Nikos Gounakis , HY563 Project , csdp1254

1) & 2)

see `collection.py` . Performs an HTTP request to fetch the collection and stores it in a class.
There is also a method implemented that mimics the baseline (predict random words from the text)
see `metrics.py` for the `F1` and `EM`

3) Question Type Prediction & Entity Type Prediction

see `sklearn` folder, `collection.py` and `WekaApi.py`

We used a machine learning approach to predict the question and the entity type. In order to produce training data for each of the two problems we used the questions from the provided collection along with their labels (target classes). We feed a sentence-transformer (`sentence-transformers/all-MiniLM-L6-v2`) model with the question to get numeric features and we export a weka file containing the features and the corresponding label for each sample. So at the end we have two weka files, one with features and question types and one with features and entity types.

Note: We merged the entity types in questions that had two entity types.

We used the `AUC` metric because we have a multiclass classification and class imbalances.
For training method we used 10-fold cross validation.

We had some problems using the exported Weka model file in python, so we used Weka only to compare the classifiers. Then using sklearn we trained and exported the models to have them ready for future use. In the following tables we can see the results. It seems that RandomForest predicts better the question type and SMO predicts better the entity type.

Question Type Prediction

Model	AUC
Vote	0.486
RandomForest	**0.792**
SMO	0,770

Entity Type Prediction

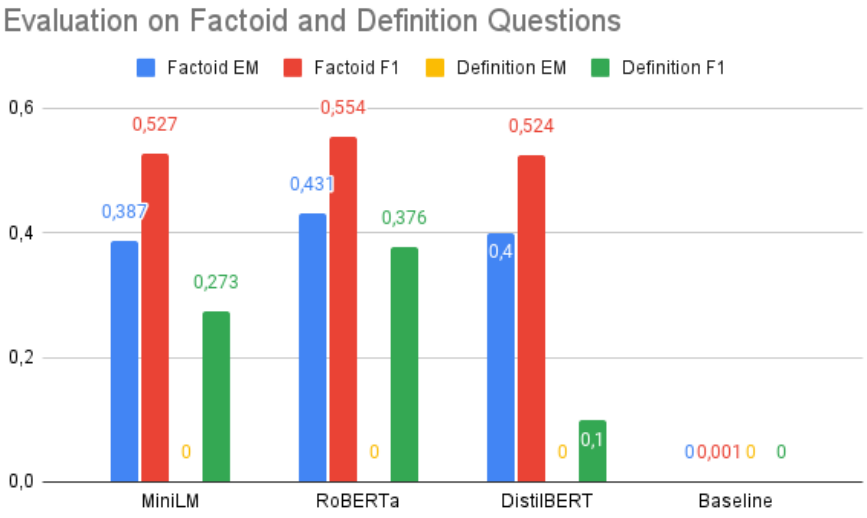
Model	AUC
Vote	0,432
RandomForest	0,796
SMO	**0,820**

4) & 5) Answer Extraction

In this stage we compare pretrained neural models for the task of answer extraction.

We used 3 pretrained models from huggingface, minilm , roberta and distilbert. See the `answer_extraction` module for the code.

First we performed an evaluation and considered the `Factoid` and `Definition` questions. The Metrics we used are `EM` (Exact Match) and `F1` .



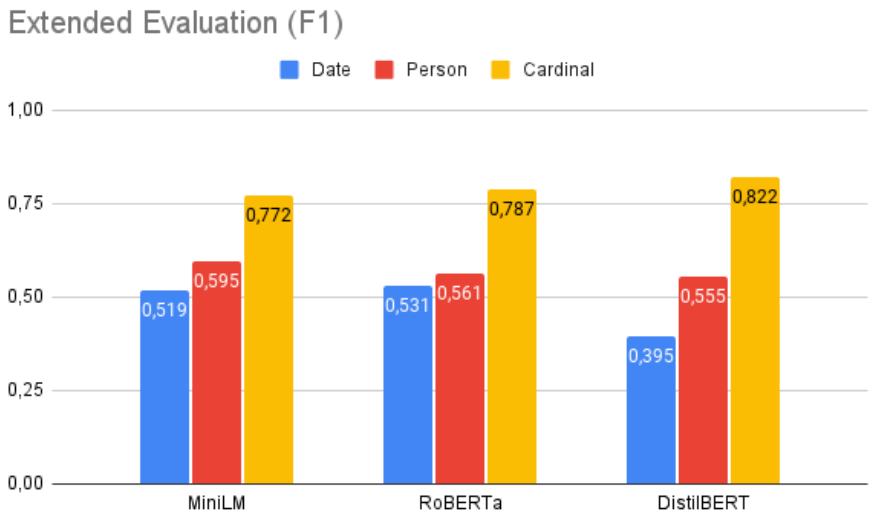
We see that the models surpass the baseline as the baseline achieve almost zero score. We can clearly see that RoBERTa model is outperforming the rest achieving the highest F1 (0.554) in factoid questions as (0.376) the definition questions. Furthermore, we see that in definition question none of the models achieved non-zero score and that may be due to the difficulty of the definition questions.

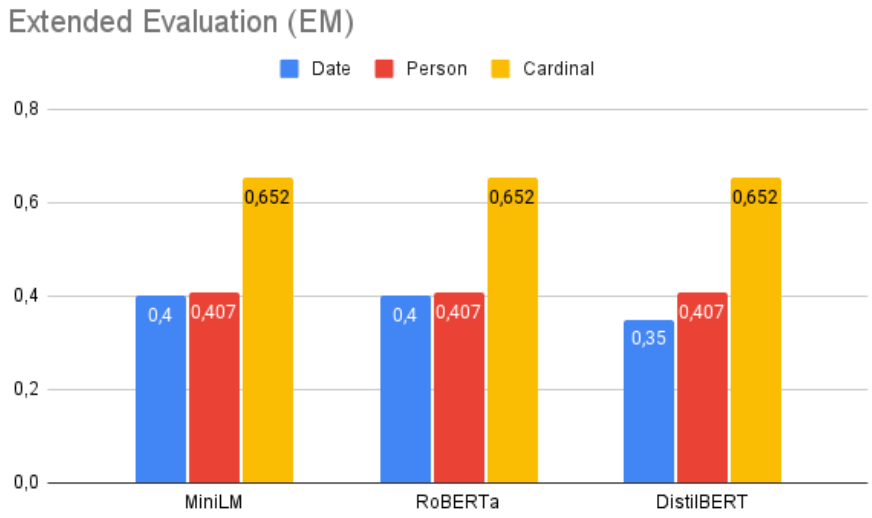
We also measure the time of the evaluation for each model.

Model	Time (s)
MiniLM	76.4
RoBERTa	167.4
DistilBERT	89.5

MiniLM has scores close to RoBERTa and it is almost 2 times faster in evaluation time.

We extend further the evaluation by considering Factoid question of type `Person` , `Date` and `Cardinal`





As we can see now each model achieves the best F1 score in a specific question category. (Date-RoBERTa, Person-MiniLM, Cardinal-DistilBERT). In the Exact Match visualization we see that the models have almost the same scores. That may be caused because the questions that the models achieved exact match were too easy, so the difficult ones failed in all models.

6) Confirmation Questions

In this chapter we evaluate 2 methods for the confirmation question.

1. We use the sentence similarity (`sentence-transformers/all-MiniLM-L6-v2`) to create embeddings for question and passage and `deepset/roberta-base-squad2` for answer extraction. The pattern is to get the similarity of the embeddings of the answer and the topic text
2. We use `cross-encoder/qnli-distilroberta-base` to get a score using the answer and the topic text

for both methods if the score is above 0.7 we consider and answer as `Yes` otherwise `No`

Method	EM	Time (s)
Embedding Similarity	**0.457**	27.1

Method	EM	Time (s)
Cross-encoder	0.4	**7.6**

With embedding similarity we achieve the highest score for confirmation questions but it is more practical to use the cross-encoder because it is almost 4 times faster.

7) Improvement

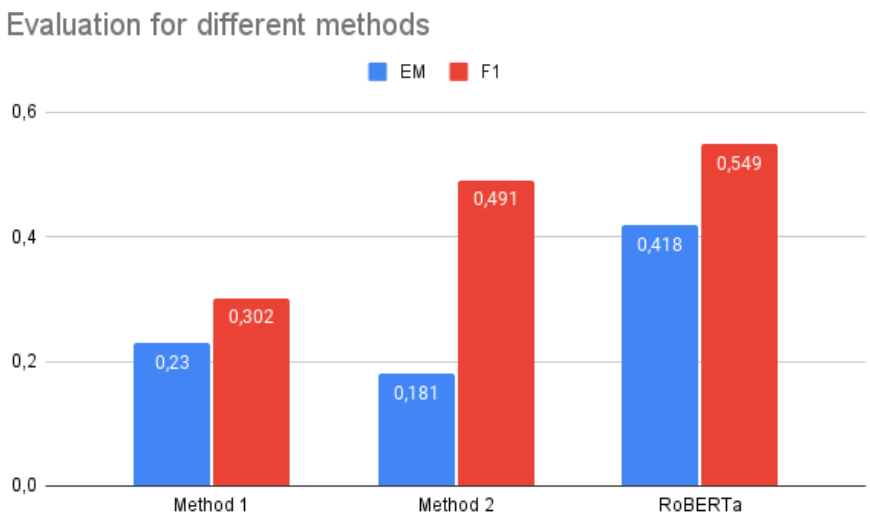
In this stage we apply a named entity recognition to the question make use of the top entity. Then we do a SPARQL query to the dbpedia to fetch the `rdfs:comment` of the found entity.

We evaluate 2 methods.

Method 1: If an entity is recognized in the question and a `rdfs:comment` exists then we consider only the `rdfs:comment` for the answer extraction (otherwise we answer from the topic text).

Method 2: We merge the above method result with the answer from the topic text.

We evaluated these methods with the same metrics as before and we used the RoBERTa model for the answer extraction.



We present the Average F1 and EM through Factoid and Definition questions. As we can see the methods seem to not help making the score better. This maybe is caused due to the fact the rdfs:comment can contain less helpful information about the question and the concatenation in `Method 2` is lowering the precision more than increasing the recall, so the F1 is dropping.

Passage Retrieval

In this section we use the `qnli-distilroberta-base` . For each question we try to predict the text from the collection that can be used for answer extraction. For this experiment we used the `Accuracy` metric. We achieved **0,606** accuracy finding he correct text. In our opinion is not very a pleasing results as it plays a crucial role in the answer extraction. Wrong text can lead to fail answering the question.

Conslusion

We can see in general that pleasant scores are achieved by the models, but further improvement must be done in order to increase the effectiveness. Some could be to train a neural model to fetch answers matching an answer type or use further SPARQL queries to enrich the content or use passage retrieval in the topic text to mark a smaller area of the topic text, that the answer could be in. For now a RoBERTA models seems more accurate but we propose the MiniLM because it is a lot faster and has not much difference in scores with RoBERTa.