Sentence Splitter
Multilingual Natural Language Processing
Homework 2

Ercoli Fabio Massimo
802397

Della Porta Nicolò
1920468

1 Introduction

Quoting [Redaelli and Sprugnoli, 2024] "Sentence splitting, that is the segmentation of the raw input text into sentences, is a fundamental step in text processing". According to [Frohmann et al., 2024] the main challenges are:

- Robustness to missing punctuation
- Effective adaptability to new domains
- High efficiency

According to [Redaelli and Sprugnoli, 2024] we can add to the list:

- Multilinguality

A sentence splitting approach that works well for English may not work well to split another language.
In this project we implemented two types of models for sentence splitting, using an Italian corpus as train and validation set. The first kind is based on embedding models, the second is based on generative LLMs. We aim to analyze and compare the two approaches. We also plan to test the models out of the original domain of training. The training model is a golden-annotated text from I Promessi Sposi (Quarantana) by Manzoni.

2 Methodology

2.1 Embedding-based

We fine-tuned a series of pretrained embedding models using the train and validation datasets provided by the homework guide.

The original dataset provides two large texts (one to use as train and the other as validation) along with the golden labels to mark the end of sentence (1) and all the rest of the words (0).

In order to make the datasets suitable for training, we had to apply some transformations. We needed to group words into sequences, and to tokenize words into tokens, aligning the golden labels consistently. With the constraint that the number of tokens of each sequence must fit the max length of the embedding models, for instance 512.

We generated a dataset using sequences of 192 words each. Notice that the number of tokens for each sequence will be strictly greater, since for each word the tokenizer of the model will produce one or more tokens.
Before using the datasets, we still need to align (as mentioned above) the labels to the tokens. The alignment strategy we applied consists of keeping 1 as the first token generated from a word having label 1, and using 0 for all the other cases.
One aspect that we had to address was the fact that the label distribution is very unbalanced for this use case. Most of the labels are 0s, and few are 1s. We applied 2 different strategies.
First, we set the **load_best_model_at_end** training argument to **true**, using **metric_for_best_model** set to F1. This implements early stopping based on F1, rather than on accuracy (which would be misleading for such an unbalanced dataset).
Second, we overrode the loss function to weight misclassification of class 1 (end of sentence) 30× more than class 0. This weighting factor was motivated by the label distribution (96.7% class 0 vs 3.3% class 1). We implemented this using a custom weighted trainer.

During inference, we don't need to align the labels, since the labels are generated by the model.

2.2 LLM-based

Fine-tuning an LLM is usually prohibitive or impossible using modest hardware, such as a single GPU with 16 GB / 24 GB RAM. So we applied two techniques to reduce the amount of memory required (and also the time) to fine-tune the models: LoRA (Low-Rank Adaptation) and parameter quantization.

A special focus was given to creating good prompts for both the training and inference phases. We first generated new datasets from the ones we used for the embedding-based solution, in which for every sequence we have:

- *input_text*: here we simply concatenate all words of the sequence into a single text
- *output_text*: here we group all the words that belong to the same sentence on different lines, numbered sequentially

In the process also we add the spaces between the words, paying attention to not introduce a space before a punctuation sign or after an apostrophe.

This is not a prompt yet. To create the final prompts, we created dictionaries of roles (system / user / assistant) with contents (we called them conversations) and passed them to the LLM tokenizer in order to have them converted into the proper chat templates.

We used the Supervised Fine-Tuning (TRL SFT) Trainer to train the model.

The fine-tuned models are (as usual) pushed to the Hugging Face repository, so that they can be used for inference from anywhere.

3 Experiments

3.1 Embedding-based: Training Evaluation

The performance goal we set is to maximize the F1 score on the validation set. Our expectation is that a good result on the validation set can generalize nicely on the test set.

For the embedding-based solution, we tested the following base models:

- BERT base cased
- ModernBERT-base-ita (Dec 2024)
- Italian BERT XXL (most established)
- XLM-RoBERTa base
- XLM-RoBERTa large
- Italian ELECTRA

For the LLM-based solution, we tested the following base models:

- Qwen3 4B
- Meta Llama 3.1 8B Instruct
- Mistral 7B Instruct v0.3 bnb 4bit
- Minerva 7B Instruct v1.0

As we said, both test and validation set come from the same novel: *I Promessi Sposi*.

Table 1: Best F1-epoch Embedding-based

Models	Best Epoch	F1	Valid loss
BERT base cased	6	0.9922	0.0014
ModernBERT base ITA	12	0.9938	0.0097
BERT base ITA XXL cased	11	0.9922	0.0021
XLM RoBERTa base	4	0.9953	0.0014
XLM RoBERTa large	12	0.9953	0.0087
Electra ITA XXL disc.	20	0.9953	0.0015

3.2 Out of Domain Evaluation: Eval Dataset

We designed experiments to test the same models on different authors. For this purpose, we planned to use the dataset from [Redaelli and Sprugnoli, 2024], in which different authors from the same century are used. The dataset has already the gold results annotated by the authors of the paper.

We evaluated the f1 score of the predictions, having the annotated novels as ground truth.

We produced labels from both the annotated novels and from predictions. For simplicity of the implementation, for this phase, we produce a label for each character, instead of a label for each token.

The golden-annotated novel rows were grouped in chunks of equal size of lines, to produce sequences.

We tested the in-domain novel (Quarantana) and 3 novels (Pinocchio, I Malavoglia and Cuore) from different authors of the same period.

3.3 Out of Domain Evaluation: Test Dataset

We'll try the test set on the two best embedding-bases models (according to the eval dataset results), Italian BERT XXL and Italian ELECTRA and one LLM-based model (Minerva 7B Instruct v1.0).

4 Results

4.1 Embedding-based: Training Evaluation

Table 1 reports the F1 scores and the losses on validation set for the best-F1 epoch, training the network with 30 epochs. This is of course an in-domain evaluation, since the validation set is from the same author (Manzoni) and even also the same novel (I Promessi Sposi). On training, the F1s on the validation set look very good.

Table 2: Out-Of-Domain Evaluation: Eval Dataset

Models	Quara.	Pinocc.	Malav.	Cuore
BERT base cased	0.9922	0.8058	0.7304	0.8757
ModernBERT ITA	0.9675	0.8495	0.8128	0.8864
BERT ITA XXL	0.9938	0.9947	0.9205	0.9591
XLM RoBERTa B	0.3953	0.5427	0.1980	0.2424
XLM RoBERTa L	0.5275	0.6618	0.3559	0.5
Electra ITA XXL	0.9923	0.9785	0.9171	0.9655

Table 3: Out-Of-Domain Evaluation: Test Dataset

Models	Type	F1
BERT ITA XXL	embedding-based	0.8770
Electra ITA XXL	embedding-based	0.8994
Minerva 7B Instruct v1.0 B	LLM-based	0.7445

4.2 Out of Domain Evaluation: Eval Dataset

When we evaluate the same models on different novels we got the result reported in table 2.

According to our tests, multilingual base models (XLM RoBERTa base and large) got very bad results, even in the same domain (Quarantana).

It seems to be better to use BERT base, even if it is not trained on Italian. This base model, like ModernBERT ITA, works very good in domain (Quarantana) and decently out of domain (other novels).

BERT ITA XXL and Electra ITA XXL got great results both in domain and out of domain.

The novel that seems to be the most difficult to split for all models is I Malavoglia. Best result is usually achieved in domain (as expected).

4.3 Out of Domain Evaluation: Test Dataset

A special focus was given to the evaluation of the performance on the test set. Not to make any design choice but to provide a final evaluation of the trained models (seeing those as immutable instances).

From the F1 score we computed, see table 3, embedding-based models (carefoully choosen) perform better than large LLM-based model.

Also LLM-based models have a series of further issues:

- they are one order of magnitude slower than embedding-based
- we cannot use CPU hardware even for inference
- they can hallucinate

4.4 What are the recurrent errors committed by the models?

As inference notebooks output we have the three cvs files:

- Lost_in_language_recognition-hw2_split-bert-base-italian-xxl-cased.csv
- Lost_in_language_recognition-hw2_split-electra-base-italian-xxl-cased.csv
- Lost_in_language_recognition-hw2_split-Minerva-7B-based.csv

Thanks to these files, we can analyze the miss-classification of the models.

Looking at the errors it seems that the models tend most of the time to split the sentence where it is not necessary.

For instance, "C'era una volta..." or "C'era una volta... -" has been misclassifed by all our trained models as a sentence.

A rare case in which the models do not recognize a new sentence, is:

"guardò dentro un armadio che stava sempre chiuso, e nessuno;"

All the models classified ';' as 0. Other cases are still about the presence of the sign ';'.

Looking at the log of the 'sentence_splitter_out_of_domain_test_generative.ipynb' we can see at some few hallucinations (not possible with embedding-bases models).

Some words have been transformed by the model, for instance 'messe' has been reported in the output as 'mise' or 'trasfigurito' in 'trasfigurato' or 'imprestezziti' in 'impresciuttiti'. We can say that the model may use a more-common-used word in the output (violating the fact that the output only should refer to the input words!)

One allucination is about portion of the sequence are not present (they were not reported in the output), but this fenoumenon seems to be quite rare. A word ('Sì! –') in the input has been completed by the model to produce the output.

5 Conclusions

The embedding base models are very efficient (they usually require a number of parameter at least 1 order magnitude less than the LLM-based ones). Also if a good base Italian model is chosen (for instance BERT ITA XXL) results are great both in domain and out of domain.

The LLM-based ones also seem to provide good results in terms of quality, but they can hallucinate, while it is simply not possible using an embedding-based approach.

Moreover, the latter are much more heavy to train and to use at inference time.

With the only exception of qwen3-4b-unsloth-bnb-4bit that takes 5 minutes to be trained the other take from 40 minutes to 1 hour to be trained, on a max memory GPU of 22.069 GB.

Furthermore, we had to use LoRA and the parameter quantization in order to train the models on a modest hardware.

All in all, the team has thinks that using an embedding based model would be better the natural choice to address the sentence splitting use case.

The AI seems to follow the common software engineer rule: there is no common receip for everthing: LLMs work grate for a lot of use cases, but not in all the case.

A Notebook Descriptions Appendix

A.1 Notebook List

We have produced 5 notebooks:
sentence_splitter_discriminative_training.ipynb: it trains several embedding-based models.
sentence_splitter_generative_training.ipynb: it trains some LLM-based models.
ss_out_of_domain_eval_discriminative.ipynb: it performs the out of domain evaluations for (already deployed) embedding-based models on evaluation dataset.
ss_out_of_domain_test_discriminative.ipynb: it performs the out of domain evaluations for (already deployed) embedding-based models on test dataset.
ss_out_of_domain_test_generative.ipynb: it performs the out of domain evaluations for (already deployed) generative models on test dataset.

A.2 Google Colab Execution

The notebooks have been developed to be run on a JupiterLab environment.

To run those on Google Colab, you should replace:

```
os.environ['HF_TOKEN']
```

with

```
userdata.get('HF_TOKEN')
```

and also you need to add the following line to the input:

```
from google.colab import userdata
```

Moreover, if you want to run the trainings part, you need to change the Hugging Face repository (in the notebook *fax4ever*), providing consistently your Hugging Face token.

While you don't need to set the Hugging Face token to run the inference notebooks.

B Models and Datasets Appendix

All the fine tuned models and the datasets were published to Hugging Face hub.

B.1 Embedding-based released models

- bert-base-cased-sentence-splitter
- ModernBERT-base-ita-sentence-splitter
- bert-base-italian-xxl-cased-sentence-splitter
- xlm-roberta-base-sentence-splitter
- xlm-roberta-large-sentence-splitter
- electra-base-italian-xxl-cased-discriminator-sentence-splitter

B.2 LLM-based released models

- qwen3-4b-unsloth-bnb-4bit-sentence-splitter
- meta-llama-3.1-8b-instruct-unsloth-bnb-4bit-sentence-splitter
- mistral-7b-instruct-v0.3-bnb-4bit-sentence-splitter
- Minerva-7B-instruct-v1.0-sentence-splitter

B.3 Datasets

The main Dataset to train the embedding-based models and LLM-based model are:

- manzoni-192
- llm-manzoni-192
- sentence-splitter-ood-192
- sentence-splitter-ood-128

Those are strictly derived from the CSVs files that were provided to do the Homework. No other data then that was used to train the models.

Table 4: Out-Of-Domain Evaluation: Embedding-based (TAKE-2!)

Models	Quara.	Pinocc.	Malav.	Cuore
BERT base cased	0.9859	0.8854	0.7826	0.9364
ModernBERT ITA	0.9130	0.5603	0.6667	0.8144
BERT ITA XXL	0.9939	0.8774	0.9121	0.9486
XLM RoBERTa B	0.3952	0.5426	0.2115	0.2424
XLM RoBERTa L	0.5275	0.6618	0.3559	0.5
Electra ITA XXL	0.9938	1.0	0.9348	0.9385

C Indeterminism Appendix

We tried to make the runs as deterministic as possible, but without affecting the performance. It means that some output derived from indeterministic algorithms may change with the executions. Making all deterministic would have affected too much the performance. So, it means that if you run everything from scratch, you could find slightly different results.

For instance the second time we re-trained everything from scratch we got different results for the out of domain evaluations, see Table 3.

Anyway, we can confirm the same analysis we made on the take-1. That's the hard life of the data scientists, even in the Deep Learning era!

References

Markus Frohmann, Igor Sterner, Ivan Vulić, Benjamin Minixhofer, and Markus Schedl. 2024. Segment any text: A universal approach for robust, efficient and adaptable sentence segmentation. *Preprint*, arXiv:2406.16678.

Arianna Redaelli and Rachele Sprugnoli. 2024. Is sentence splitting a solved task? experiments to the intersection between NLP and Italian linguistics. In *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*, pages 813–820, Pisa, Italy. CEUR Workshop Proceedings.