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

Because a sentence splitting 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 an embedding model, the second is based on a generative LLM. We aim to analyze and compare the two approaches. We also plan to test the models out of the original domain of training, that is an annotated text from I Promessi Sposi (Quarantana) by Manzoni.

2 Methodology

2.1 Embedding-based

We fine-tuned a pretrained embedding model 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) together 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 with 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 Embedding-based: Out of Domain Evaluation

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 true.

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

The golden-annotated novel rows were grouped in chunk 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 LLM-based

Results for LLM-based (generative) models will be evaluated manually, simply taking an out of domain text fragment from Cuore novel and pass it as input to the target test models.

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: Embedding-based

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

4.2 Embedding-based: Out of Domain Evaluation

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 does 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 more difficult to split for all model is I Malavoglia. Best result is usually achived in domain (as expected).

4.3 LLM-based

All the 4 LLM-based tested models provide the same output to the text we passed to the prompt. The output text is reported on Appendix A and looks totally correct.

Unfortunatelly, even the inference requires a GPU and also you can usually use only one model on the same notebook run also for inference.

The models can allucinate. For instance, we experienced in some runs that sometimes characters, in particular white spaces had been fabricate by the model: they were not present in the input text to split passed by the user.

4.4 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 of the results but they can allucinate. in the input prompt, but they were present in the output: this is simply not possible using an embedding-based approach. Also they are much more heavy to train and to use at inference time. For instance if you cannot use different LLMs models in the same notebook, for memory reason using a commercial GPU. Furthermore, we had to use LoRA and the paramater quantization in order to train the models on a modest hardware.

A LLM output Appendix

The sentence splitting output returned by the LLM-based model is:
1. Non era un legno di lusso, ma un semplice pezzo da catasta, di quelli che d'inverno si mettono nelle stufe e nei caminetti per accendere il fuoco e per riscaldare le stanze.
2. Non so come andasse, ma il fatto gli è che un bel giorno questo pezzo di legno capitò nella bottega di un vecchio falegname, il quale aveva nome mastr'Antonio, se non che tutti lo chiamavano maestro Ciliegia, per via della punta del suo naso, che era sempre lustra e paonazza, come una ciliegia matura.
3. Appena maestro Ciliegia ebbe visto quel pezzo di legno, si rallegrò tutto; e dandosi una fregatina di mani per la contentezza, borbottò a mezza voce: "Questo legno è capitato a tempo; voglio servirmene per fare una gamba di tavolino."

The Minerva-base model also returned the token <|*eot_id*|> at the very end.

B Notebook Descriptions Appendix

We have produced 4 notebooks:

- **sentence_splitter_embeddings**: To train an embedding-base model.
- Meta Llama 3.1 8B Instruct
- Mistral 7B Instruct v0.3 bnb 4bit
- Minerva 7B Instruct v1.0

Notice that the notebooks have been developed to be run on a JupiterLab environment: if you are trying to run 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.

C Models and Datasets Appendix

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

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.