Word Sense Disambiguation

NATURAL LANGUAGE PROCESSING PROJECT REPORT

Francesco Frattolillo: 1783651
Sapienza University of Rome
Artificial Intelligence and Robotics
francescofrattolillo95@gmail.com

1 Word Sense Disambiguation

In the Natural Language it often happens that we use the same word but with completely different meaning. A simple example is the following:

- I will not *bow*

- Give me that *bow*

As we can see, the word *bow* in the first sentence stands for an action, while in the second sentence the same word stands for an item. The disambiguation of a word given its context it's a pretty simple task for a human, while it can be a really though one for a machine.
The task of determining the correct meaning of a target word in a given context is usually referred as **Word Sense Disambiguation**(WSD). Most of the algorithms used in the WSD task can be mainly divided in two big categories: supervised and knowledge-based. Despite the fact that the supervised approaches usually works better and achieve state of the art performance, they require a huge number of sense annotated data in order to be trained. Knowledge based approaches use lexical knowledge bases such as dictionary and thesauri, and hypothesize that context knowledge can be extracted from definitions of words.
Since we are going to use the supervised method it's useful to have a better understanding of the task. The main idea is to find an alternative substitute word for a target word that is ambiguous in a context. A set of words that can be used interchangeably in the same context is called a *synset*. In this work we are going to use Wordnet, that is a lexical database where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets).

2 Data and Preprocessing

The training data used is the SemCor that are contained in the *Raganato's Framework*. In our corpus, there will be certain words that occur very frequently, such as *the, is*, and so on, and there are certain words that occur infrequently. Since these frequent words usually are needless in the disambiguation tasks, while the infrequent words can play an important role, to maintain a balance between these two, we applied subsampling of frequent word, with threshold of 10^{-5} and we discarded all the words with a lower frequence than 6. Successively we created different vocabularies for all the lemmas, part of speech and categories of every instance word found in the training set.

3 Model

The model used in this homework is a simple bi-lstm network similar to the one used in Raganato et al[1]. The main difference is that we removed the word embedding structure in the first layer. The problem with word embedding is that we have only

one representation for a given word, hence it becomes hard to describe the possible polysemy of that word in different context. In order to solve this problem, we used a sense embedding layer. The sense embedding architecture that we have used is called **elmo**(Embeddings from Language Models). It is able to create a different embedding for a word, given different surrounding words.
Another big problem in word embedding, is that we have a representation for every word found in the training data, but we cannot process a different word never seen during training. Elmo solve this problem thanks to the fact that its embedding is based on character instead of words. The first model that we build is only able to retrieve the fine-grained sense of a word. The second model has three different dense layers as output and three different losses that are summed together:

- Fine-grained dense layer and a fine-grained sense loss.

- Part of speech dense layer and part of speech loss.

- Coarse-grained (wordnetDomain or Lexicographer) Dense and coarse-grained loss.

The ability of solving different (but connected) task with the same underlying architecture and features, its called *Multi-Task Learning.* One important thing to notice is that if we find a new world during evaluation, our system will always give an answer thanks to the elmo layer. This is really good when we have to predict a part of speech or a coarse-grained sense, but it can be a problem when it comes to the fine-grained sense. In order to solve this problem we check ,during the fine-grained prediction, if the word is in our vocabulary, otherwise we will use the Wordnet API in order to predict the most frequent sense of that word, given its lemma and its pos.

4 Experiment and Settings

The main parameters that can be modified in our architectures are

- hidden size of the bi-lstm layer (suggested values: [64,128,516])

- batch size of the sentences (suggested values: [10,32])

- learning rate (suggested values: [0.1,0.01,0.001])

In order to check the parameters value the main idea was to set a grid search in order to test different order of magnitude for these parameters. Since the unavailability of a powerful computer, i was able to run only one of the configurations (batch_size = 10, hidden_size = 64 and learning_rate = 0.01) for 79 epochs.
During this training the accuracy over the training and development data was really high, while during the predictions it was lower (results are reported in the tables). This was probably due to an overfitting.
The first training was done using wordnetDomain coarse-grained senses as coarse-grained output. During the second training, done on the lexicographer coarse-grained i have

added a dropout probability over the lstm layers, in order to reduce the risk of overfitting. The elmo embedding dimension was fixed and it cannot be changed through the model that is available on tensorflow hub.
The sentences in our training data are organized in batches and they are padded with respect to the longest sentence in the batch.
I've used the SemCor data both as training and development set. Every eight training iterations there are two development iterations, where we just compute the loss and the accuracy values without update the optimizer. Obviously all the dataset have been shuffled every epoch.

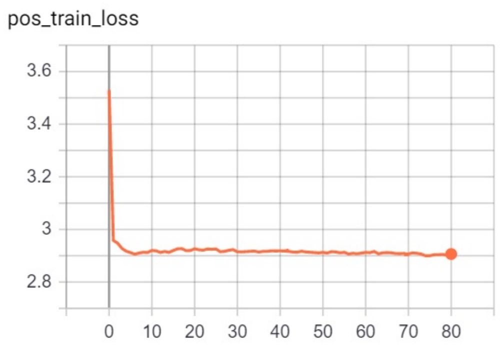
5 Multilingual Model

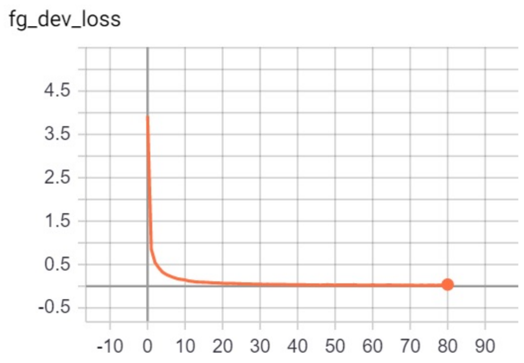
The multilingual model is a slightly modification of the previous one. Instead of the elmo module, we have used the ElmoForManyLanguages module. The pre-trained weights for different languages can be found at **ElmoForManyLanguages**.
In order to use this network we have restored part of the weights of the MultiTask model architecture.
A file that maps word to babelnet_ids has been created for every language. Babelnet groups words in different languages into sets of synonyms, called Babel synsets.
Thanks to the file *babelnet2wordnet.tsv* given as resource for this homework, we can retrieve the corresponding wordnet_id.
Obviously we have also created different vocabularies for every language. The network works and it's able to predict every word, but for better results it probably requires a little fine-tuning for some epochs.

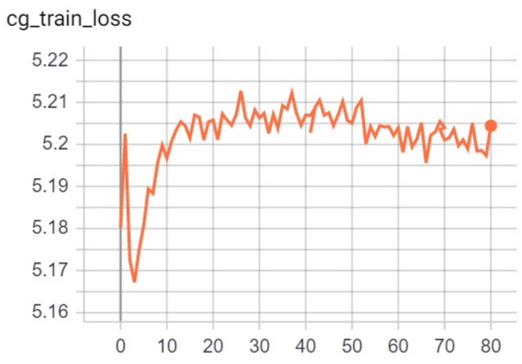
6 Possible Idea

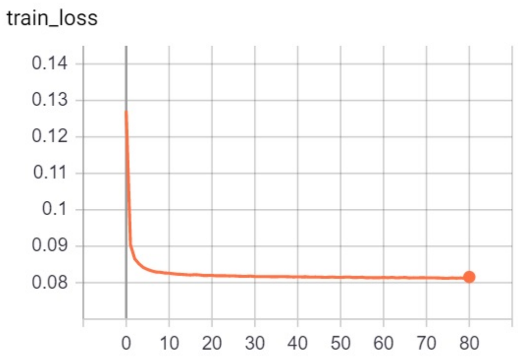
One of the ideas that i've had but that i was unable to implement for time and technologies issue, was to introduce an adversarial module.
Since adversarial networks have proven successfully in different tasks, the idea was to use a discriminator network that can receive as input both a disambiguated sentence or the output of our main network, and it gives as output a value that indicates how likely the input is a disambiguated sentence.
Then this value is used in addition to the main network loss. The two network will train at the same time and improve each other.

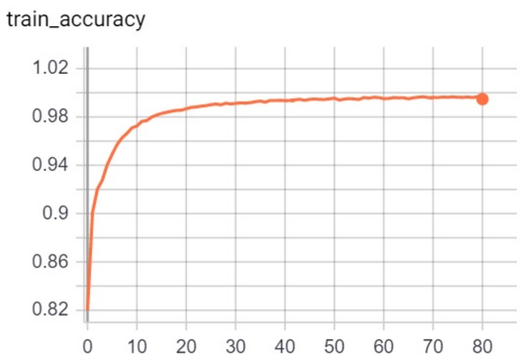
Evaluation Results					
	semeval2007	semeval2013	semeval2015	senseval2	senseval3
MultiTask Model	0.858	0.832	0.646	0.648	0.698

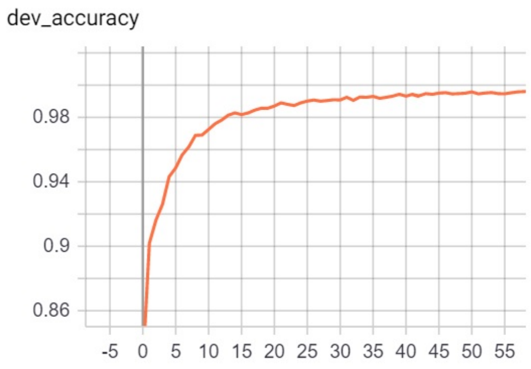












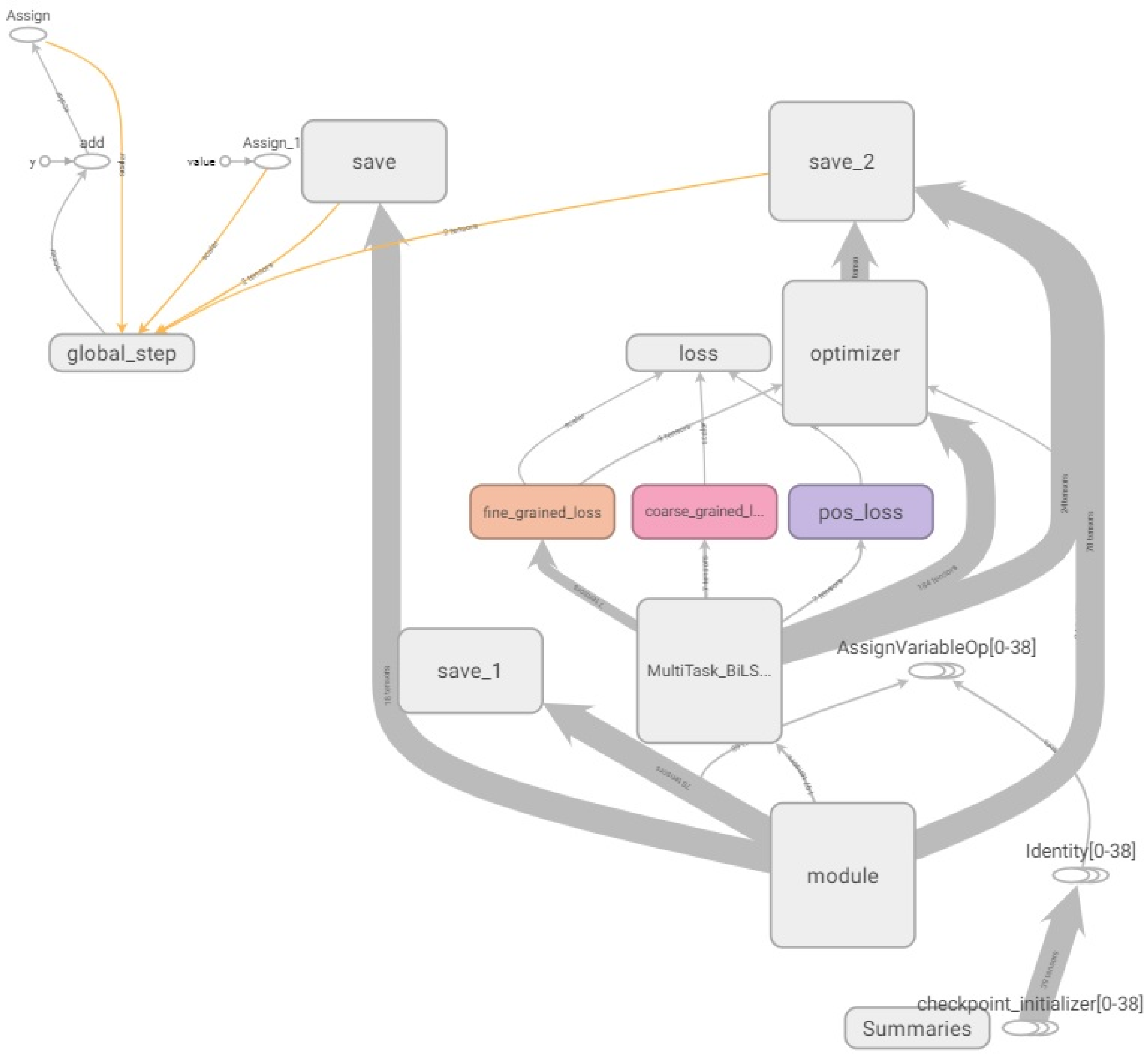


Figure 1: Tensorflow graph

References

[1] Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli. *Neural Sequence Learning Models for Word Sense Disambiguation*, 2017.

[2] Matthew E.Peters,Mark Neumann, Mihit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. *Deep contextualizee word representations.*

[3] Ignacio Iacobacci, Mohammad Taher Pilehvar and Roberto Navigli. *SensEmbed: Learning Sense Embeddings for Word and Relational Similarity.*

[4] Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli. *Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison.*