Homework 3

Data Mining Technology for Business and Society

Deadline: **11 June 2021 23:59 (Rome Time Zone)**

Having **EXACTLY TWO** students per group is **RECOMMENDED**.

The total length of the report cannot exceed 4 pages.
It is forbidden to print or store this document, you can only read this document online.
It is forbidden to submit software written with Python-Notebook.
Only ".py" software is considered as a valid solution.

The software must be commented.

Data and software are available at:

Sentence-Transformers: https://www.sbert.net/

GENRE: https://github.com/facebookresearch/GENRE

Dataset(s): https://github.com/facebookresearch/KILT

To test on leaderboard:
https://eval.ai/web/challenges/challenge-page/689/overview

The main idea is to fact check claims using the knowledge that modern language models, such as BERT, acquire during pre-training.
FEVER will be used as the main dataset, sentence-transformers and GENRE as the main packages.

Part 1

In this part of the homework, you have to preprocess the FEVER dataset, use an embedding model on each sentence and train a classifier over the generated embeddings.

Part 1.1

Download the train, dev and test set from https://github.com/facebookresearch/KILT. In particular:

- http://dl.fbaipublicfiles.com/KILT/fever-train-kilt.jsonl
- http://dl.fbaipublicfiles.com/KILT/fever-dev-kilt.jsonl
- http://dl.fbaipublicfiles.com/KILT/fever-test_without_answers-kilt.jsonl

A claim should have the following structure:

```
{'id': 33078,
 'input': 'The Boston Celtics play their home games at TD Garden.',
 'output': [{'answer': 'SUPPORTS',
   'provenance': [{'bleu_score': 0.517310804423051,
     'end_character': 404,
     'end_paragraph_id': 1,
     'section': 'Section::::Abstract.',
     'start_character': 227,
     'start_paragraph_id': 1,
     'title': 'Boston Celtics',
     'wikipedia_id': '43376'}]}]}
```

Preprocess all claims in the FEVER train, dev and test set as follows:

1. Keep just the first entry in the output list and discard all keys from the “output” list, except for “answer”. Each datapoint should now have this structure:
   ```
   {'id': 33078,
    'input': 'The Boston Celtics play their home games at TD Garden.',
    'output': [{'answer': 'SUPPORTS'}]}
   ```
   For the test set there is no “output” list

2. Use the sentence-transformers python library to get sentence embeddings for the 'input' field of each claim. It is mandatory to use the 'paraphrase-distilroberta-base-v1' model. Append the embedding to the claim dict:
   ```
   {'id': 33078,
    'input': 'The Boston Celtics play their home games at TD Garden.',
    'output': [{'answer': 'SUPPORTS'}],
   "claim_embedding": array(...)}
   ```

3. Save the results in the files "emb_train.jsonl", "emb_dev.jsonl", "emb_test.jsonl"

Part 1.2

Using the claim embeddings as input vectors, representing the labels as 0 ("REFUTES") and 1 ("SUPPORTS"), you must train at most two binary classifiers (e.g. SVM) on the train set, then get the performance on the dev set. It is mandatory to finetune the hyperparameters of the models (using e.g. GridSearch or RandomSearch).

Choose a valid metric to evaluate the performance of the classifier(s).

Results for 1.2

By using **at most two** pages of the report, you must:
.) put in the report the complete “Grid-of-Parameters” you used to increase the performances for each classifier.
.) put in the report the best configuration you found for each classifier.
.) for each classifier, put in the report the confusion matrix computed on the dev set associated to the best hyperparameter configuration. Confusion matrix should have the following structure:

CLASSIFIER N	PREDICTED REFUTES	PREDICTED SUPPORTS	RECALL
ACTUAL REFUTES	TN	FP	TNR (RECALL ON REFUTES)
ACTUAL SUPPORTS	FN	TP	TPR (RECALL ON SUPPORTS)
PRECISION	NPV (PRECISION ON REFUTES)	PPV (PRECISION ON SUPPORTS)	ACCURACY

where N in the first cell may be 1 or 2, depending on the classifier.

.) describe briefly (maximum 6 sentences) how you proceeded with the optimization of the hyperparameters and the results obtained.
.) if the goal was to be sure to correctly catch all REFUTES claims (i.e. label them as REFUTES), which metric should you look at? And what trivial solution could be adopted? Use at most 1 sentence.
.) for each classifier, get predictions for the official test set associated to the best hyperparameter configuration. Put the predictions in a file named “test_set_pred_N.jsonl”, where N may be 1 or 2, depending on the classifier. The jsonl file(s) should have the following structure:

```
{"id": "75397", 'output': [{'answer': "SUPPORTS"}]},
{"id": "156709", 'output': [{'answer': "REFUTES"}]},
...
```

Part 2

In this part of the homework, you have to improve the predictions using the functionalities of the GENRE python package to get additional information about the main entity of each claim.

Part 2.1

This part can take a long time to run. It is feasible to complete it by the deadline, especially if you follow the recommendations of the fifth lab; but if you encounter too many problems, perform it only on a portion of the data (use only the first X% of the train records; use the whole dev set and the whole test set anyway) and don't leave part 2.2 undone. If only a fraction of the train set is used, specify this in the report.

Retrieve the KILT knowledge source from the following link:
abstract_kilt_knowledgesource.json

For train, dev and test set:

1. Use GENRE End-to-End Entity Linking on the 'input' field of each claim. It is mandatory to use the "e2e_entity_linking_wiki_abs" pretrained model, either the transformers (recommended) or fairseq one. For each claim, using only the first result of the method, extract the Wikipedia page name found by the method for each entity. Be careful to follow the recommendations in Lab 5 regarding the use of GENRE and its possible issues.

   Example of a GENRE result for one sentence:
   [{'logprob': tensor(-0.1509),
   'text': ' The { Boston Celtics } [Boston Celtics] play their { home } [Home (sports)] games at { TD Garden } [TD Garden]'},
   {'logprob': tensor(-0.1911),
   'text': ' The { Boston Celtics } [Boston Celtics] play their { home } [Home advantage] games at { TD Garden } [TD Garden]'},
   {'logprob': tensor(-0.2203),
   'text': ' The { Boston Celtics } [Boston Celtics] play their { home games } [Home (sports)] at { TD Garden } [TD Garden]'},
   ...]
   --- The result highlighted in yellow is the one to keep.

   Append the results to the claim dictionary.

   {'id': 33078,
   'input': 'The Boston Celtics play their home games at TD Garden.',
   'output': [{'answer': 'SUPPORTS'}],
   "claim_embedding": array(...),
   "wikipedia_pages": ["Boston Celtics", "Home (sports)", "TD Garden"]}

2. For each claim, retrieve all the abstracts in the knowledge base related to the "wikipedia_pages". Order the abstracts according to their length in ascending order (from shortest to longest), then concatenate them separated by a whitespace " ". Add the resulting string to the claim dictionary. If there is no page in the "wikipedia pages" list or none of the pages are found in the KILT knowledge source, consider as if you've retrieved a string with only one blank space (" ").
   {'id': 33078,
    'input': 'The Boston Celtics play their home games at TD Garden.',
    'output': [{'answer': 'SUPPORTS'}],
   "claim_embedding": array(...),
   "wikipedia_pages": ["Boston Celtics", "Home (sports)", "TD Garden"],
   "wikipedia_abstract": "In sports, home is the place and venue identified… The Boston Celtics are an American professional basketball team… TD Garden, sometimes colloquially referred to as simply the Garden, is a…"}

3. Use the sentence-transformers python library to get sentence embeddings for the "wikipedia_abstract". It is mandatory to use the 'paraphrase-distilroberta-base-v1' model. Since the abstract is usually longer than the claim, set the embedding model max_seq_length to 256 (must be shown in the code). Append the embedding to the claim dict:
   {'id': 33078,
    'input': 'The Boston Celtics play their home games at TD Garden.',
    'output': [{'answer': 'SUPPORTS'}],
   "claim_embedding": array(...),
   "wikipedia_pages": ["Boston Celtics", "Home (sports)", "TD Garden"],
   "wikipedia_abstract": "In sports, home is the place and venue identified… The Boston Celtics are an American professional basketball team… TD Garden, sometimes colloquially referred to as simply the Garden, is a…",
   "abstract_embedding": array(...)}

4. Save the results in the files "emb_train2.jsonl", "emb_dev2.jsonl", "emb_test2.jsonl". Alternatively, you can overwrite the previous "emb_train.jsonl", "emb_dev.jsonl", "emb_test.jsonl" files.

Part 2.2

Repeat what you have done in Part1.2, but now use as input vectors the concatenation of claim_embeddings and abstract embeddings. If the previous input matrix had size (number_of_claims, embedding_size), the new concatenated matrix should have size (number_of_claims, embedding_size*2). It is mandatory to use the same classifier(s) as Part1.2, as well as the same evaluation metric.

It is important to remark that **it is not** requested to find a unique hyperparameter configuration that is good for both Part1.2 and Part2.2, but, what is requested, is to find a good ad-hoc hyperparameter configuration for each part. While it is mandatory to perform the hyperparameter configuration on the same hyperparameters as Part1.2, the parameter space may differ between the two parts.

Results for 2.2

By using **at most two** pages of the report, you must:
.) put in the report the complete "Grid-of-Parameters" you used to increase the performances for each classifier.
.) put in the report the best configuration you found for each classifier.
.) for each classifier, put in the report the confusion matrix computed on the dev set associated to the best hyperparameter configuration. Confusion matrix should have the following structure:

CLASSIFIER N	PREDICTED REFUTES	PREDICTED SUPPORTS	RECALL
ACTUAL REFUTES	TN	FP	TNR (RECALL ON REFUTES)
ACTUAL SUPPORTS	FN	TP	TPR (RECALL ON SUPPORTS)
PRECISION	NPV (PRECISION ON REFUTES)	PPV (PRECISION ON SUPPORTS)	ACCURACY

where N in the first cell may be 1 or 2, depending on the classifier.

.) describe briefly (maximum 6 sentences) how you proceeded with the optimization of the hyperparameters and the results obtained.
.) for each classifier, get predictions for the official test set associated to the best hyperparameter configuration. Put the predictions in a file named "new_test_set_pred_N.jsonl", where N may be 1 or 2, depending on the classifier. The jsonl file(s) should have the following structure:

```
{"id": "75397", 'output': [{'answer': "SUPPORTS"}]},
{"id": "156709", 'output': [{'answer': "REFUTES"}]},
...
```

There was an error in the test set prediction file format for part 2.2. The error has been corrected, but given the imminent deadline, you can leave this file as you did.
Wrong format before correction:

```
{"id": "75397", "label": "SUPPORTS"}
{"id": "62037", "label": "SUPPORTS"}
...
```

Bonus Part (optional)

This part is optional.

You can submit predictions on the test sets for FEVER at https://eval.ai/web/challenges/challenge-page/689/overview and get listed in the official KILT leaderboard.

Results for Bonus Part

Only if you do this Bonus Part, the final report may be of five pages. By using **at most one** page of the report, you must:
.) Specify which prediction file you have used for the submission on EvalAI (test_set_pred_1.jsonl, test_set_pred_2.jsonl, new_test_set_pred_1.jsonl or new_test_set_pred_2.jsonl).
.) Add a screenshot with your ranking and score on the leaderboard.

Where/What To Send

At the end of the process, you have to create a **zip** file with **ONLY** the following:

1. The software for addressing Part_1: /DMT_2020/HW_3/part_1/sw/ (.py files).
2. The software for addressing Part_2: /DMT_2020/HW_3/part_2/sw/ (.py files).
3. The Part_1 jsonl output file(s): /DMT_2020/HW_3/part_1/test_set_pred_1.jsonl /DMT_2020/HW_3/part_1/test_set_pred_2.jsonl (if you trained a second classifier)
4. The Part_2 jsonl output file(s): /DMT_2020/HW_3/part_2/new_test_set_pred_1.jsonl /DMT_2020/HW_3/part_2/new_test_set_pred_2.jsonl (if you trained a second classifier)
5. The final report in **PDF**: /DMT_2020/HW_3/report.pdf .
6. **PLEASE, DO NOT PUT THE DATASETS IN THE ZIP FILE.**

Not following the submission instructions will negatively impact your grade

The name of the zip file must have this format:
DMT_2021__HW_3__StudentID_StudentName_StudentSurname_StudentID_StudentName_StudentSurname.zip
Finally you must send the ".zip" file to siciliano@diag.uniroma1.it with the following email subject:
DMT_2021__HW_3__StudentID_StudentName_StudentSurname_StudentID_StudentName_StudentSurname.

P.S.
For any problem, doubt or consideration, please send a single email to both siciliano@diag.uniroma1.it and giorgio.barnabo@uniroma1.it.