NLP Homework 2: Word Sense Disambiguation

Leonardo Lavalle
1838492
Sapienza, University of Rome
lavalle.1838492@studenti.uniroma1.it

1 Introduction

Word Sense Disambiguation (WSD), is the task of associating a word in context with its correct meaning from a finite set of possible choices [6]. The focus of the homework is on *coarse-grained* WSD, where the similar senses of a polysemous word (*fine-grained*) are clustered to obtain a smaller number of sense distinctions. I treated the problem as a single-label classification problem, assuming that a target word can only have one sense in a given context. The challenges to face are multiple, in particular the one regarding *Zipf* law (Figure 1), which makes the task not trivial at all. There's indeed a vastity of polysemous words that appears only once (Figure 2). Recent works employ Knowledge Graphs integration ([1], [3]), sense glosses ([5], [2]) or treat the problem as a multi-label classification [3] arguing that the single-label case does not reflect how human beings disambiguate text and I firmly agree on that. Anyway, for the sake of simplicity and because the main goal is coarse-grained WSD, I found the approach of selecting the most suitable sense from a predefined subset of a sense inventory highly reasonable.

2 Methods

I've employed as starting architecture the one used in [3] and [7] by Sapienza NLP group. It's simply formed by a *Transformer Encoder* and a *Classifier*. Treating WSD as a multiclass token classification problem, the model is trained to maximize the probability of the single most appropriate sense by minimizing a cross-entropy loss.

Preprocessing. Look at the data! Without a quality dataset there's no way of achieving good results, especially for WSD. In the present case, the given dataset is quite clean and does not required much work. In fact, w.r.t. the previous homework, the *cleaning* tokens operations are very basic and not so aggressive. I've also filter out the sentences which deviates from the data majority. As we can see in Figure 3 on the right part, by filtering using lower and upper length bounds, the histogram shape looks good and data statistics remained basically unchanged. Thanks to Transformers and WordPiece tokenization algorithm, all the preprocessing part of building the vocabulary and all the concerns about generating as few UNK tokens as possible does not exist anymore.

Sense Inventory. Since we are dealing with sense-annotated data in a supervised fashion, the first thing to do is to create a *sense inventory*, that is the set of all possible senses taken from WordNet. To do so, I considered all senses stored in files given for the homework. The inventory comprehend all the coarse-grained senses (2086 senses from *train*, 750 from *val* and 781 from *test*) with a size of 2158 and all the fine-grained ones with a size of 4476. Of course it doesn't cover in general all the possible senses of all the possible polysemous words (another big problem of WSD). Therefore I implemented a strategy to deal with unknown senses.

Transformer Encoder. How powerful are Transformers architectures is not a secret anymore. I used *Bert*, *Roberta* and *Deberta* as FastTokenizer and Encoders. Their usability and ease of access thanks to HuggingFace library makes them nowadays the first choice for tokens embedding. In practice, to embed the target word to disambiguate, I first summed the last four hidden layers representation of the encoder (common practice) and then sum or average (it was an hparam) all the output tokens relative to the target word (because WordPiece might have generate multiple tokens). The result is a 768 (*base* architecture) or 1024 (*large* one) size vector which is then fed into the classifier network.

Classifier. Starting from the exact architecture

used in [3], the input goes first through a BatchNorm layer, then a non-linear layer with Swish activation function [4] and finally a linear layer. During my experiments I changed Swish with ReLu and set 512 as hidden dimension for the non-linear projection.

Glosses. As suggested, I implemented an approach which employs *glosses*. I developed the idea of GlossBert [5] by framing it as a token-level binary classification task and using the classifier described above. Being glosses only about fine-grained senses, I had to first train a fine-grained system and then trace back coarse-grained senses at prediction time. I attempted an alternative approach, yielding unsatisfactory outcomes, where I directly trained a coarse-grained model using concatenated input glosses derived from the fine-grained senses related to the coarse one.

3 Results

Experimental Setup. I trained all my models from scratch with PyTorch Lightning using *Adam* as optimizer and *ReduceLROnPlateau* as learning rate scheduler. Techniques as weight decay, dropout and early stopping have been exploited to avoid overfitting. I trained all the *base* models with FP16 mixed precision training due to the limited computational resources of my local machine. For this reason I had to carefully consider the optimal number of Bert layers to unfreeze and/or the appropriate batch size to use. *Large* models were instead trained on Google Colab.

In Table 1 we can see how the *baseline* reaches a relative high accuracy, despite the fact that all Bert layers were freezed and no dropout were used. Beginning from a better version (BERT base), I tried to increasingly improve it. I leveraged *lemmas* of the input words as inputs themselves or exploited words *POS tags* by giving them as additional input following a similar approach to [8]. Setting lr to 4e-5 (after an exhausting search), using the novel POS tags strategy, apllying ReLu instead of Swish activation function and *base* Bert as encoder, led to my best performing model (**0.9484**) for coarse-grained WSD. Batch size was set to 64. The same exact model for fine-grained WSD reaches 0.8257 (Table 2). This outcome is rather obvious if we think about the different difficulty level of the two tasks. To understand if these two models are really learning and they are not simply predicting the most frequent senses, I conducted some tests. First I tested on both WSD tasks a system which always predicts the most frequent sense (i.e. the first one) which achieves 0.5014 on fine and 0.6796 on coarse WSD (not so low results). Then I analysed how many times my two models predict the first sense and it comes out that they do that only 15.38% (fine) and 19.55% (coarse) of the time. I can infer that my models are learning.

Coarse vs Fine. The first that came to my mind to enhance accuracy was to use a fine-grained model to predict coarse senses. But none of my fine models beated my WSD *base* model (Figure 4 for plots). I've also attempted to guide fine models to achieve greater results in coarse task by modifying the loss and giving a reward each time they predicted a fine sense in the correct homonym set (it slightly improves performance). To strengthen fine models I additionally tried to apply a coarse model (Table 2) with the aim of pre-filtering out the possible fine senses to choose, but with negative outcomes.

I employed *large* Bert encoders, hoping in great improvements, but actually the most evident difference (+0.0222) is within Gloss models (Figure 5). In fine systems there's instead an enhancement which is then not reflected in coarse senses prediction (both 0.9407, Table 3). I must admit that I felt disappointed upon witnessing the less than satisfactory effectiveness of Gloss models, but I'm quite sure that with higher GPU RAM we can easily overcome WSD *base* model. Anyway, all the models for coarse-grained task are comparable (Table 3) and there aren't big differences.

Difficult senses. During dataset preprocessing I noticed that in the coarse-grained case there were many words with only one possible sense and as we can see in Figure 6 most of polysemous words have the range of possible meanings between 1 and 5. This of course, helps my models performance. Therefore, I tested my coarse systems on the most difficult words, that are the ones which have at least 5 possible meanings. In this way we can really appreciate models capability of predicting the right coarse sense and truly understand which are the best or not. In Table 4 it's clear that WSD *base* far exceeds all the other fine-grained models which only at prediction time select the coarse sense. This means that there exists an inherent difficulty of choosing a single fine-grained sense to a polysemous word and it indicates that the way to follow for future research is the one of the multi-label framing already depicted by [3].

Images and Tables

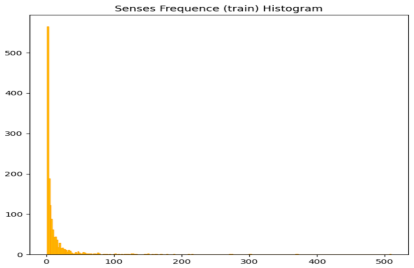

Figure 1: Histogram of synsets frequence. The distribution is exactly following *Zipf law*. Few senses occur very frequently while the majority of senses occurs infrequently.

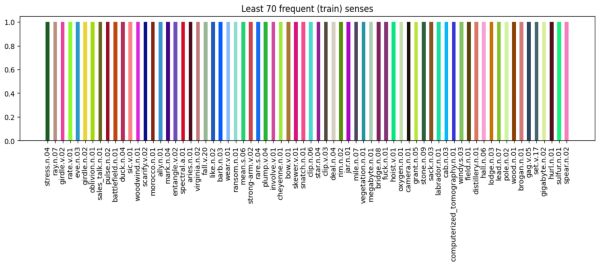

Figure 2: Plot of the 70 least frequent (they appear only once) coarse-grained senses in the training set.

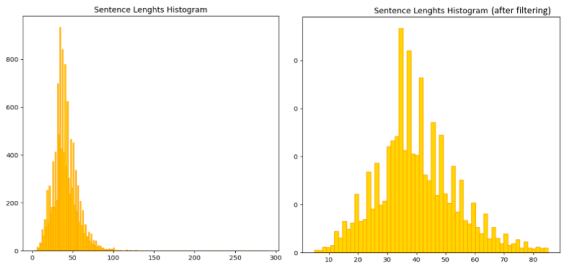

Figure 3: Histogram of training sentences length. On the left the one before the sentence filtering and on the right the one after. The "sentence outliers" are the points which contribute to the long and thin tail of the left histogram. In fact, the mean value is 40, but we have the maximum sentence length that reaches 289. After the filtering, only 141 sentences (1.14%) are excluded.

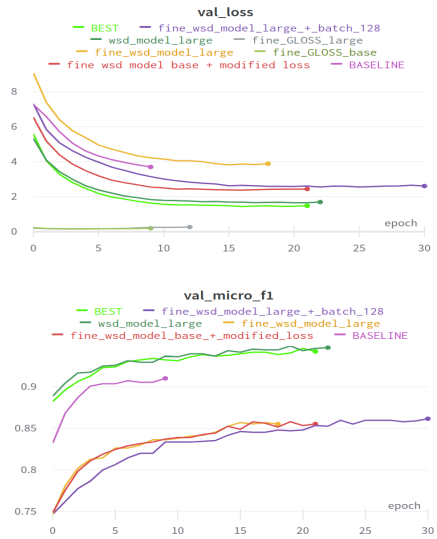

Figure 4: Validation losses (on top) and validation micro F1-scores (on bottom) WandB plots of the main trained models. The BEST model depicted in green is the WSD *base* we talk about in the report because is the best performing one. Notice how low are the losses of the two Gloss models (binary cross-entropy reaches lower values).

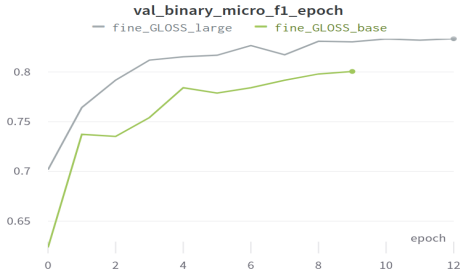

Figure 5: Plots of val binary micro F1-score of my Gloss models. It's evident how better is the *large* model with respect to the *base* one. The promising aspect of this type of systems lies in the tendency for performance to rapidly improve with increased GPU RAM.

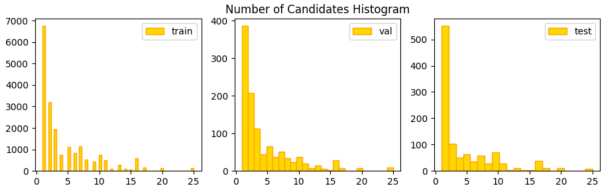

Figure 6: Number of candidates histograms of the polysemous words across *train*, *val* and *test* set. The trend is nearly the same among the 3 curves and it can be observed that most of the words to disambiguate have the range of possible meanings between 1 and 5.

Model	Accuracy
Baseline	0.9050
BERT base	0.9362
+ lemmas	0.9357 *(-0.0005)*
+ pos tags	0.9439 *(+0.0077)*
+ ReLu	0.9457 *(+0.0018)*
+ 4e-5 lr	
WSD base *(bert)*	**0.9484** *(+0.0027)*
WSD base *(roberta)*	0.9376
WSD base *(deberta)*	0.9376

Table 1: Improvements of accuracy starting from the *baseline* and arriving to the WSD *base* model that is actually the best model in coarse-grained WSD. Bert encoder achieves the highest accuracy with respect to Roberta end Deberta ones.

Model	Accuracy
fine WSD base	0.8257
+ modified loss	0.8230 *(-0.0037)*
+ coarse filter	0.8044 *(-0.0223)*
fine WSD large	0.8248
fine WSD large (128 batch)	**0.8341**
fine Gloss WSD base	0.7989
fine Gloss WSD large	0.8211

Table 2: Results of accuracy of my trained models on fine-grained WSD. The first model is exactly the same as WSD *base* but trained on a fine-grained dataset. "Base" stands for *bert-base-uncased* pretrained encoder and "large" for *bert-large-uncased*.

Model	Accuracy
WSD base	**0.9484**
WSD large	0.9457
fine WSD base	0.9407
fine WSD base + modified loss	0.9416
fine WSD large (128 batch)	0.9407
fine Gloss WSD base	0.9351
fine Gloss WSD large	0.9407

Table 3: Results of accuracy of my trained models on coarse-grained WSD. In the first part are grouped the models trained on a coarse-grained dataset. The second group of models represents instead architectures trained on fine-grained senses, but leveraged at inference time to predict coarse-grained senses.

Model	Accuracy
WSD base	**0.7000**
WSD large	0.6670
fine WSD base	0.5574
fine WSD base + modified loss	0.5410
fine WSD large (128 batch)	0.5738
fine Gloss WSD base	0.4918
fine Gloss WSD large	0.5738

Table 4: Results of accuracy of the same models of Table 3 but tested on the most difficult words to disambiguate (60 in *test* set). These words are the ones which have at least 5 coarse meanings.

References

[1] Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online. Association for Computational Linguistics.

[2] 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.

[3] Simone Conia and Roberto Navigli. 2021. Framing word sense disambiguation as a multi-label problem for model-agnostic knowledge integration. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3269–3275, Online. Association for Computational Linguistics.

[4] Steffen Eger, Paul Youssef, and Iryna Gurevych. 2018. Is it time to swish? comparing deep learning activation functions across NLP tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4415–4424, Brussels, Belgium. Association for Computational Linguistics.

[5] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.

[6] Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2).

[7] Riccardo Orlando, Simone Conia, Fabrizio Brignone, Francesco Cecconi, and Roberto Navigli. 2021. AMuSE-WSD: An all-in-one multilingual system for

easy Word Sense Disambiguation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 298–307, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[8] Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.