Homework 1A Task 2

Federico Raschiatore 2071132

Abstract

This homework aims to address the critical aspect of input data preparation for leveraging a language generative model (LLM). The manner in which input data are formatted and presented significantly impacts the performance and effectiveness of an LLM. In particular, this study focuses on transforming input data originating from the SENTIPOLC dataset into a format suitable for comprehension by an LLM.

1 Introduction

NERMuD (Named-Entities Recognition on Multi-Domain) is a task presented at EVALITA 2023 consisting in the extraction and classification of named-entities in a document, such as persons, organizations, and locations.

2 Input shape

In input we have three datasets: "ADG", "FIC", and "WN", already split in train, dev and test.

The input dataset have the following shape:

```
L'            O
astronauta    O
Umberto       B-PER
Guidoni       I-PER
,             O
dell'         O
Agenzia       B-ORG
Spaziale      I-ORG
Europea       I-ORG
,             O
svela         O
ai            O
bambini       O
i             O
segreti       O
della         O
Stazione      B-LOC
Spaziale      I-LOC
.             O
```

As we can see each word is marked with a tag. On the left side we have the word that we are interested in, on the right side we have the relative tag splitted by a "TAB". The possible tags are "O", "B-XXX" and "I-XXX". The tags B-XXX and I-XXX specify the class of the relative words, the possible tags are PER ("Person"), ORG ("Organization") and LOC("Location"). We are interested only on the final part of this tags so we cut only the last 3 letters (e.g. "B-PER" -> "PER").

3 Reframe method

All the sentences are splitted one from each other using and empty array, so first of all I splitted the sentences and grouped them in an array.

I also create an array containing all the entities with a label different from 'O'. Each entity is associated with the relative label.

For each entity i randomly select a sentence that contain the selected word and i build the JSON with the following shape.

```
{
    "sentence_id": int,
    "text": str,
    "target_entity": str,
    "choices": List[str],
    "label": int,
}
```

Each field have the following meaning:

- id is an incremental number starting from 0
- text is the corpus of the selected sentence
- "choices" is the list of the possible classification that could be done for the sentence

  ["persona", "località","organizzazione"]

- "label" is a number that represent the index of the correct classification of the sentence referring to the precedent list

4 Prompts

I created 3 prompts and tested them for 100 samples using 2 different techniques. The main problem for the evaluation is the lack of models trained on an Italian vocabulary, so the first technique is based on the translation of the compiled prompt from italian to english and than i used a BERT model for the classification. Instead in the second techniques i used a BERT multilingual model trained on multiple languages. Both the applied methods are not perfect in fact didn't reach an high accuracy but they are good enough to understand which prompt works better.

- "Data la seguente frase '{text}' in questo contesto la parola '{target_entity}' si riferisce ad una (persona), ad una (località) o ad un' (organizzazione) ?"
- "Analizzando la frase '{text}' come classificheresti la parola '{target_entity}' date le seguenti possibili classi (persona), (località), o (organizzazione) ?"
- "La parola '{target_entity}' nel testo '{text}', è usata riferendosi ad una (persona), ad una (località) o ad un' (organizzazione) ? "

The results with the first method for each file are the following:

Prompt	Accuracy
`Prompt 1`	30%
`Prompt 2`	18%
`Prompt 3`	22%

Table 1: Accuracy for NERMuD_ADG_dev.jsonl

Prompt	Accuracy
`Prompt 1`	36%
`Prompt 2`	32%
`Prompt 3`	38%

Table 2: Accuracy for NERMuD_ADG_test.jsonl

Prompt	Accuracy
`Prompt 1`	28%
`Prompt 2`	16%
`Prompt 3`	22%

Table 3: Accuracy for NERMuD_ADG_train.jsonl

The results with the second method for each file are the following:

Prompt	Accuracy
`Prompt 1`	60%
`Prompt 2`	30%
`Prompt 3`	68%

Table 4: Accuracy for NERMuD_FIC_dev.jsonl

Prompt	Accuracy
`Prompt 1`	56%
`Prompt 2`	74%
`Prompt 3`	48%

Table 5: Accuracy for NERMuD_FIC_test.jsonl

5 Instruction to run the code

To run the code is sufficient execute all the cells. Be sure that a GPU device is available.

Prompt	Accuracy
Prompt 1	52%
Prompt 2	52%
Prompt 3	50%

Table 6: Accuracy for NERMuD_FIC_test.jsonl

Prompt	Accuracy
Prompt 1	18%
Prompt 2	14%
Prompt 3	26%

Table 7: Accuracy for NERMuD_WN_dev.jsonl

Prompt	Accuracy
Prompt 1	42%
Prompt 2	32%
Prompt 3	44%

Table 8: Accuracy for NERMuD_WN_test.jsonl

Prompt	Accuracy
Prompt 1	46%
Prompt 2	52%
Prompt 3	56%

Table 9: Accuracy for NERMuD_WN_train.jsonl

Prompt	Accuracy
Prompt 1	56%
Prompt 2	50%
Prompt 3	48%

Table 10: Accuracy for NERMuD_ADG_dev.jsonl

Prompt	Accuracy
Prompt 1	38%
Prompt 2	38%
Prompt 3	44%

Table 11: Accuracy for NERMuD_ADG_test.jsonl

Prompt	Accuracy
Prompt 1	26%
Prompt 2	26%
Prompt 3	30%

Table 12: Accuracy for NERMuD_ADG_train.jsonl

Prompt	Accuracy
Prompt 1	24%
Prompt 2	34%
Prompt 3	24%

Table 13: Accuracy for NERMuD_FIC_dev.jsonl

Prompt	Accuracy
Prompt 1	20%
Prompt 2	28%
Prompt 3	14%

Table 14: Accuracy for NERMuD_FIC_test.jsonl

Prompt	Accuracy
Prompt 1	14%
Prompt 2	10%
Prompt 3	22%

Table 15: Accuracy for NERMuD_FIC_test.jsonl

Prompt	Accuracy
Prompt 1	38%
Prompt 2	40%
Prompt 3	46%

Table 16: Accuracy for NERMuD_WN_dev.jsonl

Prompt	Accuracy
Prompt 1	32%
Prompt 2	38%
Prompt 3	22%

Table 17: Accuracy for NERMuD_WN_test.jsonl

Prompt	Accuracy
Prompt 1	28%
Prompt 2	30%
Prompt 3	32%

Table 18: Accuracy for NERMuD_WN_train.jsonl