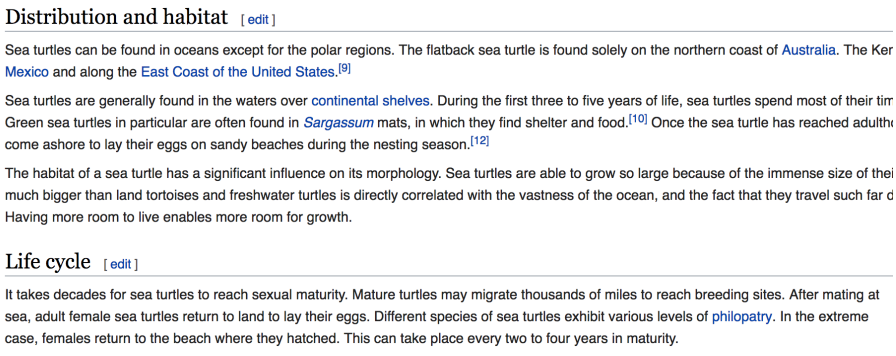
$*.cbor.paragraphs$: This file has all paragraphs coming from various articles with unique paragraph id

$*.cbor.articles.qrels$: This file has relevant paragraphs id mapping with the article associated with that paragraph

$*.cbor.toplevel.qrels$: This file has relevant paragraphs id mapping with the article top level sections associated with that paragraph This data was not prepared specifically for our task, hence we had

to prepare our own training and evaluation set using this data. From the trec car data, we had information about wikipedia articles, section topics associated with those articles, and paragraphs associated with these section topics. Our goal was to prepare a pairs of (query,paragraph) from wikipedia articles and label would be 1 if the paragraphs is relevant for the query or 0 if not relevant.

We created our queries by concatenating the article name and the section name. True paragraphs for this concatenated query would be the paragraphs falling under this section and false paragraphs would be the paragraphs from other section of the same articles. We used the paragraphs from the same article for the false pairs, as in real search engine we are suppose to get the output paragraphs related to our query only, hence we have to use relevant paragraphs for false pairs. This makes the identification task much more difficult for the model as false paragraphs would also have some relevant information. Let's take the example of the Sea Turtle article and see how we created our training set.



Distribution and habitat [edit]

Sea turtles can be found in oceans except for the polar regions. The flatback sea turtle is found solely on the northern coast of Australia. The Ker
Mexico and along the East Coast of the United States.[9]

Sea turtles are generally found in the waters over continental shelves. During the first three to five years of life, sea turtles spend most of their tim
Green sea turtles in particular are often found in *Sargassum* mats, in which they find shelter and food.[10] Once the sea turtle has reached adulth
come ashore to lay their eggs on sandy beaches during the nesting season.[12]

The habitat of a sea turtle has a significant influence on its morphology. Sea turtles are able to grow so large because of the immense size of thei
much bigger than land tortoises and freshwater turtles is directly correlated with the vastness of the ocean, and the fact that they travel such far d
Having more room to live enables more room for growth.

Life cycle [edit]

It takes decades for sea turtles to reach sexual maturity. Mature turtles may migrate thousands of miles to reach breeding sites. After mating at
sea, adult female sea turtles return to land to lay their eggs. Different species of sea turtles exhibit various levels of philopatry. In the extreme
case, females return to the beach where they hatched. This can take place every two to four years in maturity.

Figure 1: Paragraph screenshot from wikipedia article : Sea Turlte.
Here, the "Distribution and habitat" section has three paragraphs.

Query Q: Distribution and habitat of Sea turtle

True paragraph P1:

[2]http://trec-car.cs.unh.edu/datareleases/index.html

"Sea turtles can be found in oceans except for the polar regions. The flatback sea turtle is found solely on the northern coast of Australia. The Kemp's ridley sea turtle is found solely in the Gulf of Mexico and along the East Coast of the United States."

False paragraph P2:

"It takes decades for sea turtles to reach sexual maturity. Mature turtles may migrate thousands of miles to reach breeding sites. After mating at sea, adult female sea turtles return to land to lay their eggs. Different species of sea turtles exhibit various levels of philopatry. In the extreme case, females return to the beach where they hatched. This can take place every two to four years in maturity."

Hence pair (Q,P1) would have label 1 and pair (Q,P2) would have label 0. This way we created 300K total pairs, 150K true pairs and 150K false pairs on which we trained our model.

3.2 Settings

We used the GloVe pre-trained embeddings from Pennington et al. (2014) [9]. For our experiments we used the pretrained embeddings of dimension 50. We observed many small paragraphs having only 3 to 4 words, which doesn't convey any information and adds noise, hence we removed paragraphs having a length less than 10. Apart from that, we also observed that, most of the paragraphs contain highly relevant information in their first few lines, hence we cut off the paragraphs at 50 words for computational efficiency. By doing so, we were able to increase the computation speed and also reduce the memory cost. After performing mentioned data cleaning, our vocab size was about 150K words. If a paragraph length is less than 50 words, then we just append with zeros to be consistent with the size, and similarly for the queries. We used Prince GPU instance available at NYU for all our experiments, which gave is high computation speed. Here below are the other hyper parameters information:

Parameter	**Value**
Learning rate	0.001
Embedding dimension	50
Batch size	50
max doc length	50
min vocab freq	3
Gradient Clipping limit	5
loss function	softmax cross entropy with logits
Initializer	Xavier initializer
Optimizer	Adam optimizer

3.3 Evaluation

For each experiment, we report accuracy, precision and recall our models achieve. Accuracy is important to get a sense of the overall classification performance for it provides information of the proportion of samples well classified by the model. Recall is important to measure the document retrieval performance, whereas in order to minimize the possibility to miss any relevant documents that might contain part of the answer, precision is the most important metric at this point for selecting noise would result in providing a final non-relevant answer.

Accuracy is the fraction of paragraphs classified correctly as true or false:

$$\text{Accuracy} = \frac{\text{Number of paragraphs classified correctly}}{\text{Total classified paragraphs}}$$

Precision is the fraction of paragraphs classified as true, that are relevant to the query:

$$\text{Precision} = \frac{\text{Number of paragraphs classified as relevant which are truly relevant}}{\text{Total number of paragraphs classified as relevant}}$$

Recall is the fraction of the paragraphs that are relevant to the query that are classified as true:

$$\text{Recall} = \frac{\text{Number of relevant paragraphs classified correctly}}{\text{Total number of relevant paragraphs}}$$

3.4 Results

The training time is reported per epoch. There are about 237K training samples per epoch.

Model	Accuracy	Precision	Recall	Training time
Baseline 1 (bi-linear Product)	0.57	0.56	0.60	1 min/epoch
Baseline 2 (simple concatenation + MLP)	0.60	0.58	0.72	1.5 mins/epoch
Baseline 3 (pointwise diff and mul + MLP)	0.58	0.58	0.59	3 mins/epoch
CNN + Attention+ concat + MLP	0.62	0.60	0.72	3.5 mins/epoch
CNN + Attention+ pointwise diff and mul + MLP	0.63	0.60	0.71	4 mins/epoch
CNN + Attention+ compare and aggregate	0.66	0.66	0.66	7 mins/epoch
RNN + Attention+ concat + MLP	0.72	0.71	0.75	12 mins/epoch
RNN + Attention+ pointwise diff and mul + MLP	0.73	0.74	0.72	13 mins/epoch
RNN + Attention+ compare and aggregate	**0.74.5**	**0.75**	**0.75**	15 mins/epoch

With our baseline models, we were not able to achieve more than 0.6 accuracy. Precision and recall were also low for our baseline. Implementing CNN with attention for query and paragraph was able to give us good improvement in terms of all three metrics. Best results with CNN based model was 0.66 of accuracy, precision, and recall. Replacing CNN with RNN for pre-processing query and paragraphs gave us surprising jump in the accuracy. With our best RNN model, we got around 0.75 of accuracy, precision, and recall for this complex classification task. Only

One thing to note here is the time taken during training of different models. Baseline models are really fast as we have less number of parameters and the networks are smaller. As expected, CNN based models were much faster than RNN based models. Where best CNN based model (CNN + attention + compare + aggregate) takes around 7 to 8 minutes to finish one epoch on NYU GPU cluster, RNN based model (RNN + attention + compare + aggregate) takes almost 15 minutes to finish one epoch.

4 Conclusion

4.1 Conclusion of the study

In this research study paper we presented our different approaches to detect the good paragraphs from the web search engine retrieved document for the user query. We uniquely created a data set, pairs of query and paragraph, from the TREC data set. We were able to get maximum accuracy of 73% for the difficult task of detecting only required good paragraphs from the pool of all related paragraphs. We observe that RNN based attention models outperforms the CNN based attention models with high margin of accuracy. But on the other hand, RNN based models are much more slower than CNN based models. For the particular task of information retrieval, paragraph retrieval time could be important for user satisfaction. Our models can be directly plugged to a search engine system capable to return paragraphs. From our model, some simple additional steps could be added in order to provide a summarized answer. One could rank the paragraphs selected by the model by relevancy to the question as well as remove the redundant paragraphs, in order to provide an extractive summary answering the initial query. Due to the time constraint, we were not able to analyze the performance of other frameworks used in some other natural language sequence matching tasks such as Memory Networks.

4.2 Learning takeaways

After having completed the Deep Learning and Computational Approaches to Natural Language Processing classes, our objective was to apply what we learnt in these classes and to gain meaningful hands on experience in NLP, domain in which we wish to specialize, as well as to complete the last semester of our MSDS by working on a challenging real world problem.

Our insightful discussions with NYU academics as well as all the readings we had to do in order to come up with an effective solution to our task allowed us to acquire a good knowledge of the state of the art in NLP in this domain. Also, we feel that we have acquired valuable knowledge for our future career in conducting an end-to-end RD project, from task definition to data acquisition /

transformation, to building appropriate models to answer our problem given our prior knowledge and literature we could find on the topic. Lastly, we also feel that this research project greatly contributed to let us reach our goals since we could feel the great interest of recruiters for this project during all the interviews we had. We will both join highly research oriented departments with Deep Learning roles after NYU Graduation.

4.3 Acknowledgements

First of all, we would like to thank particularly R. Nogueira for his always very helpful advice. It was a real pleasure to work with him on such an interesting and promising project and we really enjoyed our weekly always insightful discussions. We would also like to thank S. Lauly for his interest in our work, his help and valuable feedback. Finally, we would like to thank Professor K. Cho for his supervision and for giving us the opportunity to join this research project.

References

[1] Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572*, 2017.

[2] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[3] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.

[4] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

[5] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[8] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

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