Natural Language Processing
Multilingual Word Sense Disambiguation

Professor: Roberto Navigli
Student: Ahmed ElSheikh - 1873337

17 January 2020

Contents

1 Introduction

The purpose of this report to demonstrate different approaches for word sense disambiguation task on English and in a multilingual setting. Inspiration was drawn from multiple papers [1] [2] [3] [4] [5] [6] [7] [8] in order to create the experiments showcased here, modify, build and evaluate the performance and the results achieved.

2 Data Preprocessing

2.1 English WSD

I used SemCor which is a manually semantically annotated corpus made up of 37,176 English sentences with all its sense annotations built from the sense inventory WordNet, it provides POS tags and lemmas for each word, words can be found in 2 categories, which are Word Format (WF) - which means the word is not ambiguous - or an instance - which means the word has more than a sense and needs disambiguation. SemCor's 37k sentences contain ≈26k unique fine grained senses, 151 domains, 47 lex-names, and ≈36k POS-LEMMA pairs [1].
This dataset was parsed using *lxml.etree.iterparse()* to form sentences of lemmas only $training - data$ and lemmas combined with instances $labeled - training - data$. Since the dataset was already cleaned, minimal preprocessing was required such as lowering all the tokens, to avoid data sparsity. Sense annotations are provided in Wordnet sense keys, which were mapped into BabelNet synsets for fine-grained disambiguation, while for the coarse-grained disambiguation I used WN domains and lexicographer IDs.
From our data (unlabeled, labeled) I built my input vocabulary and output vocabulary (total of ≈ 58k unique tokens) constructed using a word-index dictionary based on all the unique lemmas. Another 2 were built for the pos-tags, and lex IDs (3 vocabularies in total); they were used in the Multitask-learning task. It was mentioned earlier that we have our training data divided into labeled and unlabeled data, the unlabeled data was set of tokens, the labeled data was also tokens, however, the words with of the category instance were changed to $< lemma >\_< sense\_key >$ same applies for lex and pos tagging $< lemma >\_< LEX\_ID >$ and $< lemma >\_< POS >$ respectively, each pair of these was assigned only its candidates for both cases (fine and coarse)-grained cases so to increase the restriction on what the model trains, this restriction makes it more difficult to the model to learn given that the ambiguity as per word increased. To have batches of equal size of the data we added first $'n'$ sentences again to the end of the given dataset. The mappings from Wordnet to BabelNet or from BabelNet senses to Lex IDs or Domains were fetched using the mappings provided.

2.2 Multilingual WSD

OneSec was used which is an automatically annotated data annotated with word meanings in 6 languages - I only used 5 of them EN, IT, FR, ES, DE. This dataset follows the previously mentioned dataset in terms of token categories (Word Formats and instances), all 5 languages were parsed one after the other resulting in ≈ 1.2M sentences following the same format of $< lemma >\_< POS >$ containing ≈ 297k unique tokens, and ≈ 3.5k unique sense tokens, 47 lex-names, 5 pos-tags, shared across the 4 languages [IT, FR, ES, DE], and integrating SemCor English sentences. OneSec was chosen over Train-O-Matic based on its robust approach that was able to achieve better scores than Train-O-Matic as well as better scaling across languages and domains.[10]

3 Models

The WSD problem was formulated in the terms of a tagging problem, having an input sequence $x =< x_1, x_2, ..., x_T >$, which should be tagged and produced output sequence $y =< y_1, y_2, ..., y_T >$. In this section, we discuss different models built

3.1 BiLSTM

This is the base architecture for the rest of the networks/models within this report, it is composed of an embedding layer, followed by Unstacked BiLSTM, Dense, Masking, Softmax prediction layer. It was used for comparison between models implemented. It used to retrieve the relations between the sequence of tokens (as per sentence) in both directions hence the name.

[1]Most frequent synset was be.v.01, Most frequent Domain was factotum, Most frequent Lex was adj.all. Added to, the dataset was imbalanced; due to the overwhelming presence of the factotum Domain, 10 lexes appeared the most

3.2 Bi-LSTM with Attention

Attention layer was added to the baseline model, to exploit the fact the some tokens in the sequence might be of a higher importance than the others. The Self attention layer was trained to force the model focus on the most informative parts of the sentence, in order to condense it to form its 'context_vector' -which is conditioned on both the states and the output, this is due to its ability to check previous and future information, this helps us more in getting the correct predictions. It was added to the model just before the prediction layer

3.3 Seq2Seq with Attention

Given the problem's formulation, this task can be approached in terms of mapping a sequence (input) to another (output). Encoder learning from the given input sequence using an Attention BiLSTM architecture, passing the 'context_vector' generated by the attention layer -as mentioned earlier- to the decoder (another LSTM) generating output sequence, based on the input and context vector. Output of the decoder passes through dense layer, before, passing through Masking and the final Softmax layer.

3.4 Multitask Learning

Our aim is to improve the model capabilities in its task, by exploiting the common properties across multiple tasks, Which was done by training the model to predict correct sense, pos, LEX, this was done by adding 2 more Softmax layers to the model, using the same loss function (Categorical cross-entropy) and sum over all losses (Sense_losses, LEX_losses, POS_losses). This was one approach to feed the model pos-tags, lex-ids, senses. Yet, another approach was taken feeding the model Lex-IDs, Domains and Senses as the sense contain POS-tag in it. This was done by feeding the data to the model to a Stacked BiLSTM first BiLSTM was focused only on the coarse grained predictions, then the 2nd BiLSTM was forced to focus only on the fine-grained prediction, allowing it to improve the model's result while leaving 1t layer to focus on features extraction, solely.

3.5 Masking

In order for the model to train properly and converge to a solution, it was required to reduce the solution space dimensions; as trying to get probabilities for more than 58k words would result in infinitesimal difference, A Mask was built to mask all the irrelevant words while predicting each word, this was done by creating a 3D mask (np.ndarry) [Batch_size $\times$ max_len $\times$ classes_num], filling it with negative infinity for irrelevant words and zeros for the candidate synsets - zeros were also assigned for a set of 10 random other words (regardless of the category) to increase the numerical stability, during loss computation - which were obtained from the WordNet API, in the case of multilingual WSD, same procedure was followed. This mask was added to the senses logits before passing the logits to Softmax layer.

3.6 Embedding Layer

We used different approaches for the Embedding Layer, training an Embedding layer on the given dataset (whether SemCor or OneSec) while training the model to extract the text features. Each word in our dictionary was represented by a 400D vector (embeddings size). Since, our vocabulary was approximately 58k unique tokens, so we had 58k $\times$ 400 embeddings matrix.

Secondly, GloVe (global vectors for word representation) performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space[9]. The pretrained uncased model used was trained on 2B tweets total of 27B tokens with a vocabulary of 1.2M tokens, output embeddings vector of 200-dim.

Added to, using ElMo (Embeddings from Language Models) providing models with pre-trained contextualized embeddings, consequently, for each sample the model provides the contextual embedding of its token, as providing the lemma only without the POS tag means losing information. Finally, the embeddings size used is 1024. learnable aggregation of the 3 layers, and a fixed mean-pooled vector representation of the input.

Yet another pretrained language model, BERT (Bidirectional Encoder Representations from Transformers) producing contextual representations of words in a text. The model used was a large cased model with 12 attention heads and embeddings Size of 768 (small BERT model), fine tuned on SemCor by learning the aggregation of the embeddings from the last 4 layers. Both ElMo and BERT were used to prove the hypothesis of that integrating contextual embeddings with word embeddings enhanced the WSD task. Both of them, can be found as 2 seperate modules on tensorflow hub, and the were wrapped in a tf.keras layer; in order to utilize them. For the BERT embeddings sentences had to be preprocessed and tokenized using Bert Tokenizer.

3.7 Multilingual WSD

For the multilingual WSD task, SensEmBert vectors, which is context-Enhanced Sense Embeddings for Multilingual WSD. Each sense id for 146312 was represented by a vector of the size 2048, and was fine tuned on the dataset. Those 146,312 senses were shared across multiple languages.

The data used for the multilingual WSD was preprocessed and fed the same way as for the English WSD, however only the first 10K sentences were taken from each language; because of the computational resources limitations. To build the mask in the training phase for this task, the file 'lemma2synsets' file provided was used to fetch the candidate synsets as per word (of instance category). As well as, using the same file for the fallback strategy MFS 'Most-Frequent-Sense'.

SensEmBert vectors are constructed by fetching all the relevant textual information from Wikipedia as per concept in the semantic network, followed by computing the vector representation for each relevant word of the target synset then merging the contextual information computed earlier and enriches it with additional knowledge from the semantic network, so as to build an embedding for the target sense [6].

Multilingual WSD implemented the set of experiments were reduced to only experimenting the Zero Shot Learning (passing to the model no data, only multilingual embeddings) and Passing both multilingual data and multilingual embeddings to the model.

4 Expiremental Setup

The sentences were fed to the model in batches, they were generated online, and padded as per batch to make sure not to feed the system all the data with maximum length which might be an extreme value as per the given dataset, resulting in consuming the memory and time. All the words of WF -word format- category to a single class denoted by $< WF >$ to further reduce our training time and increasing the model's performance.

4.1 Training and testing

Data was fed to the model in batches of 16 sentences, padded as per the maximum length of a sentence in the batch, and masked as per batch, in order to optimize our usage of the computational resources at our disposal; as we only had 8 sentences of size 285 words and 8 sentences of 1 word, this was the case with SemCor dataset. Models were using Adadelta optimizer(using default values of learning rate, decaying rate), all of models were built with 128 neurons. Dropout rates and recurrent dropout rates were fixed to 0.2. All the training was done using Google's cloud computing resources, Colab. Models were trained on 5 epochs using a batch size of 8 sentences/batch due to resources limitations

For prediction, the models - used in English WSD - were predicting BabelNet Synset, Lexicographer label. And for the multilingual WSD, outputs BabelNet Synsets only due to computational resources limiations. For the words that were with no synset assigned, the Most-Frequent-Sense backoff strategy was implemented assigning for the out-of-vocabulary tokens $< UNK >$. And, as per the tokens with an uassigned domains was set to $factotum$, same for unknown lexnames.

In order to test the model, all the evaluation datasets were used, provided by the WSD evaluation framework SensEval-2, SensEval-3, SemEval-2007, SemEval-2013 and SemEval-2015. SE07. In the case of the Multilingual WSD using SemEval-2013 eval dataset (each for the 5 languages) the F1-Score as a performance metric using the predictions provided by the model (not provided mappings). Results are discussed in the next section.

[2]Experiments that were not mentioned here, nor in the appendix, showed poor results.

5 Results

All results and graphs of the mentioned experiments are provided in the tables below[2] . The best performing model in English WSD was the multitask Attention_BiLSTM model with ElMo pretrained embeddings. This proves that integration of LexNames, Pos-tags enabled model to generalize better, utilizing more pieces of information. ElMo pretrained embeddings helped the model achieve better F1_scores (in comparison to the regular embeddings layer and GloVe pretrained embeddings); as ElMo's word representations are functions of the entire input sentence[11]. Finally, with more computational resources it will be easier to perform a larger grid-search and train the model on even more data, will results in improved results. Same model was used for Multilingual WSD except that the embeddings layer was fed with the pretrained model *SensEmBert* as mentioned earlier[3]. Using embeddings layer, GloVe showed very poor results in comparison with ElMo embeddings for the English WSD. Finally, Zero Shot learning on Multilingual WSD showed poor results (so they were not mentioned.

6 Future Work

Integrating CRF - Conditional Random Fields, which are statistical models used for predict sequences of labels giving sequences of input (same formulation as WSD problem), considering also the context. CRF are able to encode known relationships between observations. On the other hand, On the other hand, CRF can be too computationally complex given the high number of labels and sparsity.
Implement loss mask, where we multiply zeros for words of Word Format category, so our model focuses more on learning the synsets.
Changing the networks' architectures, find optimal values for the hyper-parameters utilizing an optimization method like population based training optimization.
Using another language models such as XLNet for the embeddings layer. Replicate the idea of having a continuous label space [4], use Universal Sentences Encoder pretrained embeddings
As known, the more data the better for our model, parsing more datasets.

References

[1] Raganato, Delli Bovi and Roberto Navigli. Neural Sequence Learning Models for Word Sense Disambiguation paper.

[2] Stefano Melacci, Achille Globo, Leonardo Rigutini. Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources

[3] Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang and Zhifang Sui. Incorporating Glosses into NeuralWord Sense Disambiguation

[4] Sawan Kumar, Sharmistha Jat, Karan Saxena, Partha Talukdar. Zero-shotWord Sense Disambiguation using Sense Definition Embeddings

[5] Daniel Loureiro, Allıpio Mario Jorge. Language Modelling Makes Sense: Propagating Representations through WordNet for Full-CoverageWord Sense Disambiguation

[6] Bianca Scarlini, Tommaso Pasini, Roberto Navigli. SENSEMBERT: Context-Enhanced Sense Embeddings for MultilingualWord Sense Disambiguation

[7] Zero-shot Learning of Classifiers from Natural Language Quantification. Shashank Srivastava, Igor Labutov, Tom Mitchell

[8] Attention Is All You Need paper by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

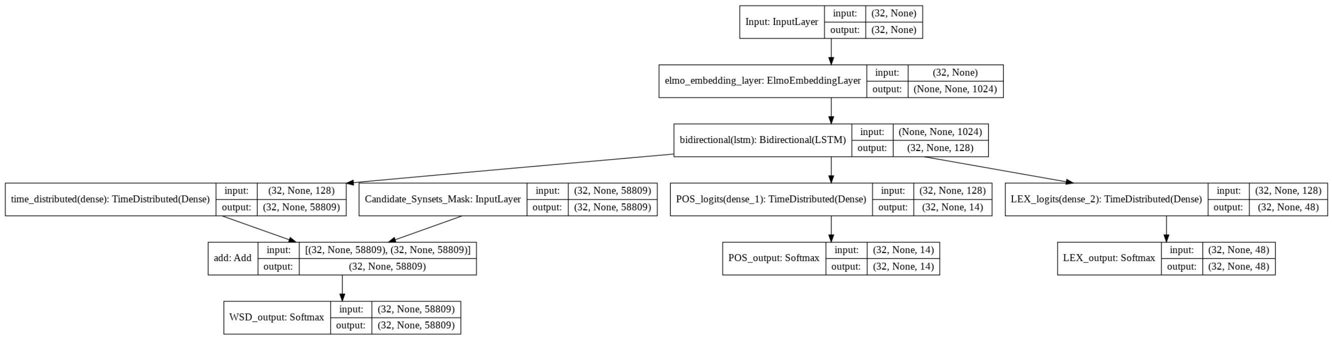
[9] GloVe Website https://nlp.stanford.edu/projects/glove/

[10] Just "OneSeC" for Producing Multilingual Sense-Annotated Data by Bianca Scarlini, Tommaso Pasini and Roberto Navigli https://www.aclweb.org/anthology/P19-1069.pdf

[11] Deep contextualized word representations, Matthew E. Peters†, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer

[3]Could not test Multitask Seq2Seq with Elmo Embeddings; due to computational complexity and limitations of computational resources.

A English WSD



Input: InputLayer | input: (32, None) | output: (32, None)
elmo_embedding_layer: ElmoEmbeddingLayer | input: (32, None) | output: (None, None, 1024)
bidirectional(lstm): Bidirectional(LSTM) | input: (None, None, 1024) | output: (32, None, 128)
time_distributed(dense): TimeDistributed(Dense) | input: (32, None, 128) | output: (32, None, 58809)
Candidate_Synsets_Mask: InputLayer | input: (32, None, 58809) | output: (32, None, 58809)
POS_logits(dense_1): TimeDistributed(Dense) | input: (32, None, 128) | output: (32, None, 14)
LEX_logits(dense_2): TimeDistributed(Dense) | input: (32, None, 128) | output: (32, None, 48)
add: Add | input: [(32, None, 58809), (32, None, 58809)] | output: (32, None, 58809)
POS_output: Softmax | input: (32, None, 14) | output: (32, None, 14)
LEX_output: Softmax | input: (32, None, 48) | output: (32, None, 48)
WSD_output: Softmax | input: (32, None, 58809) | output: (32, None, 58809)

Figure 1: EN_WSD Model's Architecture

Model's Architecture	SE07	SE2	SE3	SE13	SE15
Attention_BiLSTM	64.3	70.1	69.1	67.9	65.4
Seq2Seq_Attention	62.2	68.6	67.4	66.8	63.7

Table 1: F1_scores (Fine Grained)

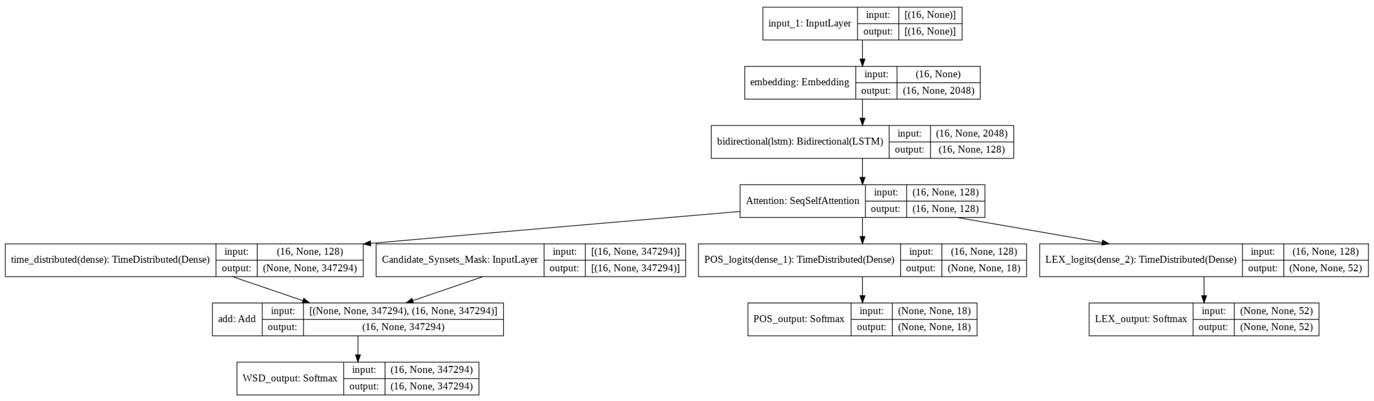
Model's Architecture	SE07	SE2	SE3	SE13	SE15
Attention_BiLSTM_ElMo	85.7	87.2	83.7	82.3	84.9
Seq2Seq_Attention_ElMo	81.5	84.1	82.6	78.8	79.92

Table 2: F1_scores Coarse Grained (Domains)

Model's Architecture	SE07	SE2	SE3	SE13	SE15
Attention_BiLSTM_ElMo	78.6	87.0	84.1	82.7	82.3
Seq2Seq_Attention_ElMo	74.8	82.9	79.1	77.4	78.2

Table 3: F1_scores Coarse Grained (Lexes)

B Multilingual WSD



input_1: InputLayer | input: [(16, None)] | output: [(16, None)]
embedding: Embedding | input: (16, None) | output: (16, None, 2048)
bidirectional(lstm): Bidirectional(LSTM) | input: (16, None, 2048) | output: (16, None, 128)
Attention: SeqSelfAttention | input: (16, None, 128) | output: (16, None, 128)
time_distributed(dense): TimeDistributed(Dense) | input: (16, None, 128) | output: (None, None, 347294)
Candidate_Synsets_Mask: InputLayer | input: [(16, None, 347294)] | output: [(16, None, 347294)]
POS_logits(dense_1): TimeDistributed(Dense) | input: (16, None, 128) | output: (None, None, 18)
LEX_logits(dense_2): TimeDistributed(Dense) | input: (16, None, 128) | output: (None, None, 52)
add: Add | input: [(None, None, 347294), (16, None, 347294)] | output: (16, None, 347294)
POS_output: Softmax | input: (None, None, 18) | output: (None, None, 18)
LEX_output: Softmax | input: (None, None, 52) | output: (None, None, 52)
WSD_output: Softmax | input: (16, None, 347294) | output: (16, None, 347294)

Figure 2: Multilingual_WSD Model's Architecture

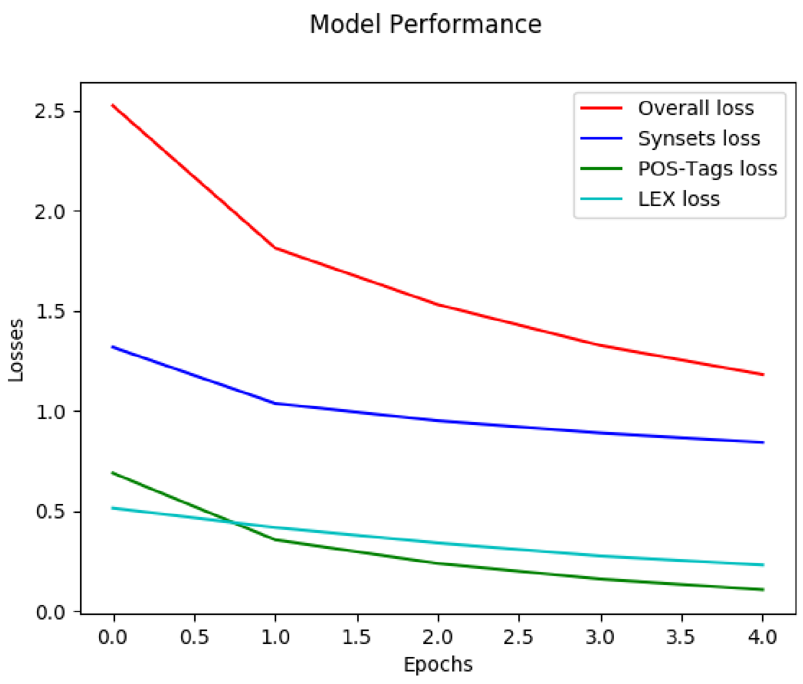


Figure 3: Multilingual_WSD Model's loss

Model's Architecture	FR	IT	ES	DE
Attention_BiLSTM	57.2	49.3	61.48	58.32

Table 4: F1_scores Fine Grained Multitask Multilingual WSD, test set: SE13