

Homework - Deep Learning and Natural Language Processing

Data Mining Technology for Business and Society 2020

Deadline: 14 June 2020 23:59 (Rome Time Zone)

The main idea is to fact check claims using the knowledge that modern language models, such as BERT, acquire during pre-training (for BERT, by reading Wikipedia and BookCorpus) [0]. We will use FEVER [1] as the main dataset and LAMA [2] as the main repository.

[0] Language Models as Knowledge Bases? <https://arxiv.org/abs/1909.01066>

[1] <https://fever.ai/resources.html>

[2] <https://github.com/facebookresearch/LAMA>

For each task the submission should include:

1. A brief document describing what you did and the choices you made (max 2 pages)
2. The code of your scripts, 4 scripts (task1.1.py, task1.2.py, task1.3.py, task2.py)
3. The best measured accuracy on the dev set, for all tasks (i.e., t1.1, t1.2, t1.3 and t2; 4 numbers to report)
4. A file with the prediction on **the official test set** (1082 examples), in jsonl format, corresponding to the best dev accuracy (i.e., not for all task, just for the strategy/task that gave you the best accuracy on the dev set)

Example of a prediction file to submit:

```
{"id": "78526", "label": "REFUTES"}  
{"id": "62037", "label": "SUPPORTS"}  
...
```

official test set link: [Piazza link](#)

Evaluation metric: accuracy, number of times your model outputs the correct prediction

Task 0. Preprocessing

Preprocess all claims in the FEVER train set [3] and dev set [4]:

1. Discard all datapoints with label NEI (not enough information). Keep only datapoints with label SUPPORTED or REFUTED.
2. Use flair [5] Named Entity Recognition to get all entities for all claims (i.e., sentence) in train and test.
3. Consider only entities with a single token (no space and no minus sign), that are contained in the BERT vocab [6]
4. Consider only datapoints with **exactly** one entity with a **single token** in the BERT vocab.

5. Add the entity information, including start (inclusive) and end (exclusive) character in the original claim, to each datapoint.

Example:

```
{ "id": "...",  
  "label": "SUPPORTS",  
  "claim": "Oliver Reed was a film actor." } -> DISCARD, Oliver Reed  
is an entity, but not single token
```

```
{ "id": "...",  
  "label": "NOT ENOUGH INFO",  
  "claim": "Henri Christophe is recognized for building a palace in  
Milot." } -> DISCARD, not enough info
```

```
{ "id": "78526",  
  "label": "REFUTES",  
  "claim": "Lorelai Gilmore's father is named Robert." } -> KEEP,  
Robert single-token entity in BERT vocab. Despite there are two  
entities (i.e., "Lorelai Gilmore" and "Robert") there is exactly one  
entity with a single token in the BERT vocabulary (i.e., "Robert")
```

Enrich the datapoint as follow

```
{ "id": "78526",  
  "label": "REFUTES",  
  "claim": "Lorelai Gilmore's father is named Robert.",  
  "entity": {  
    "mention": "Robert",  
    "start_character": 34,  
    "end_character": 40  
  }  
}
```

The **start_character** is the index of the first character for the entity mention,
end_character the last. Example:

"Lorelai Gilmore's father is named Robert."[34:40] == "Robert"

[3] <https://s3-eu-west-1.amazonaws.com/fever.public/train.jsonl>

[4] https://s3-eu-west-1.amazonaws.com/fever.public/paper_dev.jsonl

[5] <https://github.com/flairNLP/flair>

[6] https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip

Task 1. Use LAMA to get predictions for masked entities

The task involves masking all specified single-token entities in **the official test set** and get BERT predictions for those, as well for the dev set (for which you need to report only the best accuracy).

To get prediction for a masked token you should use the LAMA repository (<https://github.com/facebookresearch/LAMA>). In particular, you should clone the repository locally and write a script to solve each task (that you should submit to complete the homework). Alternatively there are some instructions in the README on how to use lama in your code that you can follow.

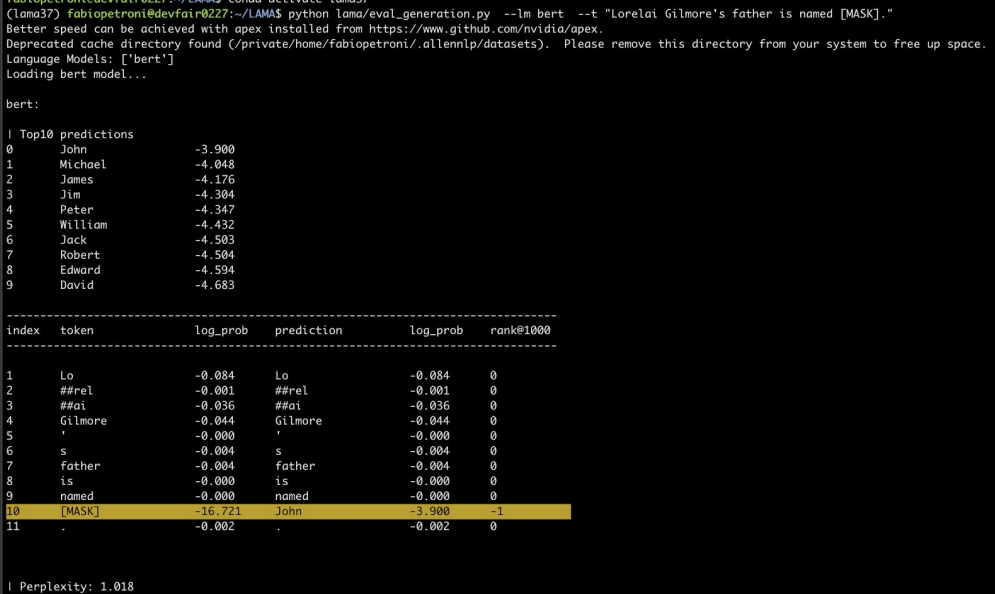
Note 1: In LAMA, modify the `download_models.sh` script to download only the **bert-base-cased model**. Concretely keep only this “if statement” in the file https://github.com/facebookresearch/LAMA/blob/master/download_models.sh#L105 remove all the rest.

Note 2: after creating the conda environment and installed the dependencies as in the README, you might need to run:

```
export PYTHONPATH=.
```

Example for a single claim:

```
python lama/eval_generation.py \
--lm "bert" \
--t "Lorelai Gilmore's father is named [MASK]."
```



```
(lama37) fabiopetroni@devfair0227:~/LAMA$ python lama/eval_generation.py --lm bert --t "Lorelai Gilmore's father is named [MASK]."
```

Better speed can be achieved with apex installed from <https://www.github.com/nvidia/apex>.
Deprecated cache directory found (/private/home/fabiofetroni/.allennlp/datasets). Please remove this directory from your system to free up space.
Language Models: ['bert']
Loading bert model...

bert:

Top10 predictions

0	John	-3.900
1	Michael	-4.048
2	James	-4.176
3	Jim	-4.304
4	Peter	-4.347
5	William	-4.432
6	Jack	-4.503
7	Robert	-4.504
8	Edward	-4.594
9	David	-4.683

index	token	log_prob	prediction	log_prob	rank@1000
1	Lo	-0.084	Lo	-0.084	0
2	##rel	-0.001	##rel	-0.001	0
3	##ai	-0.036	##ai	-0.036	0
4	Gilmore	-0.044	Gilmore	-0.044	0
5	'	-0.000	'	-0.000	0
6	s	-0.004	s	-0.004	0
7	father	-0.004	father	-0.004	0
8	is	-0.000	is	-0.000	0
9	named	-0.000	named	-0.000	0
10	[MASK]	-3.900	John	-3.900	-1
11	.	-0.002	.	-0.002	0

Perplexity: 1.018

Note 3: the `eval_generation.py` tool is just a way to graphically show BERT predictions. You should build an automatic script to get predictions (taking inspiration from `eval_generation.py`) and not solving the tasks manually!

Task1.1 Compute the accuracy on **the dev set** by using a single string matching heuristic, that is, the first prediction should be equal to the masked entities for a SUPPORT (i.e., `[MASK]==Robert`), different for a REJECT (i.e., `[MASK]!=Robert`). You could try to optimize the heuristic by applying some sort of string normalization (e.g., lowercase, stemming).

Task1.2 Compute the accuracy on **the dev set** using a similar heuristic but considering the top10 prediction, that is, associate a SUPPORT label if the correct entity is among the top10 predictions of BERT, REJECT otherwise. Again, You could try to optimize the heuristic by applying some sort of string normalization (e.g., lowercase, stemming).

Task1.3 Compute the accuracy on **the dev set** using a probability threshold for the correct entity. What is the probability that BERT associates with the correct token? Create a plot with threshold in the x-axis and accuracy in the y-axis.

Task 2. Train a classifier on MASK and gold representations

Use the FEVER train set to train a binary fact-checking classifier as follows.

You can use the dev set in [7] for hyperparameters finetuning.

For a datapoint, get BERT contextual representations for both the sentence with the entity and the sentence masking the entity.

Example:

Representation masking the entity `Lorelai Gilmore's father is named [MASK]`.

Representation with the entity `Lorelai Gilmore's father is named Robert`.

To obtain a vectorial representation, pick the last layer of the contextual embedding associated with the `[MASK]` token, in the first case, and with the entity token, in the second case.

The contextual representation can be obtained with LAMA, look at

`lama/get_contextual_embeddings.py`

As an example, modify the code, line 11 to:

```
sentences = [
    ["Lorelai Gilmore's father is named [MASK]."],
    ["Lorelai Gilmore's father is named Robert."],
]
```

Run:

```
python lama/get_contextual_embeddings.py --lm "bert"
```

```
(lama37) fabiopetroni@devfair0227:~/LAMA$ python lama/get_contextual_embeddings.py --lm bert
Better speed can be achieved with apex installed from https://www.github.com/nvidia/apex.
Deprecated cache directory found (/private/home/fabiopetroni/.allennlp/datasets). Please remove this directory from your system to free up space.
Language Models: ['bert']
Loading bert model...

bert:
Moving model to CUDA
Number of layers: 12
Layer 0 has shape: torch.Size([2, 13, 768])
Layer 1 has shape: torch.Size([2, 13, 768])
Layer 2 has shape: torch.Size([2, 13, 768])
Layer 3 has shape: torch.Size([2, 13, 768])
Layer 4 has shape: torch.Size([2, 13, 768])
Layer 5 has shape: torch.Size([2, 13, 768])
Layer 6 has shape: torch.Size([2, 13, 768])
Layer 7 has shape: torch.Size([2, 13, 768])
Layer 8 has shape: torch.Size([2, 13, 768])
Layer 9 has shape: torch.Size([2, 13, 768])
Layer 10 has shape: torch.Size([2, 13, 768])
Layer 11 has shape: torch.Size([2, 13, 768])
sentence_lengths: [13, 13]
tokenized_text_list: [['[CLS]', 'Lo', '##rel', '##ai', 'Gilmore', '"', 's', 'father', 'is', 'named', '[MASK]', '.', '[SEP]'], ['[CLS]', 'Lo', '##rel', '##ai', 'Gilmore', '"', 's', 'father', 'is', 'named', 'Robert', '.', '[SEP]']]
```

As representations you should use the vectors in the last layer (i.e., index 11), corresponding to the entity.

In the example, the entity is tokenized at index 11, see `tokenized_text_list`.

To access the representation you should use the `contextual_embeddings` variable at line 27. In particular

```
contextual_embeddings[layer_index=11][sentence_index][token_index]
```

This will give you a vector of dimension 768

So in this example,

Representation masking the entity = `contextual_embeddings[11][0][11]`

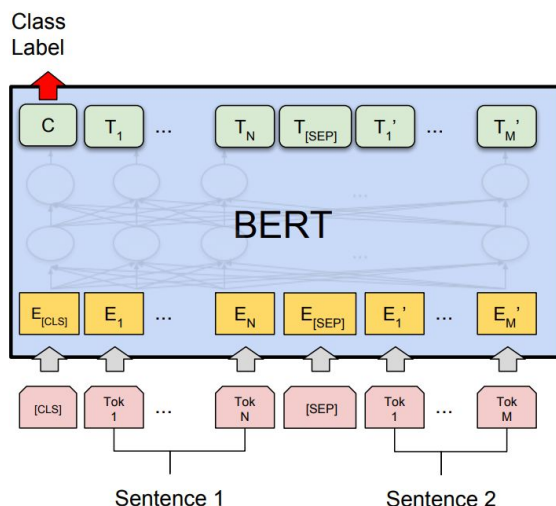
Representation with the entity = `contextual_embeddings[11][1][11]`

Use the concatenation of these two representations as input vectors and train a binary classifier, e.g., SVM, and get the accuracy performance on the dev set, as well as get predictions for **the official test set**. Remember: you can use the dev set to finetune the hyperparameters of the model (using e.g. GridSearch or RandomSearch).

Alternative solution for Task2

You can choose (as an alternative to task2) to run the classifier on top of the representation for the [CLS] token (the class label in the Figure below), that is always the first token in the sequence. In this case, you should use the complete sentence in input, without masking the entity. Alternatively you can try to finetune the BERT model for classification (without using an external classifier). Look here for an example

<https://mccormickml.com/2019/07/22/BERT-fine-tuning/>



Final note

Feel free to start pull requests to the LAMA repository during the execution of the homework if you see anything that can be improved, or if you want to add any functionality that could make life easier to other people (including your colleagues). I will try to monitor and review those.

Note 1: the code needs to be clean and well documented.

Note 2: this is not required to successfully accomplish the homework, but if you successfully land a pull request your name will be visible in the repo among the official contributors. :)

If you want more...

To successfully finish the homework only task 1 and 2 are needed.

Here you can find some ideas to continue this project, and potentially start a master thesis with Prof. Leonardi from it.

Avoid considering just single-tokens entities, but all entities.

Build functionalities into the LAMA repository to handle multi-tokens predictions.

Consider seq2seq transformers such as BART or T5, add those to LAMA.

Where/What To Send

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

1. A brief **PDF** report describing what you did and the choices you made (max 2 pages), including the best measured accuracy on the dev set, for all tasks (i.e., t1.1, t1.2, t1.3 and t2; 4 numbers to report)
2. The code of your scripts, 4 scripts (task1.1.py, task1.2.py, task1.3.py, task2.py). Comment the code to make the correction easier and more accurate.
3. A file with the prediction on **the official test set** (1082 examples), in jsonl format, corresponding to the best dev accuracy (i.e., not for all task, just for the strategy/task that gave you the best accuracy on the dev set)
4. **PLEASE, DO NOT PUT THE INPUT DATASETS AND NEITHER THE SOURCE LAMA REPOSITORY IN THE ZIP FILE.**

The name of the zip file must have this format:

DMT_2020__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_2020__HW_3__StudentID_StudentName_StudentSurname_StudentID_StudentName_StudentSurname.