NLPHW3
Word Sense Disambiguation (WSD)

Manoochehr Joodi Bigdello - 1860273

Project Overview

Word sense disambiguation (WSD) is the ability to identify the meaning of words in context. WSD is important topic in NLP because most of the human languages are ambiguous, so that many words can be interpreted in multiple ways depending upon the context where they appears and it's a challenge for machines to understand the meanings (senses) behind the words. This task is Training multi-inventory WSD system capable of performing both fine-grained and coarse-grained WSD. Dataset is Raganato format sense-tagged English corpora (semcor and semcor+omsti) with mappings from WordNet to BabelNet synsets and BabelNet to LexNames and WordNetDomains.

Preprocessing

In preprocessing phase, after parsing xml files and perform all mappings, getting data from WordNet and converting all data to id (we have dictionary for RawWords, SensesWords, POS, Lex, WnDomain), we build 4 different masks to filter our results of logits that we are going to feed in "sparese_softmax" layer and argmax for prediction.

- Pos2Sense mask: it's a (size of POS dictionary)*(size of SensesWords dictionary) to mask every possible sense to its POS tag.
- Lex2Sense mask: it's a (size of Lex dictionary)*(size of SensesWords dictionary) to mask every possible sense to its LexName tag.
- WnDomain2Sense mask: it's a (size of WnDomain dictionary)*(size of SensesWords dictionary) to mask every possible sense to its WnDomain tag.
- lemma2Sense mask: this is a little bit different from other masks, because of Tensorflow limits in GPU we have to perform this mask inside batches so it's a 3D mask (size of batch)*(size of sentences with padding)* (size of SensesWords dictionary) to mask every possible senses of a word to its lemma, this helps us to perform final evaluation as argmax of only the candidate synsets for the lemma we are trying to disambiguate. (E.g. car#noun has C=[bn:00007309n, bn:00015785n, bn:00015786n, bn:00015787n, bn:00014462n] then synset = argmax(prob[C]))

Models

We address this problem using of neural architecture bidirectional Long Short Term Memory (LSTM). We used all-words model which rely on sequence learning. We also used POS, LexName and WnDomain tags to improve the performance. As mentioned in homework assignment file we implement the MFS as our Back Off Strategy in case of no prediction for instances.

As embedding layer we used sense embedding's from our NLPHW 2 assignment and also embedding layer of Tensorflow for understanding the impact of sense embedding's on WSD.

We used BD-LSTM layer and then concat all outputs of forward and backward LSTM and feed that as input for other BD-LSTM layer on top of this one.

We also put attention layer + hidden layer on top of our sense BD-LSTM for getting information of all words for better prediction (as mentioned in paper).

For possible performance improvement we also used hierarchical modeling. We used POS, Lex and WnDomain tags as masking for results of sense layer and then feed the masked results to sparse_softmax, we used sparse_softmax because each lemma word can have exactly one sense. For minimizing we used gradient based minimizing with clip and our total_loss to minimize was the sum of all sparse_softmax parts.

We implement 7 different models for comparison:

1- **MFS Tagger**: as base line we get MFS synset of lemma from WordNet.
2- **Basic Tagger**: we just used an Embedding layer + BD-LSTM layer for Sense + attention layer
3- **Multitask$_{\mathbf{POS}}$ Tagger**: we used Embedding layer + BD-LSTM layer for POS (we compute loss for this layer with sparse_softmax as loss_pos) + BD-LSTM layer for Sense (input of BD-LSTM is output of POS BD-LSTM) with attention and hidden layer on top + POS masking (we use Pos2Sense mask to filter the possible senses based on POS) + sparse_softmax to compute loss_sense and predictions. We minimize the sparse_softmax cross entropy loss and also regularize the weights using L2 regularization. We also clipped the gradients.
4- **Multitask$_{\mathbf{Lex}}$ Tagger**: same as Multitask$_{POS}$ we just use LexNames instead of POS with Lex2Sense mask.
5- **Multitask$_{\mathbf{WnDomain}}$ Tagger**: same as Multitask$_{POS}$ we just use WnDomains instead of POS with WnDomain2Sense mask.
6- **Multitask$_{\mathbf{POS+Lex}}$ Tagger**: we used Embedding layer + BD-LSTM layer for POS (we compute loss for this layer with sparse_softmax as loss_pos) + BD-LSTM layer for LexNames (we compute loss for this layer with sparse_softmax as loss_lex) + BD-LSTM layer for Sense (input of BD-LSTM is output of POS BD-LSTM) with attention and hidden layer on top + POS and LexName masking (we use Pos2Sense and Lex2Sense mask multiplication to filter the possible senses based on POS and LexNames) + sparse_softmax to compute loss_sense and predictions. We minimize the sparse_softmax cross entropy loss and also regularize the weights using L2 regularization. We also clipped the gradients
7- **Multitask$_{\mathbf{POS+WnDomain}}$ Tagger**: same as **Multitask$_{\mathbf{POS+Lex}}$** we just use WnDomains instead of LexNames and WnDomain2Sense mask instead of Lex2Sense mask.

Training

In training we used different hyperparameters as shown in table 1, then we choose the best one, for tuning we used 5000 sentence of SemCor dataset and semeval2007 as development set. For optimizer we decided to use the AdadeltaOptimizer with learning rate of 1 as mentioned in reference paper of optimizer. We trained every model in 10 Epoch, we just used the SemCor dataset (37K sentence) for training with 2 mode, first with Tensorflow Embedding Layer and then with Sense Embedding's of NLPHW2 Assignment.

Test phase

We test our models and configurations on all 5 test set, Senseval-2 all-words, Senseval-3 all-words, SemEval 2007, SemEval 2013, and SemEval 2015, we also tested on ALL of them together to have general view of our proposed models. We predict our results in BabelNet format and then used mapping files to get LexNames and WnDomains. We report the results in 3 different cases BabelNetId's, LexNames and WnDomains output format, we also report the results on both with sense embedding's and without sense embedding's. The results were shown in Tables 2-4, bold fonts for comparing between with sense embedding's and without sense embedding's, and red highlighting is for comparing different models.

Conclusion

As we trained our model with different configurations, we can see that the **Multitask$_{\text{POS+WnDomain}}$** model is better than other models and we think it's because in training dataset after mapping the WnDomains are more accurate that LexName, also because of masking possible synsets for candidate lemma makes this model better than **Multitask$_{\text{POS}}$**, **Multitask$_{\text{WnDomain}}$** and **Multitask$_{\text{Lex}}$**. Using of sense embedding's in most of the cases increase the performance but it's not too much and we think that it can be because of our poor model in HW2.

References and Link

1- Neural Sequence Learning Models for Word Sense Disambiguation, Raganato, Delli Bovi and Navigli, EMNLP 2017
2- Tensorflow.com

Charts and Tables

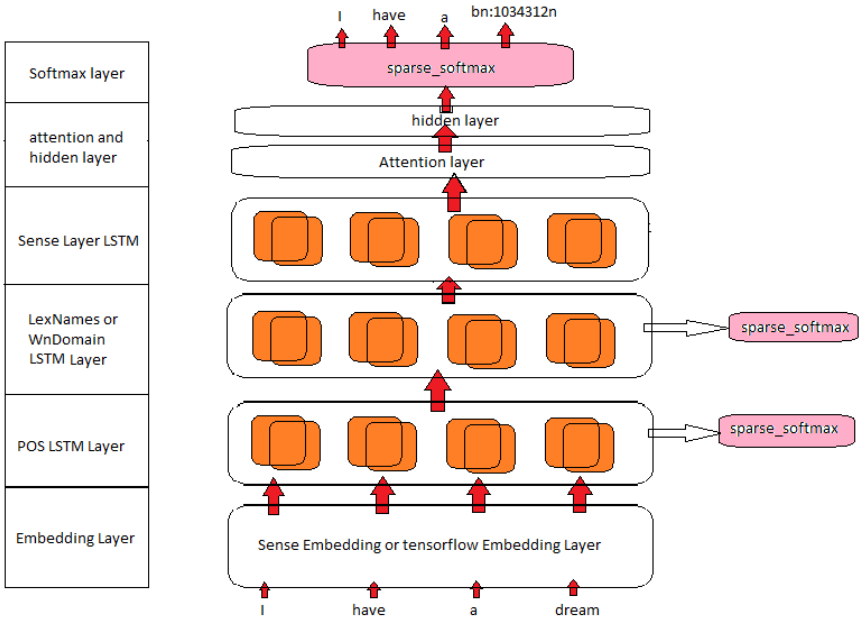

Figure 1. Our General model structure

Table 1. Hyper parameter tuning values

Hyperparametrs	values	Best ones
Batch Size	4, 8, 16	4
CLIP	5, 10	10
Embedding dimension	100	100
Number of LSTM Layers	1, 2	2
Hidden Layers	128, 256	256

Table 2. F1 accuracy of different models on each test dataset with and without sense embedding in **BabelNet** format

	Senseval2		Senseval3		SemEval2007	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	49.2	-	47.9	-	37.1
Basic Tagger	63.5	**64.1**	61.9	**62.2**	48.3	**49.8**
Multitask$_{POS}$ Tagger	**59.1**	57.9	**60.9**	60.2	**49.4**	47.4
Multitask$_{Lex}$ Tagger	**54.2**	53.9	**54.5**	54.4	**48.7**	48.1
Multitask$_{WnDomain}$ Tagger	**58.3**	58.1	**60.3**	58.3	**49.6**	48.7
Multitask$_{POS+Lex}$ Tagger		49		50.6		47.9
Multitask$_{POS+WnDomain}$ Tagger	**63.7**	63	60.7	**60.8**	48.7	**50.7**

	SemEval2013		SemEval2015		ALL	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	62.7	-	44.3	-	**50.5**
Basic Tagger	59.2	**60.9**	59.7	**60**	60.6	**61.4**
Multitask$_{POS}$ Tagger	**58.3**	57.8	**58.2**	57.7	**60.6**	59.8
Multitask$_{Lex}$ Tagger	**58.8**	58.6	51.1	**51.2**	**54.7**	54.4
Multitask$_{WnDomain}$ Tagger	**53.3**	51.4	**52.4**	50.8	**56.3**	55
Multitask$_{POS+Lex}$ Tagger		53.2		46.5		50
Multitask$_{POS+WnDomain}$ Tagger	**60**	58.6	**59.3**	58.7	**60.6**	60

Table 3. F1 accuracy of different models on each test dataset with and without sense embedding in **LexNames** format

	Senseval2		Senseval3		SemEval2007	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	58.7	-	55.5	-	43.7
Basic Tagger	81.1	**81.5**	77.3	**77.6**	67.2	**68.3**
Multitask$_{POS}$ Tagger	**76.7**	75.8	**76.5**	76	**68.3**	67.7
Multitask$_{Lex}$ Tagger	**74.2**	73.4	72.1	**72.4**	67.9	67.9
Multitask$_{WnDomain}$ Tagger	75.6	**76.3**	**76**	73.6	66.8	**67.4**
Multitask$_{POS+Lex}$ Tagger	69.1	**69.5**	**70.8**	70	65.8	**67.2**
Multitask$_{POS+WnDomain}$ Tagger	**81.2**	80.9	**76.3**	75.9	68.3	**69.2**

	SemEval2013		SemEval2015		ALL	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	74.9	-	54.5	-	60
Basic Tagger	70.3	**72.3**	77.4	**78.1**	76.3	**77.1**
Multitask$_{POS}$ Tagger	**70.2**	69.8	**76.9**	76.4	**76.7**	76.2
Multitask$_{Lex}$ Tagger	**70.8**	70.5	**71.9**	71.7	**72**	71.9

Multitask$_{WnDomain}$ Tagger	**64.5**	62.9	**70.5**	70.15	**71.9**	71.1
Multitask$_{POS+Lex}$ Tagger		64.1		66.5		67.9
Multitask$_{POS+WnDomain}$ Tagger	**71.6**	70.7	77.7	**78.4**	**76.6**	76.2

Table 4. F1 accuracy of different models on each test dataset with and without sense embedding in **WnDomains** format

	Senseval2		**Senseval3**		**SemEval2007**	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	53	-	57.4	-	69.4
Basic Tagger	58.8	**59**	67.2	**67.5**	83.9	**84.1**
Multitask$_{POS}$ Tagger	**55.3**	54.2	**66.4**	65.8	84.1	**84.6**
Multitask$_{Lex}$ Tagger	57.9	**58.3**	**67.7**	67	84.5	**85**
Multitask$_{WnDomain}$ Tagger	54.7	54.7	**65.3**	63.9	84.3	**84.6**
Multitask$_{POS+Lex}$ Tagger		55.1		63.4		81.5
Multitask$_{POS+WnDomain}$ Tagger	**58.7**	58.4	**65.7**	65.5	85.2	**85.5**

	SemEval2013		**SemEval2015**		**ALL**	
	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding	With sense embedding	Without sense embedding
MFS Tagger	-	74.8	-	49.7	-	59.6
Basic Tagger	72.2	**73.9**	57.1	**57.5**	65.3	**65.9**
Multitask$_{POS}$ Tagger	**71.5**	70	**56.6**	56.4	**65.5**	64.6
Multitask$_{Lex}$ Tagger	**71.1**	70.7	56.3	**56.4**	**65.1**	64.7
Multitask$_{WnDomain}$ Tagger	64.6	**64.7**	**52.6**	51	**61.2**	60.7
Multitask$_{POS+Lex}$ Tagger		66		50.7		60.7
Multitask$_{POS+WnDomain}$ Tagger	71	**71.7**	**56.8**	56.6	**64.7**	64.7