WSD on hierarchical-organized semantics

Emanuele Rucci, 2053183

Abstract

Word Sense Disambiguation (WSD) poses a significant challenge within the field of Natural Language Processing (NLP), requiring the accurate determination of a word's intended meaning in a given context. This document aims to explore diverse strategies documented in the literature for solving this issue. These strategies are then applied to the provided dataset, evaluating their effectiveness.

Additionally, the document analyzes the connection between identifying coarse-grained senses and enhancing the precise classification of fine-grained senses, and vice versa. By investigating this relationship, the document aims to uncover the synergistic effects of these seemingly disparate processes.

1 Introduction

WSD aims to solve the ambiguity of word meaning in context; when a word has multiple meaning (polysemy) is fundamental to disambiguate the given target word in order to fully understand the complete meaning of a sentence. In order to accomplish this task two big group of technique have been applied in literature during time: fully neural approaches ([1][2] and many others) and knowledge-based approaches [3][4]. Here in this document, we will explore neural approaches applied on the given dataset. Moreover, a couple of experiment have been devoted to study how might be possible to combine in a single model the power to extrapolate in hierarchical way deeper semantic in form of word senses.

2 Methodology

As said, have been studied models in the neural approach family for WSD. All the models generated consists of an encoder and a classification part [2]. The encoder extracts the embeddings of the sentence and the features are used to predict the senses using the classifier.

2.1 Encoder

BERT model as the encoder [5]: it is responsible of tokenization and generate the contextualized embedding for the sentence (the hidden states for each token with dimension *H*).

For each word to disambiguate the network takes the mean of the last 4 layer of BERT [6].

With these settings the task is being translated in a token-to-token tagging task: assign to each word to be disambiguated the sense predicted. However, an extra step is required to avoid the token fragmentation of a word and retrieve the complete embedding information: the BERT tokenizer might split a single word in 2 or more tokens; when this happens, the model identifies the case and then, considering a list of tokens mapped to the same word, their hidden states are averaged:

$$f = \frac{1}{k}\sum_{l=1}^{k-1} h_{j+l}$$

In the equation h_i is the hidden state of token in the list of the *k* tokens mapped to the same word.

Image 1, (a) shows a schema about the encoding process.

Moreover, have been tested two implementation strategy for the forward of the linear layer:

1. *f* contains the embeddings for all the tokens of the sentence;
2. *f* contains only the embeddings of the word to be disambiguated: from the *f* tensor are removed all the h_i correspondent to token *i* which are not mapped to a word to be disambiguated.

2.2 Classifier

All the classifiers are fully connected linear-layers and have been used 3 layers in the different settings of the experiments:

$L_1(x) = W_1\, x + b_i$ **and** $W_1 \in \mathbb{R}^{|CG|\ x\ H}$

$\mathrm{L}_2(x) = W_2\, x + b_i$ **and** $W_2 \in \mathbb{R}^{|FG|\; x\; H}$

$\mathrm{L}_3(x) = W_3\, x + b_i$ **and** $W_3 \in \mathbb{R}^{|FG|\; x\; |CG|}$

CG and FG represent the sets of coarse and fine grained senses in the dataset.

Before to apply any of this linear layer, a mask is applied to the logits (the output of the encoder) in order to ignore all the logits value for the tokens mapped to word that doesn't have to be disambiguated; differently from [2], this allows to have only one linear layer for all the polyseme instead of many ones, each for a polyseme.

Image 1, (b) shows this mask process.

2.3 One model for coarse and fine

An aspect of the given task that has been assumed, is the capacity of a model to benefit from the hierarchical organization of the senses.

To this purpose have been tested two model's architecture:

- Fine grained model to deduct coarse grained (*DeductCg – Model 8*)
- Joint learning of coarse and fine grained and re-deduction of the fine grained (*Fine-Coarse – Model 9*)

2.3.1 Deduct Cg Architecture

The first architecture that allows to deduct automatically the coarse-grained sense, consist in the training of the previous discussed architecture using the fine-grained classifier. Given the prediction of the model for a given word to be disambiguated the mapping file from the dataset is used to get the "super-sense" label: the coarse grained sense.

2.3.2 Fine-Coarse Architecture

Another study has been conducted based on the assumption that both coarse and fine classifications could mutually enhance each other. Unlike the previously discussed approach of ascending the semantic hierarchy, this study takes a different stance. Rather than attempting to move upward, the model is designed to systematically learn the importance of both tasks concurrently over time.

In the Experiments section will be discussed in the detail how this idea has been realized.

3 Experiments

3.1 Dataset

Table 1 summarizes dataset statistics: It comprises 6 files with sentences in natural language, part-of-speech info, and lemmas. The sense vocabulary uses WordNet and is composed of 2158 CG senses and 4476 FG senses. Additionally, there's a mapping from fine-grained to coarse-grained senses. Training used accuracy, equivalent to micro-F1 for single-sense word classification.

3.2 Experiments List and Setup

Table 2 reports all the experiment that have been conducted in this study as well as their identifier. The metric that has been used is the F1-score, all the models have been trained for 100 epochs (some experiments for less epoch due to Colab session duration limit but all of them were not improving anymore). The optimizer is Adam. The learning rate has been reduced with a StepLR scheduler with step size = 1 and gamma = 0.1 starting from 1e-3.

3.3 Model number 1-2

These models, trained for CG WSD follow exactly what is explained in the paragraph 2.2 and they both use the implementation 1 for the *f*.

3.4 Model number 3-4-5

Also these models are trained for CG WSD but they use instead the implementation 2 for *f*.

3.5 Model number 6

In order to leverage the capacity of LSTM of capturing contextual information, in this experiment, a bidirectional LSTM [7] with 2 layers on top of *f* is applied before to use the L_1 layer:

$$logits_{CG} = Mask_{CG} * L_1(LSTM(f))$$

Here in this experiment the best basic model (n. 2) of previous experiments is used to generate *f*.

3.6 Model number 7

In this experimental setup, employing the identical architecture as model 2, the focus was on training the model to forecast CG senses. This was achieved by enabling finetuning for the final 4 layers of the BERT model. The rationale behind this approach was to investigate the extent to which the specificity of BERT embeddings could influence the model's performance.

3.7 Model number 8

Building upon the insights provided in paragraph 2.3.1, the model undertakes the dual prediction of both Fine-Grained (FG) senses and Coarse-Grained (CG) senses. The calculation of logits, instrumental in loss computation, is executed as follows:

$$logits_{FG} = Mask_{FG} * L_2(f)$$

$$p_{FG} = \max\left(softmax(logits_{FG})\right)$$

$$p_{CG} = \mathrm{FG2CG}\,(p_{FG})$$

FG2CG is the mapping function that use the given map file to convert a FG sense into CG sense.

The central premise behind this model is its optimization for enhancing FG accuracy. The rationale underlying this approach is that if the model attains high performance levels in FG tasks, it is possible to excel in the CG context as well**.** In essence, a proficient model capable of grasping deeper semantic is expected to reach high performance in more abstract semantic task (clearly only when the task is organized in a formal hierarchical way).

3.8 Model number 9

To encode the idea in paragraph 2.3.2 in a neural architecture, the first step is to consider a Loss function that is the combination of the two losses:

$$Loss = Loss_{FG} + Loss_{CG}$$

This setup enables the joint optimization of weights both to decrease CG and FG Loss.

Regarding the classification layers: just after the BERT encoder as previous discussed, there are two linear layers (L_1 and L_3):

$$logits_{CG} = Mask_{CG} * \left(L_1(f)\right)$$

$$logits_{FG} = Mask_{FG} * L_3(logits_{CG}))$$

The primary classification layer (L1) initiates the classification process. Subsequently, the L3 layer comes into play, facilitating the transformation of logits into a mapping of fine-grained senses; Applying the candidate mask to $logits_{CG}$ it possible to get the CG prediction, and in the same way it is possible to get FG prediction applying the other candidate mask to the $logits_{FG}$. Much like the prior model, the FG prediction is employed to deduce the CG prediction. However, a potential discrepancy between the CG prediction and the deduced CG prediction can arise. In anticipation of this, the deduced coarse-grained predictions are stored, addressing the likelihood of disparities between the two predictions.

This model emits as output:

$$p_{FG} = \max\left(softmax(logits_{FG})\right)$$

$$p_{CG} = \max\left(softmax(logits_{FG}\right)$$

$$p_{CG-Ded} = \mathrm{FG2CG}\,(p_{FG})$$

3.9 Model number 10

Random baseline models which consist of a random choice over the candidate for each word to disambiguate.

4 Results

The results in Table 3 reveal important insights about the models. Here are some key points:

Embedder Impact on F1: The CG F1 scores show moderate variability across models. Surprisingly, the basic version of BERT outperforms the larger version, suggesting that greater complexity doesn’t necessarily yield to superior performance. This phenomenon raises question regarding the trade-off between number of parameters, generalization power and robustness of the outcomes.

Implementation 2 for *f*: Although the F1 scores weren't high, using Implementation 2 for f sped up the forward pass due to smaller tensor dimensions.

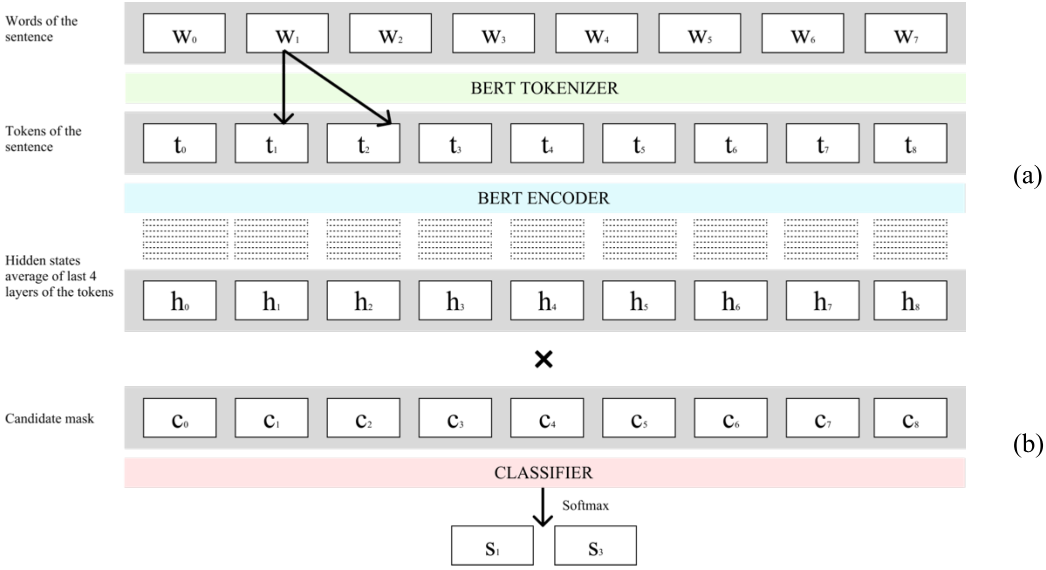
Direct CG Training: Models 2 achieve high CG F1 scores (0.9150), showing that even a basic architecture can handle the task well.

Model 8 attains the highest metric value, despite a minimal difference from Model 2. This highlights that focusing on deeper semantic aspects yields the best performance at higher semantic levels.

Joint Learning (Model 9): jointly learning hierarchical sense divisions, requires further exploration for comprehensive understanding. This experiment prompts further investigation.

In summary, incorporating hierarchical semantic information involves a blend of neural and symbolic approaches, aligning with previous works. Future research will delve into this emerging AI model category.

Figure 1: (a) The encoding part of the model. (b) The mask process.



	Training	**Validation**	**Test**
# Sentences	12339	685	686
# sentence with 1 candidate	7723	451	425
Sentence length mean	40.80	40.95	41.22

Table 1: Statistic for the training validation and test data plus the count of the number of words with 1 candidate and the mean of words length.

ID	**CG F1**	**FG F1**
1	0.9094	/
2	0.9150	/
3	0.8752	/
4	0.8629	/
5	0.8668	/
6	0.8920	/
7	0.9045	/
8	**0.9151**	**0.8041**
9	0.9143	0.7976
10	0.68	0.52

Table 3: Metric value for the models discussed in the experiments section. The ID column indicated the model identifier. For model 8, is reported the deducted CG because is always higher than the predicted CG metric. Model 10 represent the random Baseline model

ID	**Granularity**	**Embedder**
1	CG	Bert Large Cased
2	CG	Bert Base Cased
3	CG	Bert Large Cased
4	CG	Bert Large uncased
5	CG	Bert Base Cased
6	CG	Bert Base Cased + LSTM
7	CG	Bert Based Cased
8	FG&CG	Bert Based Cased
9	FG&CG	Bert Based Cased

Table 2: Experiment List: 1,2,6,7 with implementation 1 of the embedding, 2-3-4 with implementation 2, 8 with implementation 3 and 9 with implementation 4; Exp 8 has the last 4 layers of Bert finetuned.

References

1. “ Word Sense Disambiguation with Recurrent Neural Networks” – Alexander Popov, Linguistic Modelling Department IICT-BAS
2. “Using BERT for Word Sense Disambiguation” - Jiaju Du, Fanchao Qi, Maosong Sun
3. “Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone”- Michael Lesk. 1986.
4. “An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model” - Basile, Caputo, Semeraro 2014
5. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](Devlin et al., NAACL 2019)
6. [Breaking Through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information](Bevilacqua & Navigli, ACL 2020)
7. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.