Report NLP Homework 3 De Filippis Stefano, 1707247

1. Task

The task to be solved in this homework consisted in creating a model that is able to disambiguate words both in the fine and coarse grained case. In the first task the model has to the predict a babelnet id, while in the latter it has to predict either a wordnet domain or a lexicographic label. In any case the network needs to learn a function $f : O \rightarrow O \cup V$ where O is the input vocabulary of the words in the training set and V is a vocabulary of predefined senses or labels . Since the dataset used to solve the task is only partially annotated, the output vocabulary of the network is given by the union of O and V, so for words not annotated I use as correct labels the word itself during the training phase.

2. Architecture

In order to solve the task defined, I tried different architectures.

2.1 Basic

The basic architecture that was developed was a simple BLSTM with 2 layers (one forward and one backward) with hidden size of 512, learning rate of 0.02, dropout of 0.4. On top of it there is an embedding layer given by the elmo architecture that creates an embedding of the words in input of size 1024.

2.2 Attention Mechanism

In this type of architecture I used as building block the previous network and added an attention layer before the final soft-max layer. In both cases the final context vector is concatenated to each output of the BLSTM for the sequence and this is used as input of the soft-max layer.

2.2.1 Simple Attention

The attention mechanism employed was the one reported in the Raganato et al., 2017. Check image 1 to see the equations implemented. Where c is the context vector computed as the weighted sum of the hidden state vectors $h_1, ..., h_T$. $H \in R^{n \times T}$ is the matrix of hidden state vectors, with n the hidden state dimension and T is the input sequence length.

2.2.2 Bahdanau Attention

In a second approach I implemented the Bahdanau Attention mechanism from the *Bahdanau et al. 2016* paper. Check image 2 to see the equations implemented. Where h_t is the concatenation of the BLSTM layers outputs and h_s is the concatenation of the hidden cell states of the last layer units if the BLSTM.

2.3 Multilearning

This architecture is again build upon the network just described and it tries to improve model generalization by defining and optimizing different losses while keeping a shared representation, which in this case is the output of the BLSTM and the attention mechanism. In practice, after the attention mechanism there are n soft-max layers, with n equals to the number of task to learn and the final loss that is optimized during the training is the weighted average of the n.

2.3 Hierarchical Multilearning

This last solution, instead, was inspired by the *Sanh et al. 2018* paper. Up to the attention layer the architecture is the same as the previous ones, but now the losses are optimized asynchronously. Moreover, the lexicographic predictions of each words are encoded as binary vectors and concatenated, along with the encoding of the words most frequent lexicographic labels and the context vector, to the output of the BLSTM layer and fed as input to the soft-max layer of the fine grained task. The learning pseudo-code is the following:

```
for e in epochs:
    for b in batches:
        b ← emb(b)
        optimize(lex_loss,b)
        lex_predict ← get_lex_prediction(b)
        MFC_lex ← get_most_frequent_lex(b)
        en_lex_predict ← encoding(lex_predict)
        en_MFC_lex ← encoding(MFC_lex)
```

fine_input ← concatenate(en_lex_predict,en_MFC_lex,b)
optimize(fine_loss,fine_input)

In this way I try to exploit the advantages of multi learning and the coarse grained predictions to boost the fine grained ones. The idea of one-hot-encoding the lexicographic labels was taken from *Melacci et al. (2018)* as the idea of adding the most frequent lexicographic domain to the input of the fine grained soft-max layer. Their limitation, though, was the fact that to compute their features they only used the wordnet frequency synset heuristic, since they would have needed another WSD system to have the lexicographic domains. In my case, instead, I am learning both WSD tasks at the same time, so I can access lexicographic predictions while learning the fine grained task. The problem in my approach, though, was the fact that at the beginning of the training these predictions are very bad. Therefore, I added also the most frequent lexicographic domain in order to balance this problem and gain more useful information at the beginning, meanwhile, during the training the lexicographic predictions would become more accurate and they will add more useful information that also help to alleviate the problem of the biased predictions towards the most frequent sense. Check image 3 to see a visual representation of the architecture.

3. Dataset

The data-set used for the training is the SemCor data-set while the development set was senseval3

4. Evaluation

For the evaluation I tested the models on the following dataset: semeval2007, semeval2013, semeval2015 and senseval2. For each annotated words I first use wordnet to extrapolate all the synsets associated to the couple lemma, pos and then consider as candidates only the sense that are present in the output vocabulary. Finally from the soft-max output I get the maximum between these candidates. If the lemma of an annotation was not seen at training time I simply use the MFS back-off strategy.

5. Experiments

In the following section I will report and comment all the experiments conducted

5.1 Prepocessing and Input

In solving the task I did not conduct any particular preprocessing of the input text apart from lemmatization. Moreover, I tried three different approaches in the handling of the sequence maximum length . First I tried to use a variable max length value equals to the longest sentence in the batch. In this way I did not loose any information but training was considerably slow. In a second approach I first estimated the mean length of the sentences in the data set and used as maximum length the mean plus 10. in this way the parameter was set to 30 and I truncated at the end the longer sentences. Following this approach decreased the performances but it was decisively faster. The last try consisted in using still the fixed value for the maximum length, but I split longer sentences in sub sentences and created new entries in the dataset. Doing so, the performances slightly increased with respect to the previous case but were still lower then using a variable parameter, because I lost some context information present in the whole sentence. Moreover, training time was quite high, since I doubled the dataset. In the end I settled for the second approach because the model was faster to train and the following approaches were able to better generalize also with the limited and truncated data.

5.2 Attention Mechanisms

Not both the attention mechanisms helped the model to improve the performances. In particular, the first attention mechanism lowered in general the performances of the model with respect to the base model. On the contrary, the Bahdanau attention mechanism proved to be a successful addition to the original model and increased the performances of 1% in general. Therefore, in the successive models I settled for Bahdanau solution as attention mechanism.

5.3 Synset Embedding

Another experiment I did was to use the the synset embeddings I developed in homework2. I mainly tried two different approaches. In the first one for each target word in a sentence I took all the wordnet synsets and averaged their embeddings, which was then concatenated to the elmo embedding as input to the the network. This solution, however, was not successful and decreased considerably the performances of the network. In the second approach, instead, I only concatenated the embedding of the most frequent sysnset associated to a

certain lemma and POS tag. In this case, the results were quite good and the model increased the performances of about 1% with respect to the basic model + Bahdanau attention.

5.4 Synset Predictions

Following the work presented in Vial et al. 2019 and due to the improvements that I gained in homework 2, I tried to improve the model generalization power by substituting during the learning the prediction of the sense key with simply the synset. Therefore, every sense label was replaced with the wordnet synset it belonged to and the learning happened on this new reduced task. The main idea behind this approach is the fact that different sense in the same synset actually have the same meaning and so they disambiguate the same concept. By developing this approach I could reduce the output vocabulary and drastically improve the model performances of almost 3% with respect to the previous case. Obviously, during the evaluation, given the synset prediction of a lemma, I can easily go back to the sense key through the wordnet interface and build predictions for the original fine grained task.

5.5 Multilearning

The task that were learned during this approach were the fine grained task and the lexicographic domain task. Moreover, for the fine grained task I tried both the sense key predictions and the more general approach of the synset predictions. In both cases the model improved the performances with respect to the BLSTM+attention, but the improvements were bigger when the synset predictions were used for the fine grain task. In this case the increase of performances was of 0.3% while in the other case the increase was almost of 0.2%. I also tried to do multi learning with synset labels and sense key but this approach was not successful and actually decreased the performances with respect to the other approaches.

5.6 Hierarchical Multilearning

This approach, along with the synset predictions and the addition of the synset embeddings was the architecture that yielded some of the best results. Also in this case I tried both the synset predictions and sense key predictions for the fine grained task and in both cases the performances increased with respect to the multi learning architectures. In this case, though, the biggest improvement was registered when the sense keys were used for the fine grain task and it amounted to almost 2%. In the other case, instead, the increase in performances was of 0.2%. I think this difference in improvement is due to the fact that with the sense key predictions the model tend to suffer of over-specification and this architecture really helped to improve over generalization while the employment of synset prediction already quite increased the generalization power of the model.

5.7 Coarse Grain Boosting

Since the success registered with the hierarchical multi learning, I tried to increase the performances of the BLSTM+attention model for the fine grained task by passing information about the most frequent lexicographic domain of the target words in a window of 3. Indeed, for each target word I added as information the encoding of the most frequent lexicographic domain of the target word itself, the preceding one and the next one. I used the synset prediction for the fine grained task and try to concatenate the additional information either to the context vector or to the word embeddings. In both cases the performances decreased and this may be due to the bias of the predictions towards the most frequent sense, but the second approach behaved slightly better than the other one.

5.8 Coarse Grain Task

Also for the coarse grain task I trained dedicated models. The wordnet domains task was too prone to overfitting due the over presence of the factotum class and I stopped the experiments to the BLSTM + Bahdanau attention. On the other hand, the lexicographic task benefited from the multi learning approach which yielded the highest performances, with an increase of more than 1% over the BLSTM+attention model, even if performances of hierarchical multi learning was sligthly inferior.

5.9 Best Model

For the fine grained task the best model was obtained with the hierarchical multi learning architecture where synset predictions were used for the fine grained task. For the wordnet domain the best model was the BLSTM+Bahdanau attention while for the lexicographic task the best model was obtained with the multi learning architecture where sysnet predictions were used for the fine grained task.

Experiment Result on Fine Grain Task (in bold is the final chosen model)

					Dev.	Test	Test	Test	Test	All	All	All	All	
Architecture	**emb.**	**epoch**	**lr**	**drop.**	**SE3**	**SE2**	**SE07**	**SE13**	**SE15**	**Nouns**	**Verbs**	**Adj.**	**Adv.**	**All**
BLSTM variable max lenght	Elmo 1024	30	0.02	0.4	64.8%	67.9%	56.5%	61.4%	66.2%	65.6%	56.5%	78.8%	80.1%	64.7%
BLSTM fixed max lenght	Elmo 1024	30	0.02	0.4	64.2%	68.4%	58.7%	61.2%	65.2%	65.9%	54.8%	79.1%	81.2%	64.6%
BLSTM split longer sentences	Elmo 1024	30	0.02	0.4	64.5%	68%	57%	61.2%	64.7%	65.7%	56%	78.6%	80.5%	64.6%
BLSTM+att. Raganato	Elmo 1024	30	0.02	0.4	65.4%	65.7%	54.1%	59.9%	65.8%	64.6%	53.6%	81.6%	78%	63.6%
BLSTM+Bahd.	Elmo 1024	30	0.02	0.4	66.1%	68.8%	58.5%	61.6%	67.5%	66.7%	56%	82%	81.5%	65.6%
Multi learning (sense+lex)	Elmo 1024	30	0.02	0.4	66.8%	68.6%	57.4%	61.1%	67.5%	66.8%	56.8%	80.6%	79.8%	65.8%
BLSTM+Bahd. +avg synset emb.	Elmo 1024 syn. 500	30	0.02	0.4	64.1%	65%	54.5%	59.7%	64.5%	65%	51.9%	77.7%	77.5%	62.8%
BLSTM+Bahd+ MF synset emb.	Elmo 1024 syn. 500	30	0.02	0.4	66.2%	69.3%	61.1%	63.6%	65.9%	68.1%	56.1%	80.9%	80.9%	66.2%
Hierarchical Multi Learning (sense+lex)	Elmo 1024 lex emb 45 (one hot)	30	0.02	0.4	68.1%	68.7%	60.4%	65.6%	68.2%	69.7%	54.7%	85.9%	81.5%	67.2%
BLSTM+Bahd. (synset predict)	Elmo 1024	30	0.02	0.4	69.6%	71.2%	59.1%	64.1%	68.9%	69.6%	57%	85%	82.9%	68.1%
BLSTM+Bahd. (synset predict) +lex window in attention	Elmo 1024 lex emb 45 (one hot)	30	0.02	0.4	69.1%	69.7%	59.8%	61.7%	68.8%	67.6%	56.5%	86.3%	84.1%	67%
BLSTM+Bahd. (synset predict) +lex window word emb.	Elmo 1024 lex emb 45 (one hot)	30	0.02	0.4	68.6%	70.1%	61.3%	64.1%	68.7%	68.7%	55.9%	85.4%	81.8%	67.3%
Multi learning (synset+lex)	Elmo 1024	30	0.02	0.4	69.8%	71.2%	60.7%	64.9%	68.7%	69.9%	57.3%	85.7%	83.8%	68.4%
Multi learning (synset+sense)	Elmo 1024	30	0.02	0.4	69.1%	69.3%	61.8%	63.8%	68%	68.6%	56.5%	84%	82.4%	67.3%
Hierarchical Multi Learning (synset+lex)	**Elmo 1024 lex emb 45 (one hot)**	**30**	**0.02**	**0.4**	**69.8%**	**71%**	**61.3%**	**64.2%**	**71.7%**	**69.9%**	**57.5%**	**87%**	**83.8%**	**68.6%**

Experiment Result on Coarse Grain Task (in bold is the final chosen model)

					Dev.
Architecture	**emb.**	**epoch**	**lr**	**drop.**	**SE3**
BLSTM+Bahd. Wordnet dom.	**Elmo 1024**	**30**	**0.02**	**0.4**	**87.3%**
BLSTM+Bahd. lex dom.	Elmo 1024	30	0.02	0.4	83%
Hierarchical Multi Learning (sense+lex)	Elmo 1024 lex emb 45 (one hot)	30	0.02	0.4	84.1%

Multi learning (synset+lex)	**Elmo 1024**	**30**	**0.02**	**0.4**	**83.9%**
Hierarchical Multi Learning (synset+lex)	Elmo 1024 lex emb 45 (one hot)	30	0.02	0.4	83.8%

Image 1

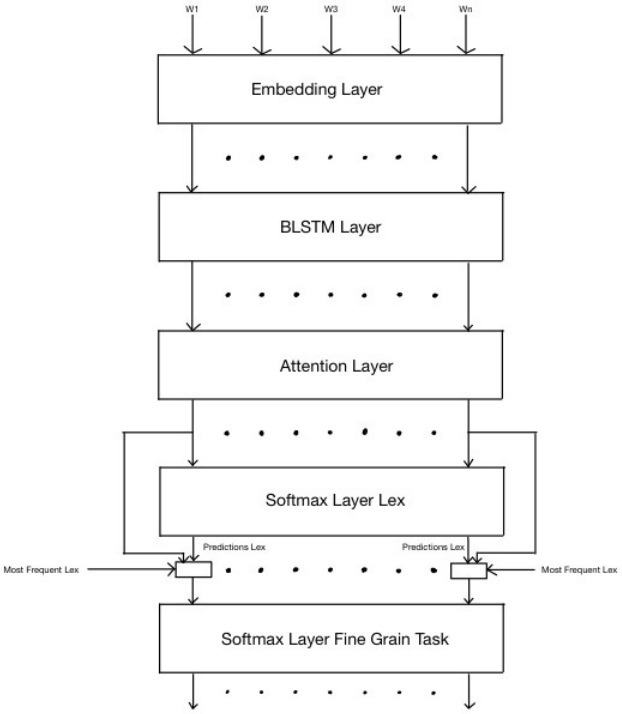
$$\mathbf{u} = \omega^T \tanh(H)$$
$$\mathbf{a} = softmax(\mathbf{u})$$
$$\mathbf{c} = H\mathbf{a}^T$$

Image 2

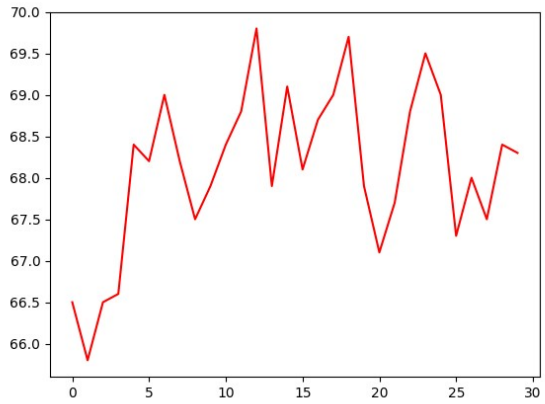
$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]}$$
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]}$$
$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]}$$

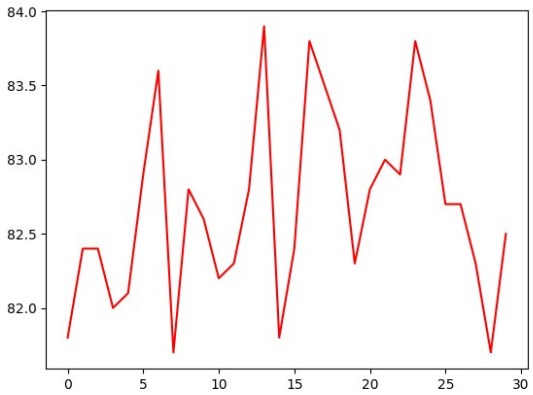
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1}\boldsymbol{h}_t + \boldsymbol{W_2}\bar{\boldsymbol{h}}_s\right)$$

Image 3



Plot of Fine Grain Task and Lexicographic Domain Task Development Set F1 score on Best Model





References

- *Neural Sequence Learning Models for Word Sense Disambiguation,* Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli
- *Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources,* Stefano Melacci , Achille Globo , Leonardo Rigutini
- *Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation* , Loïc Vial Benjamin Lecouteux Didier Schwab
- *A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks* ,Victor Sanh , Thomas Wolf , Sebastian Ruder
- *NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE,* Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio