INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

Word Sense Disambiguation

by

Shanu Kumar

EE391A: Undergraduate Project

under supervision of
Dr. Harish Karnick
Department of Computer Science and Engineering

April 2018

Abstract

Word Sense Disambiguation

Word sense disambiguation (WSD) is the ability to identify the meaning of words in context. We address this problem using series of end-to-end neural architectures using bidirectional Long Short Term Memory (LSTM). We propose two variants for WSD: an end-to-end word specific neural model and all-words neural model. In the word specific models we have to train models for every disambiguation target word. We addressed this issue using the all-words model which rely on sequence learning. We also used POS tags to improve the performance. We tried different variants of attention mechanisms for the all-words model. Performance was boosted by using convolutional neural networks (CNN) which captures local features around the words that is normally what humans do for predicting the senses. We further improved the performance using hierarchical models. We used POS tags as hierarchy and used two variants as soft masking and hard masking.

Contents

Chapter 1

Introduction

Most of the human languages are ambiguous, so that many words can be interpreted in multiple ways depending upon the context where they appears. For example consider the following sentence:

(a) I like **rock** music.

(b) That mountain is solid **rock**.

The word **rock** clearly has different meanings in both the sentences.

While most of the time humans do not even think about the ambiguities of language, but in most of the natural language processing tasks like Summarization, Relation Extraction, Machine Translation, etc word embedding are used as the basic building block for the model. Word embedding like *word2vec* (Mikolov et al. [1]) and *Glove* (Pennington et al. [2]) trained to optimize a generic task independent objective function and all words have single vector irrespective of its different senses. Thus, by dealing with lexical ambiguity a WSD model brings numerous benefits to a variety of downstream tasks and applications. We address the problem of WSD using bidirectional LSTMs. We propose two completely different models:

- An end to end neural model for every word that has to be disambiguated by focusing on an individual word rather than a global model shared between all words like Kågebäck and Salomonsson [3].
- A single end to end neural model (Raganato et al. [4]) for joint disambiguation of the target text as a whole in terms of a sequence labeling problem.

Chapter 2

Background

2.1 Word embeddings (GloVe)

Word embeddings is a way to represent words as real valued vectors in a semantically meaningful space. Global Vectors for Word Representation (GloVe), introduced by Pennington et al. [2] is a hybrid approach to embedding words that combine a log-linear model, made popular by Mikolov et al. [1], with counting based co-occurrence statistics to more efficiently capture global statistics. Word embeddings are trained in an unsupervised fashion, typically on large amounts of data, and is able to capture fine grained semantic and syntactic information about words. Therefore pre-trained word vectors like GloVe are used to initialize the input layer of a neural network or some other NLP model.

2.2 Bidirectional LSTM

Long short-term memory (LSTM) is a gated type of recurrent neural network (RNN). LSTMs were introduced by Hochreiter and Schmidhuber [5] to enable RNNs to better capture long term dependencies when used to model sequences. The flow of information is instead regulated using multiplicative gates which preserves the gradient better than e.g. the logistic function. The bidirectional LSTM, (Graves and Schmidhuber [6]) is an adaptation of the LSTM where the state at each time step consist of the state of two LSTMs, one going left and one going right. For WSD this means that the state has information about both preceding words and succeeding words, which in many cases are absolutely necessary to correctly classify the sense.

2.3 WordNet Senses

WordNet senses are stored in Lexicographer Files each of which have related words. For example, the file noun.food contains all nouns pertaining to food and drinks. There are 45 such files numbered from 00 to 44. WordNet represents senses with the help of sense keys. These are defined as follows (Theses definitions have been taken with reference from [7].

lex_sense = ss _type:lex_filenum:lex_id:head_word:head_id

- **ss_type** is a one digit number between 1 and 5 which represents the Synset of the sense. Synset can be considered equivalent to basuc POS tags and are one of of *Noun[1], Verb[2], Adjective[3], Adverb[4], Adjective Satellite[5].*
- **lex_filenum** lies between 00 and 44 and is the is the lexicographer file number of the WordNet sense. Senses are categorized into Lexicographer files based on syntactic category and logical groupings. Some examples:
  - File 04 represents nouns denoting acts or actions.
  - File 39 denotes perception related verbs. (seeing, hearing, feeling)
- **lex_id** is used to identify a sense within a lexicographer file. The lex_id and the lemma together provide a unique identity to every sense in the lexicographer file. Note that, there can be multiple senses of the same word in the same file.
- **head_word** and **head_id** are present in the sense key only if the synset is Adjective Satellite i.e. ss_type = 5.

  Satellite adjectives are adjectives with a basic meaning which when appended with some context makes more sense. For example, dry which when appended with a context enhance the meaning.

  dry + climate = arid
  thirsty = dry + throat
- **head_id** with the head word uniquely identifies the sense within a file, similar to lex_id.

Chapter 3

Related Work

3.1 Context Representation

To make any meaningful predictions about the senses of a word, the context that the word appears must be carefully taken into account. The context is generally given a numerical representation, but this is not the case with all algorithms. Some of the classical approaches towards context representation are as follows Navigli [8].

- Given a window of text $w_{nk}, \ldots, w_n, \ldots, w_{n+k}$ surrounding a focus word w_n, a context vector can be represented using bag of words inside the window.
- To improve upon the previous approach, sequential information can be added to the bag of-words context, by assigning weights to each word based on its position. Thus, we usen $\{(w_{n-k}, f(-k)), (w_{n-k-1}, f(-k+1)), \ldots, (w_n, f(0)), \ldots, (w_{n+k-1}, f(k-1)), (w_{n+k}, f(k))\}$. f is a function that maps position to weight. The weights must become larger as we move closer to the query word and taper out as we move away from the query word.
- Information like Part-of-Speech (POS) tags can be added to the word in context.
- Using chunking - Divide the sentence into smaller parts which are syntactically correct and can stand by themselves, examples include dividing the sentence into phrases (noun, verb, adjective, adverb, ...).
- Parse the sentence to generate a parse tree, process this parse tree and use information from it as the context. The parse tree stores important syntactic information which may help in WSD.

3.2 Supervised Approaches

- Using context vector train a classifier like (Naive Bayes, SVM, kNN, Decision Tree, Feed Forward NNs and Ensemble Methods).
- Semi-supervised approaches include those using BootStrapping.

3.3 Unsupervised Approaches

- **Context-clustering**: Represent the cluster as a context vector from the co-occurrence matrix, reduce dimensionality, and clusterize these vectors. While testing assign the cluster with highest similarity.
- Another way is to use weighted co-occurence graphs.

3.4 Knowledge-based methods

- These methods use a thesaurus/dictionary (e.g. WordNet) to add more information.
- **Lesks Algorithm**: Lesks algorithm is a classical standard algorithm. Lesks algorithm measures similarity as the overlap between the gloss of the context and word. $gloss(w)$ represents (bag of) words in the definitions of w. $gloss(context(w))$ represents union of glosses of all words in context. Lesks algorithm checks how much the textual definitions of words in context overlap with the definition of each sense for the word to be disambiguated. The sense with the highest gloss overlap with the context is predicted as the current sense.

$$Similarity_{Lesk} = |gloss(w) \cap gloss(context(w))|$$

Chapter 4

Word specific Model

Given a document and the position of the target word, i.e. the word to disambiguate, the model computes a probability distribution over the possible senses corresponding to that word.

4.1 Basic Model

Let the words in a sentence denoted by $\mathbf{x} = (x_1, \ldots, x_n)$, where n is the length of the sentence. The word vectors are computed using pretrained word embedding of GloVe (Pennington et al. [2]) and represented as $\mathbf{W^x} = (W_1^x, \ldots, W_n^x)$. There are two layers of bidirectional LSTM different for every word which compute the hidden states for every word $\mathbf{h^m} = (h_1^m, \ldots, h_n^m)$ for the target word m in the sentence. This will be the input to the attention layer which computes the context vector $\mathbf{c^m}$ for sentence considering the target word at the position m. Basic model is shown in the figure 5.1.

$$\mathbf{h_t^m} = \tanh(\mathbf{h^m})$$

$$\mathbf{a^m} = (a_1^m, \ldots, a_n^m) = softmax(W_c^m \mathbf{h_t^m})$$

$$\mathbf{c^m} = \sum_{i=0}^{n} a_i^m h_i^m,$$

where $W_c^m \in R^d$, d is the dimension of the hidden states of the LSTM.

$$\mathbf{y_m^s} = softmax(W_f^{sm} \mathbf{c^m} + b_f^{sm}),$$

is the predicted distribution over senses for the word at position m, where W_f^{sm} and b_f^{sm} are the weights and biases for the softmax layer corresponding to the target word type at position m.

4.2 Dropout

Dropout is a regularization technique by (Srivastava et al. [9]). We used dropout in Bidirectional LSMT and fully connected layers by dropping a word with a probability. This subsequently improved the performance of the model.

4.3 Loss

We minimize the cross entropy loss and also regularize the weights using L2 regularization. We also clipped the gradients if it is greater than fixed value.

$$loss = \sum_{i \in C^m} y_i^{tm} \log(y_i^{sm})$$

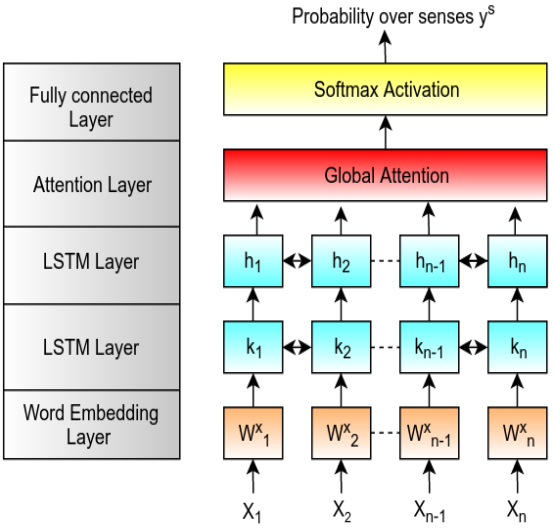

FIGURE 4.1: Basic Word specific Model (Model 1)

4.4 Modifications

- **Basic Model+POS Tags**: In this model we are first predicting the POS-tags of each word in the sentence using softmax to have a better grammatical understanding of the sentence which will improve the context vector $\mathbf{c^m}$, hence helping in predicting the sense of the target word. This model is shown in the figure 5.2
- **Basic Model+POS Tags+CRF**: We tried to improve the accuracy in predicting senses by improving the accuracy of the POS tags. Conditional Random Fields (CRF) (Lafferty et al. [10]) are better in handling sequences, so we replaced the Softmax classifier by CRF classifier.

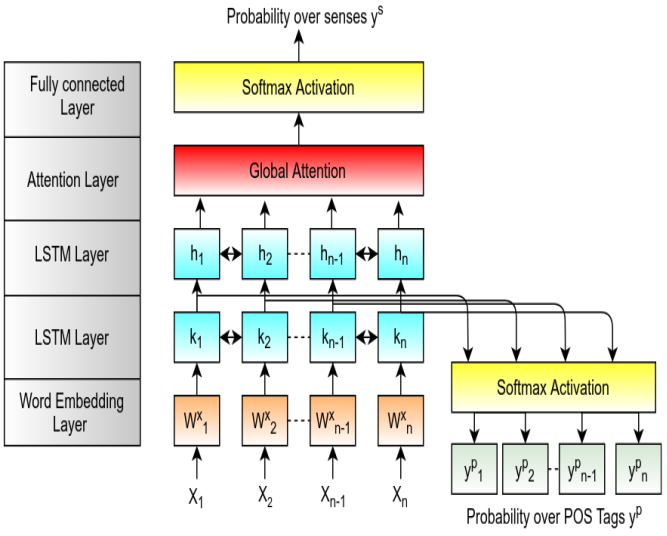

FIGURE 4.2: Basic Word specific Model + POS Tags (Model 2)

4.5 Hierarchical Model

Consider the following examples:

(a) *Set* the volume of the speaker to highest.

(b) The new *set* of rules prohibits smoking on campus.

In the first sentence, the word *set* is used as a verb while in the second sentence it is used as a noun. Thus, it is sufficient to predict the Part-of-Speech (POS) of set whileb disambiguating these two senses. However, that may not always be sufficient.

Consider the following example, "The traitor *set* fire to the palace." Here too, set is used as a verb, but the meaning is to start (a fire) rather than to adjust (the volume). To disambiguate we need more information rather than just POS tags. But first predicting POS Tags develops a hierarchical structure and further reduces the classification problem to less number of probable senses of a target word.

We developed an end to end hierarchical model shown in figure 5.3. In WordNet, every sense have a POS Tag like Noun, Verb, Adjective and Adverb. We used softmax classifier for predicting the POS Tags for the target word, then used a mask $\mathbf{M^m} \in R^{sXp}$, where s is the number of senses for the target word at the position m and p is the number of POS Tags for these senses. Let suppose the first row is Noun, then every class of sense which is noun has 0 as its value and other senses have a negative number like 10.

$$\mathbf{y_m^s} = softmax(W_f^{sm}(\mathbf{c^m} + \mathbf{M^m}) + b_f^{sm}),$$

, is the predicted distribution over senses for the word at position m.

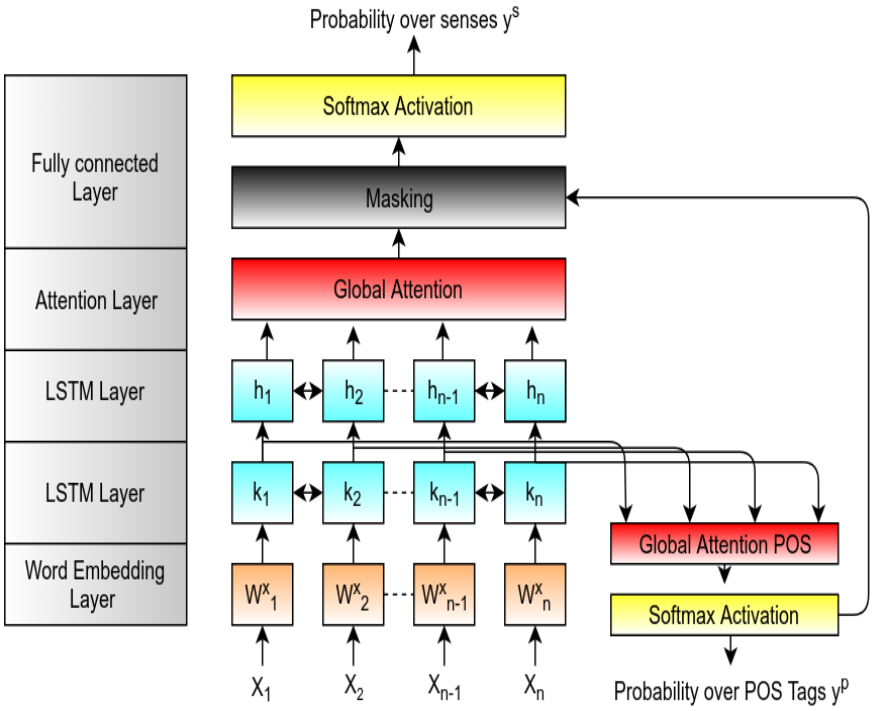

FIGURE 4.3: Hierarchical model (Model 4)

Chapter 5

All-words Model

Instead of framing a separate classification problem for each given word, we followed the work of Raganato et al. [4] which models the joint disambiguation of the target text as a whole in terms of a sequence labeling. We also proposed different variants of all-words model for improving the performance.

5.1 Basic Model

Let the words in a sentence denoted by $\mathbf{x} = (x_1, \ldots, x_n)$, where n is the length of the sentence. The word vectors are computed using pretrained word embedding of GloVe (Pennington et al. [2]) and represented as $\mathbf{W^x} = (W_1^x, \ldots, W_n^x)$. There are two layers of bidirectional LSTM different for every word which compute the hidden states for every word $\mathbf{h} = (h_1, \ldots, h_n)$ in the sentence. This will be the input to the attention layer which computes the context vector $\mathbf{c}$ for sentence. Basic model is shown in the figure 5.1.

$$\mathbf{h_t} = \tanh(\mathbf{h})$$

$$\mathbf{a} = (a_1, \ldots, a_n) = softmax(W_c\mathbf{h_t})$$

$$\mathbf{c} = \sum_{i=0}^{n} a_i h_i,$$

$$\mathbf{H} = (H_1, \ldots, H_n) = ([c, h_1], \ldots, [c, h_n])$$

where $W_c \in R^d$, d is the dimension of the hidden states of the LSTM and $[a, b]$ denotes the concatenation of two vectors a and b.

$$\mathbf{y^s} = (y_1^s, \ldots, y_n^s) = softmax(W_f^s\mathbf{H} + b_f^s),$$

is the predicted distribution over senses for all the words in the sentence.

5.2 Loss

We minimize the cross entropy loss and also regularize the weights using L2 regularization. We also clipped the gradients if it is greater than fixed value.

$$loss = \sum_{i=0}^{n} \sum_{j \in C} y_{i,j}^{t} \log(y_{i,j}^{s})$$

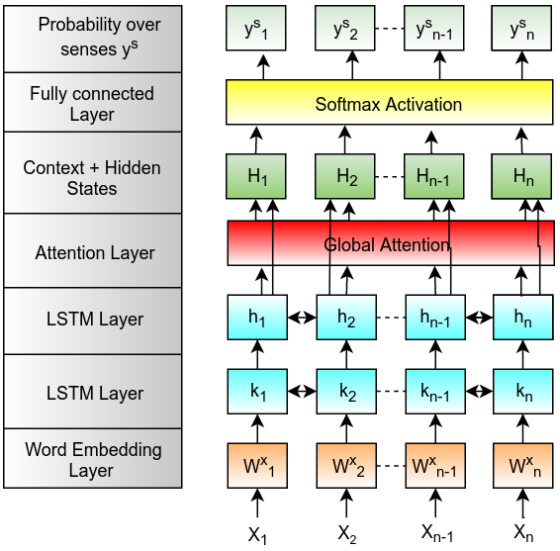

FIGURE 5.1: All-Words Model

5.3 Modifications

- **All-words Model+POS Tags**: In this model, we are first predicting the POS-tags of each word in the sentence using softmax.
- **All-words Model+POS Tags+Local Attention**:Most of the time we only need a window around the target word for predicting its sense. Instead of computing context vector for the sentence, we compute context vector around a window for the target word. Thus for every word there will be a different context vector using this, model will predict its senses. Model shown in figure 5.2
- **All-words Model+POS Tags+Local Attention+Hidden States**: Using only context vector around a windows doesn't improve the performance. but

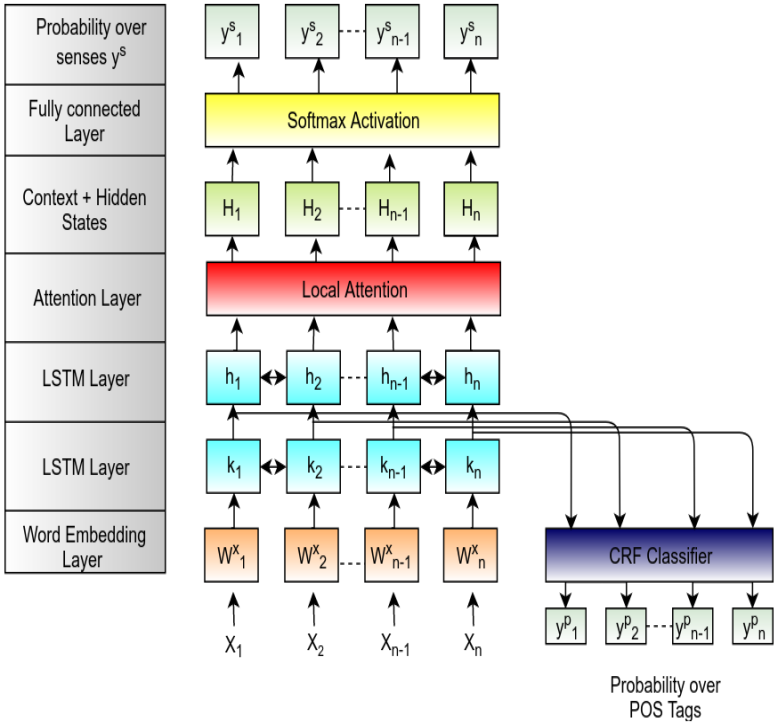

FIGURE 5.2: All-words Model+ Local Attention

concatenating the context vector of a word and its hidden statesh_i improves the performance. Model is depicted in figure 5.3

- **All-words Model+POS Tags+Gated Attention**: If model can learn when to use context vector and hidden states or their combination, then it can outperform the previous model. But results were similar to the previous one.

$$g_i = \sigma(W_c^g \mathbf{c_i} + W_h^g \mathbf{h})$$

$$H_i = g_i h_i + (1 - g_i) c_i$$

- **All-words Model+POS Tags+Local Attention+Hidden States+CRF**: Used CRF for improving the accuracy of POS Tags.

- **All-words Model+POS Tags+CNN**: When we tried the local attention by taking a window around the word, it didn't improve the model as we taking the window around the hidden states of BiLSTMs. But if the window is applied to the first layer i.e Word Vectors it makes more sense, as hidden states contain all the information up to that word. So, we tried convolutional neural networks (CNN) on the word vectors with a window size of 5 and extracted features around the word within the window (Kalchbrenner et al. [11]). This model outperform all the previous models, shown in the figure 5.4

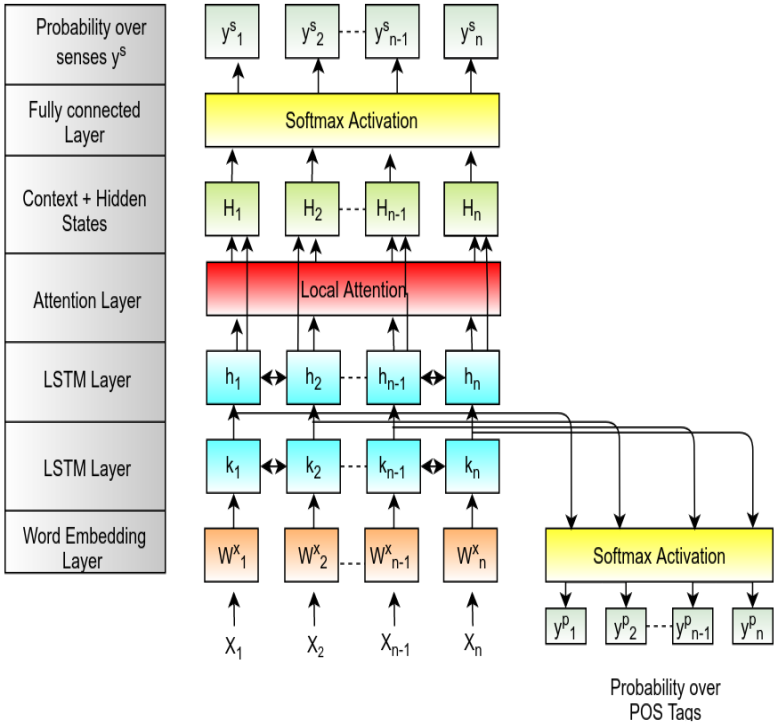

FIGURE 5.3: All-words Model+ Local Attention+Hidden States

5.4 Hierarchical Model

We also tried the hierarchical model after success in the Word specific model. Here we used two variants of masking technique after predicting the POS Tags for every word in the sentence.

- **Hierarchical Model+Soft Masking**: Here we simply multiplied the probabilities of the corresponding POS tags with the probabilities of the senses using WordNet. This model outperform the All-word Model with CNN, shown in figure 5.5
- **Hierarchical Model+Hard Masking**: Used similar masking technique to the Word specific model. Suppose a word is predicted to be noun, then all senses which are not noun are added a negative value to suppress their probabilities, thus this masking technique is hard.

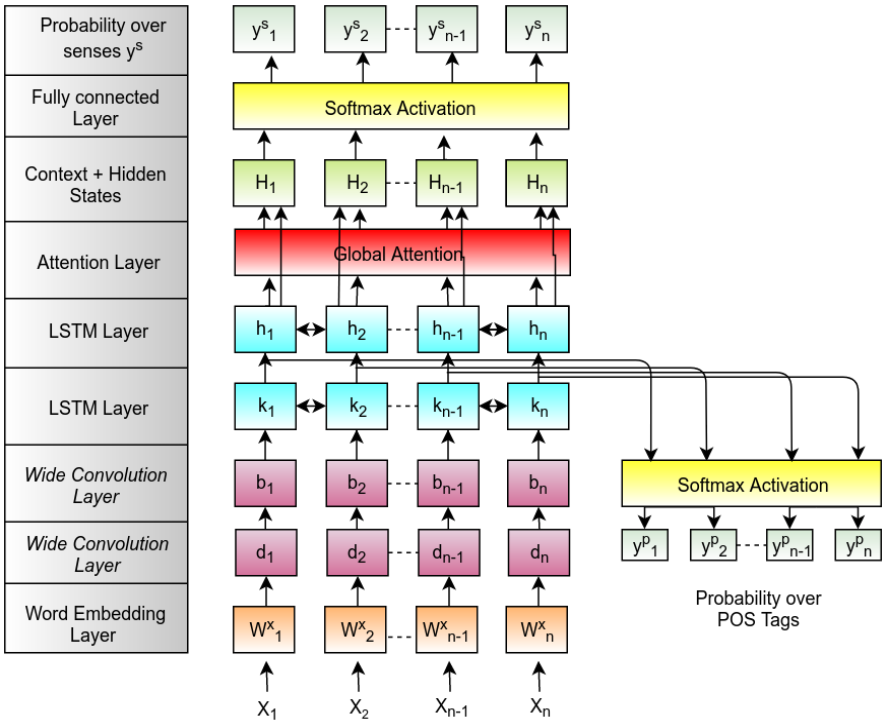

FIGURE 5.4: All-words Model+POS Tags+CNN

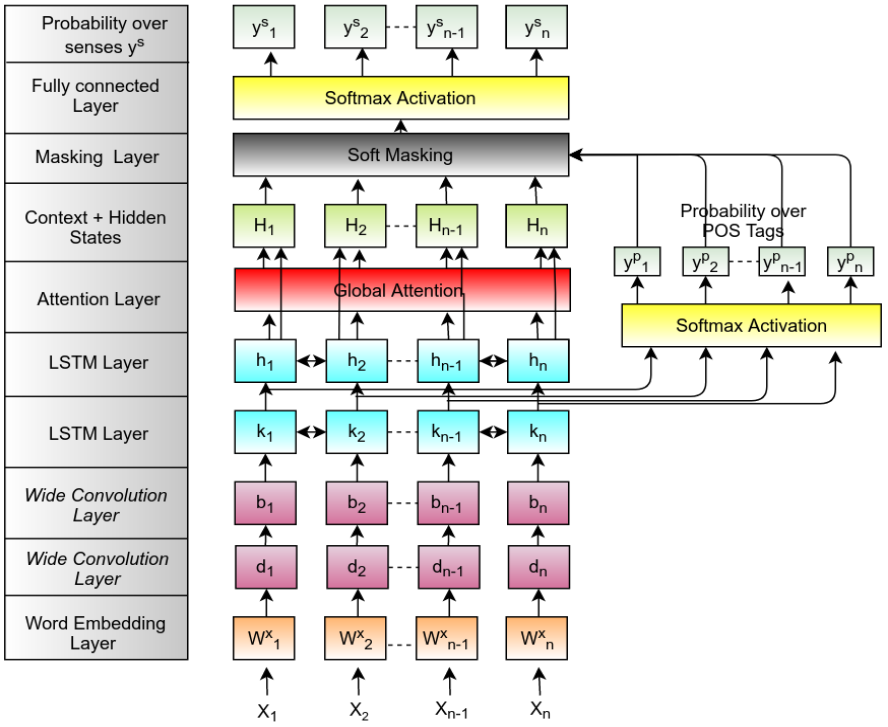

FIGURE 5.5: All-words Model+POS Tags+CNN+Soft Masking

Chapter 6

Datasets

6.1 Senseval-2: Hard, Line, Serve, Interest datasets

We use the Senseval-2 Lexical Sample Task dataset (Edmonds and Cotton [12]) for the Word specific model only which consists of 4 words- hard, serve, line and interest having 3,4,6,6 senses respectively. The data distribution across senses is shown in Table 6.1

Word	#Senses	Total # of examples	Distribution across Senses
hard	3	4333	(3455, 502, 376)
serve	4	4378	(1814, 1272, 853, 439)
interest	6	2368	(1252, 500, 361, 178, 66, 11)
line	6	4146	(2217, 429, 404, 374, 373, 349)

TABLE 6.1: Senseval-2 Four-Words Dataset (Edmonds and Cotton [12])

6.2 One Million Sense Tagged Instances Corpus

We used this dataset (Taghipour and Ng [13]) for both the models: word specific model and all-words model. For word specific model, we only trained on these words: open, force, make, point, support, serve, place. The data distribution across senses is shown in below:

Word	#Senses	# of examples	Distribution across Senses
serve	4	3421	(1941#V1, 839#V2, 529#V3, 112#V4)
place	6	3511	(1149#N1, 623#V1, 490#V2, 488#N2, 479#V3, 282#N3)
make	7	6566	(2006#V1, 1025#V2, 968#V3, 962#V4, 617#V5, 543#N6, 445#V7)
open	5	2913	(990#ADJ1, 662#V1, 632#V2, 565#V3, 64#ADJ2)
support	7	3423	(1020#V1, 670#N1, 533#V2, 503#V3, 470#V4, 170#V5, 57#N2)
force	5	3649	(1150#N1, 969#N2, 543#V1, 495#N3, 492#N4)
point	8	2766	(989#N1, 518#V1, 479#N2, 282#N3, 193#N4, 163#N5, 87#V2, 55#V3)

TABLE 6.2: One Million Dataset: SemCor+OMSTI

Chapter 7

Results

7.1 Word specific Model

Sense Word	Model	F1 Score		Accuracy	
		Train	Val	Train	Val
Force	Model-1	98.42%	91.49%	98.40%	92.01%
	Model-2	97.21%	89.74%	97.36%	90.97%
	Model-3	97.24%	90.26%	97.39%	91.49%
	Model-4	97.65%	89.33%	97.74%	90.62%
Make	Model-1	75.21%	49.33%	75.87%	51.91%
	Model-2	65.62%	50.44%	66.72%	52.34%
	Model-3	67.33%	52.59%	68.65%	54.08%
Open	Model-1	95.72%	77.30%	96.31%	82.98%
	Model-2	93.53%	77.88%	94.05%	84.03%
	Model-3	92.87%	78.35%	94.31%	84.37%
	Model-4	94.38%	76.60%	94.62%	84.55%
Place	Model-1	96.32%	83.58%	96.65%	84.89%
	Model-2	93.29%	81.68%	94.01%	83.33%
	Model-3	94.30%	83.99%	94.98%	85.07%
	Model-4	93.69%	83.24%	94.31%	84.03%
Point	Model-1	94.58%	75.63%	96.35%	83.85%
	Model-2	91.27%	73.87%	93.89%	83.59%
	Model-3	92.52%	75.60%	94.60%	83.85%
	Model-4	92.52%	75.81%	94.74%	84.63%
Serve	Model-1	90.40%	79.57%	90.96%	82.12%
Support	Model-1	90.63%	68.51%	90.25%	67.01%
	Model-2	86.47%	72.75%	85.30%	71.00%
	Model-3	87.65%	69.00%	87.05%	68.92%
	Model-4	88.80%	63.30%	88.76%	64.93%

Table 7.1: Results on One Million Dataset: SemCor+OMSTI

Target Word	F1-Score			Accuracy		
	Train	Val	Test	Train	Val	Test
Hard	89.45%	78.66%	78.11%	94.85%	89.37%	89.78
Serve	95.75%	89.80%	89.84%	96.49%	95.5%	91.94
Interest	84.32%	80.50%	72.33%	92.06%	89.06%	86.16
Line	87.98%	82.45%	78.73%	92.08%	88.75%	86.33

TABLE 7.2: Senseval-2 Four-Words Dataset (Edmonds and Cotton [12]) Results

7.2 All-words Model

Table 7.3 shows the performance of models trained on the dataset (Taghipour and Ng [13]) consisting of 680066 sentences and 45 classes of senses. These scores are evaluated on the same dataset consisting of 170016 sentences.

Model	F1-Score	Accuracy
All-word Model	65.54%	73.16%
All-word Model+PT+Local Attention	44.36%	53.75%
All-word Model+PT+Local Attention+Hidden States	52.19%	58.68%
All-word Model+PT+Gated Attention*	44.17%	53.07%
All-word Model+PT+Local Attention+Hidden States+CRF	50.65%	57.15%
All-word Model+PT+CNN	72.33%	77.93%
All-word Hierarchical Model+Soft Masking	74.04%	79.38%
All-word Hierarchical Model+Hard Masking*	70.35%	77.30%

TABLE 7.3: Results on One Million Dataset: SemCor+OMSTI, * shows that these models are early stopped

Chapter 8

Conclusion and Future Work

8.1 Conclusion

We defined, analyzed and compared experimentally different end-to-end models of varying complexities, including different variants of attention, masking mechanisms. Unlike the word specific model, where a dedicated model needs to be trained for every target word and each disambiguation target is treated in isolation, all-words model learn a single model in one pass from the training data, and then disambiguate jointly all target words within an input text. Hierarchical models outperform in both the models. Hence we can say that hierarchical models are the key for WSD task. The use of POS Tags only improved the context vector but its effect on the accuracy is negligible. Computing context vector in the window using hidden states decreased the performance of the model. Convolutional neural networks (CNN) extracts local features around a word, what actually humans do for disambiguating senses.

8.2 Future Work

As future work, we plan to extend the hierarchical model from only POS Tags to POS Tag $\rightarrow$ lexical num $\rightarrow$ lexical id using WordNet database. But we can't use this models directly for applications like Machine Translation or Summerization, only if these senses can be formed in sense vector then we can use them as word embeddings in these tasks. So we can generate sense vectors for all the words.

Bibliography

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL http://arxiv.org/abs/1310.4546.

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[3] Mikael Kågebäck and Hans Salomonsson. Word sense disambiguation using a bidirectional LSTM. *CoRR*, abs/1606.03568, 2016. URL http://arxiv.org/abs/1606.03568.

[4] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/D17-1120.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[6] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4, July 2005. doi: 10.1109/IJCNN.2005.1556215.

[7] Wordnet. https://wordnet.princeton.edu/.

[8] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41 (2):10:1–10:69, February 2009. ISSN 0360-0300. doi: 10.1145/1459352.1459355. URL http://doi.acm.org/10.1145/1459352.1459355.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[10] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014. URL http://arxiv.org/abs/1404.2188.

[12] Philip Edmonds and Scott Cotton. Senseval-2: Overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2387364.2387365.

[13] Kaveh Taghipour and Hwee Tou Ng. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 338–344. Association for Computational Linguistics, 2015. doi: 10.18653/v1/K15-1037. URL http://www.aclweb.org/anthology/K15-1037.