NLP2023-HW2 Report

Andrea Sanchietti
Matricula / 1883210
sanchietti.1883210@studenti.uniroma1.it

1 Introduction

My work mainly focused on using a BiEncoder approach to encode both glosses and target words in order to predict the definition that fits best for the given context. I decided to train this model fully on the fine grained dataset and then adapt the model to the coarse grained task.
I also designed my own model based on a Siamese Network and Contrastive Loss. The first part of this document will show the experiments with BEM while the second part will explain my Contrastive model and will compare it to the BEM one.

2 Related works

The model that inspired me the most was BEM (Blevins and Zettlemoyer, 2020) from Allen et al. because during the lecture on WSD, I had a similar intuition which was based on the idea of learning to map samples of a dataset to a manifold that is able to cluster similar inputs in a lower dimension. For the Contrastive model, I was inspired by some works in computer vision and, in particular, in person re-identification, where state of the art models uses a Siamese Network to learn which are the important features of a probe in order to lower the L2 distance of probes that come from the same identity (person).

3 BEM Model

My BEM model follows the paper cited before, with two encoders that have been trained together to produce vectors with an high cosine similarity between target words and the associated correct definition. A view on the architecture is shown in figure 1. In particular, the encoder takes as input the definition and masks the output so that only the embeddings that correspond to the target word are selected. If the tokenizer splits the target token into n different sub-tokens, I average the n resulting embedding to get a single vector that represents the semantics of the original token. For the glosses, I added the CLS token at the beginning of the sentence and took only his corresponding output embedding as representative of the whole gloss.
Once the target word and the glosses are embedded, the model rearranges the latter inside a matrix and uses a cosine similarity to have a score for each gloss. Then, the model returns the scores, computes the cross entropy loss and returns proceeds with the backward pass.

4 BEM Pretrained encoders

For the encoders, I tested both **BERT** and **RoBERTa**. I used pretrained models from Huggingface and freezed all the blocks except the last two. This took me about 99% of vram during training and 100% of gpu power, so I had to reduce the batch size to only 8 samples.

5 BEM Model Fine-tuning

In order to fine-tune the model, I set **Weights and Biases**'s sweeps with **PyTorch Lightning** and Bayesian optimization of the parameters. Each epoch took around 50 minutes, so I set the epochs to 10. I tested different learning rates from 1e-7 to 1e-4, with penalty between 0 and 1e-8 and batch size 8. I also used a **max_len** parameter set to 400 that was used to add padding to the sentences. Adding, I tested gradient clipping $\in \{1.0, 2.0\}$ while the choice of the embedding models were $\{BERT, RoBERTa\}$. Figure 2 shows the sweep of the runs. As we can see, the models that performed better were the ones with **RoBERTa** as encoder for both the sentence and the glosses, a learning rate between $1e-4$ and $1e-5$, a low weight decay $(1e-7, 1e-8)$ and gradient clipping set to 2.0. figures 3 shows the training and validation loss for the best three runs while 4 shows their respective accuracy.

sentence model	roberta-base
gloss model	roberta-base
learning rate	1e-4
gradient clipping	2.0
weight decay	1e-8
training f1	0.46
validation f1	0.49
training accuracy	0.85
validation accuracy	0.72
training precision	0.60
validation precision	0.62
training recall	0.50
validation recall	0.56

Table 1: Parameters and results for the best model. I used the weighted prediction/recall/f1 as metrics.

6 BEM Evaluation

The table 1 shows the training statistics of the best run for the **fine-grained**task:

As we can see, the scores are a little bit lower than the state of the art, however, this result may be given by the different training sets used.

After this first evaluation, I wrote a predict function that is able to map my fine-grained result to the coarse-grained test and used the *test.sh* script to run a test. The result on this task was **88.9%**, which is an acceptable result even thought it could have been better.

7 Contrastive Model

The contrastive model is based on a Siamese Network that tries to extract embeddings from tokens/sentences that are spatially close to each other. Since the training of this model took a lot of time (80 minutes per epoch), I could not property fine-tune the model. However, I was able to find a good parameter setting from the beginning and run only a few experiments.
A Siamese Network is based on two models: the first one encodes the sentence, while the second encodes a gloss. These two models share parameters, so the network can be implemented as a single embedding model that takes both as input (one at a time) and returns the embeddings. The formula of the contrastive loss is:

$$L = (Y)(-log(Y_{pred}))+(1-Y)(-log(1-Y_{pred})) \quad (1)$$

embedding model	roberta-base
learning rate	1e-5
batch size	256
gradient clipping	2.0
weight decay	1e-8
max unfreeze	4
unfreeze step	2
lr step size	2
lr gamma	0.8
training f1	0.87
validation f1	0.85
training accuracy	0.87
validation accuracy	0.86
training precision	0.88
validation precision	0.85
training recall	0.86
validation recall	0.85

Table 2: Parameters and results for the best model. Results are based on how many times the distance between a sentence and a gloss was less than a fixed threshold (0.5).

and what he does is to penalize the network when the prediction is wrong, while rewarding the correct predictions. The effect on the embedding space is that similar sentences will be close while sentences with a different meaning will be separated apart 5.

8 Contrastive Model Hyperparameters and Results

During my experiments I used **RoBERTa** as embedding model and added a gradual layer unfreeze of the embedding model, together with a learning rate decay. The best hyperparameters that I found, together with the best results are shown in table 2:

where max unfreeze, unfreeze step, lr step size and lr gamma are respectively the maximum number of blocks of the embedder to unfreeze, the unfreeze step, the learning rate decay step size and the learning rate decay. The plots of the metrics are shown in images 6, 7, 8, 9.
Note that the metrics were calculated based on a threshold set to 0.5. This means that, if the distance between the embedded target token and the gloss was lower than the threshold I considered the output of the model as 1. Otherwise, I considered the output as 0. Usually, threshold can be fine-tuned after the training in order to balance the false ac-

model	test score
BEM	88.9%
Contrastive	88.5%

Table 3: Parameters and results for the best model.

ceptance rate and the true acceptance rate, but in our case it is not important as the only score that we care of is the the CMC score at rank 1 for each gloss set.

9 Model Comparison

In this section, I will compare the BEM model and my Contrastive model on the test coarse-grained task.

In order to have the coarse prediction from the BEM model, I gave all the input glosses to the module and then I returned the hypernym that corresponded to the gloss with the highest cosine similarity.

For the Contrastive Model, I did a similar thing: I gave to the model all the glosses and returned the hypernym of the gloss with the smallest L2 distance from the input sentence.

Results are shown in table 3

As we can see, my model performed slightly worse than BEM even without a proper hyperparameter tuning. This shows how my Siamese Network model is a valid option and could yield even better results. In particular, we can see how the size of my model is half the size of the BEM model and is able to yield a similar result.

10 Other experiments and possible enhancements

Before training the embedding layers as i explained in the BEM chapters 4, I also tried to train the models with only the last layer freezed. However, this method was not able to perform as well as the one that I did because the model didn't have enough unfreezed parameters.

Another thing that I tested was to train the model in a GAN-fashion way, where I freeze one embedding model and train the other, switching them at the end of each epoch. Even this method didn't yield good results as the loss started to diverge pretty soon. However, I think that a deeper hyperparameter fine-tune of this second technique could overcome the results of my best BEM model. On the other hand, this method may require more time to find an optimal configuration and may also need more epochs to converge and since each epoch took me a lot of time, I could not explore this solution (my sweep took me about 60 hours and a lot of current).

I also wished I had time to fine-tune my contrastive model better to see whether it is able to enhance its performances.

I did not exploit the information available from the already disambiguated glosses, however, I thought of an approach that could be used in this situation: once we have the BEM/contrastive model working, we can see which is the token with the lower number of candidates and use the predicted element from that set to generate a new sentence with BART that can then be compared with the next set of candidates. It is possible to train BART in this setting since we can simply use a cross entropy loss that says to BART whether the BEM model correctly predicted the hypernym or not. As a matter of fact, if the BART model generates a good new sentence, the BEM/Contrastive model will have more correct predictions while if the generated sentence is bad, the model will fail and BART will know that the produced text was bad.

Finally, it could be interesting to use a fine-grained dataset to train my model and see whether it is able to compete for the fine-grained WSD task.

11 Conclusions

This document showed my work for this homework:

- Fine-tuning of a model based on BEM
- A novel model based on a Siamese Network and Contrastive loss.
- A comparison of these two models on the coarse-grained task
- Some ideas about a possible enhancement of the proposed approach.

Since my score for the Contrastive Model is very similar to the one achieved BEM, I decided to compete for the homework with my model.

12 Images

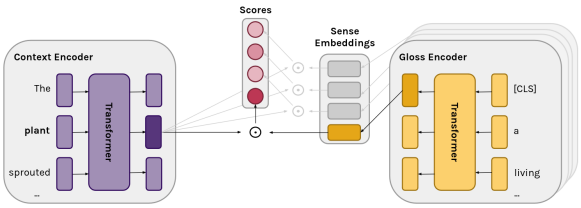

Figure 1: Architecture of the BiEncoder Model (BEM) from (Blevins and Zettlemoyer, 2020).

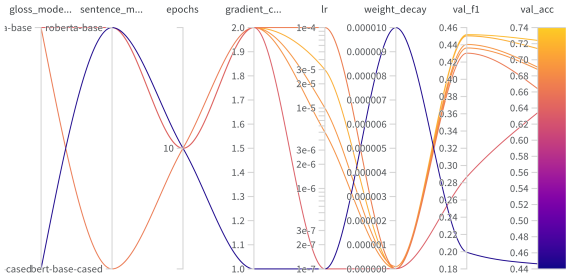

Figure 2: Sweep from wandb.

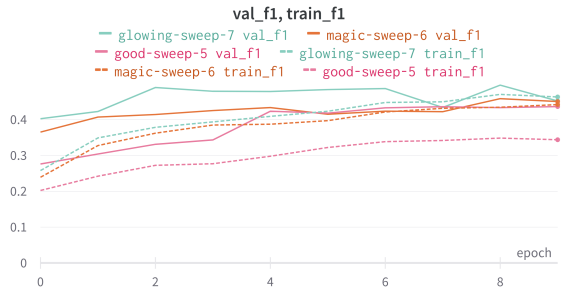

Figure 3: Training and validation loss for the best three runs.

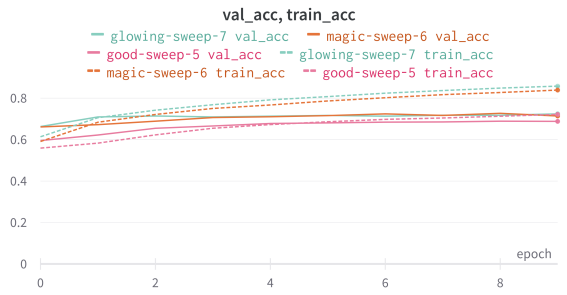

Figure 4: Training and validation accuracy for the best three runs.

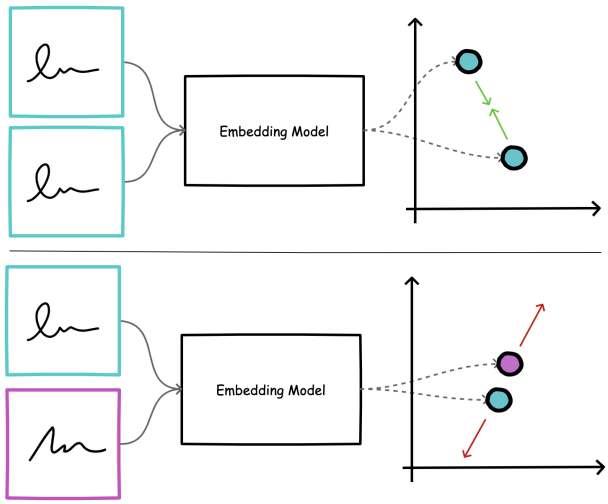

Figure 5: Visualization of the architecture of a Siamese Network and the effect of the contrastive loss on the embedding space.

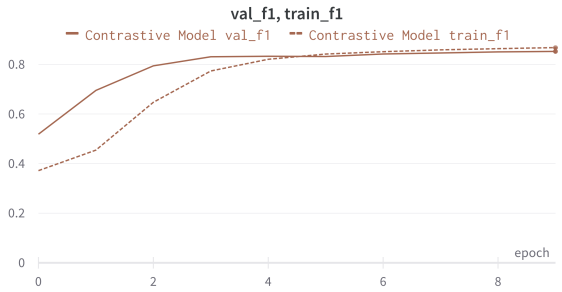

Figure 6: F1 comparison for the Contrastive Model.

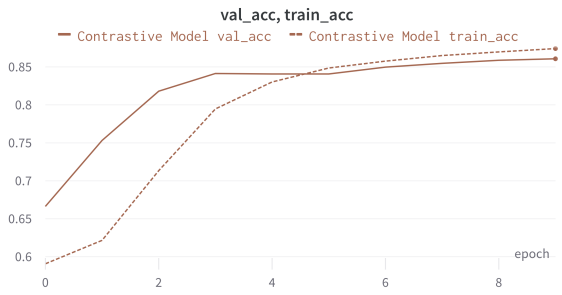

Figure 7: Accuracy comparison for the Contrastive Model.

Figure 8: Training loss for the Contrastive Model.

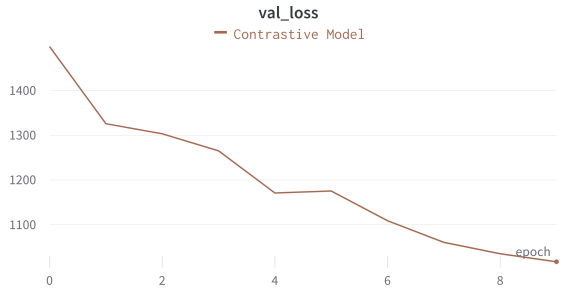

Figure 9: Validation loss for the Contrastive Model.

References

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online. Association for Computational Linguistics.