Word Sense Disambiguation by learning Long Term Dependencies

A thesis submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Lalchand Pandia

to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June 27, 2017

Statement of Thesis Preparation

1. Thesis title: Word Sense Disambiguation by learning Long Term dependencies
2. Degree for which the thesis is submitted: M.Tech
3. Thesis Guide was referred to for preparing the thesis.
4. Specifications regarding thesis format have been closely followed.
5. The contents of the thesis have been organized based on the guidelines.
6. The thesis has been prepared without resorting to plagiarism.
7. All sources used have been cited appropriately.
8. The thesis has not been submitted elsewhere for a degree.

Lalchand Pandia
(Signature of the student)

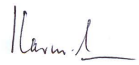
Name: Lalchand Pandia

Roll No.: 15111022

Department/IDP: CSE

CERTIFICATE

It is certified that the work contained in the thesis titled **Word Sense Disambiguation by learning Long Term Dependencies**, by **Lalchand Pandia**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Prof Harish Karnick
Department of Computer Science & Engineering
IIT Kanpur

June 27, 2017

ABSTRACT

Word Sense Disambiguation(WSD) is the task of assigning a word token in a text to a well defined word sense. For instance, the word Fox can refer to both carnivore animal and the news channel. Similarly, the word Pen can refer to child's playpen or writing instrument pen or a female swan. The inherent difficulty of sense disambiguation makes it an AI complete problem which means it can be solved by solving all the difficult problems in Artificial Intelligence(AI).

Improved Sense disambiguation can led to improvement in several natural language processing(NLP) problems such as machine translation, information retrieval and hypertext navigation, etc. Most approaches to WSD uses bag-of-words approach thereby ignoring the order of words and on language specific hand crafted features thereby making WSD difficult for resource poor languages.

With this in mind, we use sequence modeling without using explicit context window to combine local and global information to deduce the sense of a word. Our work focuses on using real valued vector representation of words and sequence modeling to achieve WSD.

To my family

Acknowledgements

I would like to thank my thesis supervisor, Prof. Harish Karnick. I thank him for providing me with the perfect balance of guidance and freedom. I would also like to thank the supportive and friendly environment in IIT Kanpur which allowed me to come in contact with machine learning and natural language processing.

I wouldn't be where I am today without the amazing support, encouragement and love from my parents and my sister.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Most words in a language are polysemous - that is have multiple meanings. For example, the word *hold* has well over a dozen meanings even when we keep our meaning space fairly coarse and do not distinguish between subtle shades of meaning.

The problem of Word Sense Disambiguation (WSD) is figuring out what a word means when it is used in a particular context. It has been a central problem in the field of computational linguistics, and in particular for Machine Translation (MT) and Information Retrieval (IR) applications. For example in MT the French word *grille* depending on context can translate to railings, gate, grid, bar, scale, schedule, etc. Similarly in IR applications when searching for the word *court* associated with judicial law it is helpful to eliminate documents containing the word *court* as associated with royalty.

For our sense inventory we use the sense definitions given in WORDNET 3.1, which is comparable to a real dictionary in its coverage and sense distinctions.

1.1 Motivation

1.1.1 Inability to preserve order of input words

The prime drawback in all the previous approaches to WSD is that the order of words surrounding the target word is largely ignored. e.g., the phrase ”Solid rock”

and ”rock solid” both convey completely different meanings.

1.1.2 Use of language specific hand crafted features

Most of the previous approaches rely on heavily hand crafted features like collocations, lemmas, etc which are difficult to achieve for poorly resourced languages.

1.2 Problem Statement

We are given unrestricted, real-world English sentences which contain the word to be disambiguated, and a set of target senses to which the word belongs.

The problem is to generate a suitable representation of the sentence and then train a multi-class classifier that can assign the correct sense to the target word present in the sentence.

The training data for learning the classifier contains text where the word in question is labelled by its appropriate sense.

sentence	Hard1	Hard2	Hard3
sentence1	1	0	0
sentence2	0	1	0
sentence3	0	0	1

Table 1.1: Training data for WSD

For example we see that in Table 1.1 we have three sentences containing the word HARD with three different senses. Hard1 refers to ”difficult”, Hard2 refers to ”dispassionate” and Hard3 refers to ”resisting weight or pressure”.

1.3 Contributions of the Thesis

In this thesis we present an approach which can be used for WSD using bidirectional LSTMs . Our approach does not require any kind of hand crafted feature engineering. We develop a supervised method based on bidirectional LSTMs. Our approach uses

the already available unlabelled data present in the public domain in the form of word embeddings to initialize the input layer of our architecture. Our approach maintains the order of words present in the sentences and disambiguates word senses by using a distributional approach to represent meaning. This is particularly appropriate for WSD where the correct sense is clearly a function of the context.

We achieved accuracies which are comparable to the best results on two datasets(Interest and Line) and even better than the prevoiusly reported results on two other datasets(Serve and Hard). We also tested our method on One Million Sense-Tagged Instances dataset and the results are encouraging. We examined several longer sentences and found that our model can capture longer dependency information required to disambigute word senses.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 describes the previous work that has addressed the WSD problem. It also describes the architecture of bidirectional LSTMs and how GloVe vectors are trained. Chapter 3 introduces our approach to WSD. Chapter 4 contains details of the experiments done to evaluate our approach. It describes the datasets used, the experimental setup and the experiments done to evaluate our performance and the results obtained. Chapter 5 contains the conclusion and possible future work.

Chapter 2

Background on Word Sense Disambiguation

A sentence S is a sequence of words $(w_1, w_2, ..., w_N)$ and we can formulate WSD (word sense disambiguation) as the task of assigning the appropriate sense(s) to all or some of the words in the sentence, i.e., to find a mapping A from words to senses, such that $A(i) \subseteq Senses_D(w_i)$, where $Senses_D(w_i)$ is the set of senses encoded in a dictionary D for word w_i, and $A(i)$ is that subset of senses of w_i which are appropriate in the context T. Usually, T consists of the sentence(s) within which the word w_i, whose senses are to be determined, is embedded.

There are two variants of WSD:

i) Lexical Sample WSD: The system is required to disambiguate a restricted set of target words usually occurring one per sentence.

ii) All-words WSD: The system is expected to disambiguate all open-class words in a text (i.e., nouns, verbs, adjectives, and adverbs).

We will be focusing on Lexical Sample WSD.

2.1 Supervised Sense Disambiguation

Supervised WSD uses machine-learning techniques for inducing a classifier from manually sense-annotated data sets. Usually, the classifier is concerned with a single word and performs a classification task in order to assign the appropriate sense to each instance of that word. The training set used to learn the classifier typically contains a set of examples in which a given target word is manually tagged with a sense from the sense inventory of a reference dictionary (often WORDNET).

2.1.1 Extracting feature vectors for Supervised WSD

Collocational features

A collocation is a word or phrase in a position-specific relationship to a target word. Collocational features encode information about specific positions located to the left or right of the target word. Typical features extracted for these context words include the word itself, the root form of the word, and the word's part-of-speech. Such features are effective at encoding local lexical and grammatical information that can often accurately isolate a given sense.

$$[word_{i-2}, POS_{i-2}, word_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+1}]$$

Bag-of-words features

A bag-of-words means an unordered set of words, ignoring their exact position. The simplest bag-of-words approach represents the context of a target word by a vector of features, each binary feature indicating whether a vocabulary word w does or doesn't occur in the context. This vocabulary is typically preselected as some useful subset of words in a training corpus.

2.1.2 Learning Algorithms

Exemplar-Based or Instance-Based Learning [1]

Exemplar-based (or instance-based, or memory-based) learning is a supervised algorithm in which the classification model is built from examples. The model retains examples in memory as points in the feature space and, as new examples are subjected to classification, they are progressively added to the model.

The distance between two examples is $\Delta(x_i, x_j) = \Sigma_{k=1}^{m} w_k d(x_{i_k}, x_{j_k})$

m=number of features for each example.

$d(v_i, v_j) = \Sigma_{k=1}^{n} |\frac{C_{i,k}}{C_i} - \frac{C_{j,k}}{C_j}|$

$C_{i,k}$ is the number of training examples with value v_i for feature f that is classified as sense k in the training corpus, and C_i is the number of training examples with value v_i for feature f in any sense. n is the total number of senses for word w.

During testing, a test example is compared against all the training examples. The sense of w for the test example is the sense of the closest marching matching training example.

Decomposable models [2]

Probabilstic classifier which approximates the joint distribution of all features by capturing most important interactions among features. This provides a principled way to combine several features.

Naive Bayes Classifier [3]

It relies on the calculation of the conditional probability of each sense s_i of a word w given the features c_j in the context.

$$\begin{aligned} \hat{s} &= argmax_{s_i \in Senses_D(w)} P(s_i | c_{-k}, ...c_k) \\ &= argmax_{s_i \in Senses_D(w)} \frac{\Pi_{j=-k}^{k} P(c_j|s_i) P(s_i)}{P(c_{-k}, ...c_k)} \end{aligned} \tag{2.1}$$

Problem: These supervised approaches make use of hand-annotated corpora.

Most of these approaches are limited by the availability of hand-annotated text. Since it is unlikely that such text will be available in large quantities for most of the polysemous words in the vocabulary, there are serious questions about how such an approach could be scaled up to handle unrestricted text. Moreover these approaches use rich feature space like Part-of-Speech(POS) of neighboring words, local collocations, syntactic relations, etc which might not be available for other languages.

2.2 Unsupervised Sense Disambiguation

Unsupervised methods have the potential to overcome the knowledge acquisition bottleneck that is, the lack of large-scale text resources manually annotated with word senses.

2.2.1 Context Clustering

Each occurrence of a target word in a corpus is represented as a context vector. The vectors are then clustered into groups, each identifying a sense of the target word.

A word w in a corpus can be represented as a vector whose jth component counts the number of times that word w_j co-occurs with w within a fixed context (a sentence or a larger context). The underlying hypothesis of this model is that the distributional profile of words implicitly expresses their semantics.

The similarity between two words v and w can be measured as follows:

$sim(v, w) = \frac{v.w}{|v||w|}$

If we put together the set of vectors for each word in the corpus, we obtain a co-occurrence matrix. As the vectors can be high dimensional latent semantic analysis (LSA) can be applied to reduce the dimensionality of the resulting multidimensional space via singular value decomposition (SVD). After the reduction, contextual similarity between two words can be measured in terms of the cosine between the corresponding vectors.

Finally, sense discrimination can be performed by grouping the context vectors of a target word using a clustering algorithm.

2.2.2 Multiple embeddings per word in vector space

Representing words by dense, real-valued vector embeddings, also commonly called ‘‘distributed representations’’, helps address the curse of dimensionality and improve generalization because words with the same semantic and syntactic features are located nearby in the vector space. But this results in conflation of all word senses in a single vector. To address this issue there have been efforts to learn multiple embeddings for a word which can represent the multiple senses.

Multi-Prototype Skip-Gram Model [4]

Given word w_I, the occurrence of one of its neighboring word w_O is described as a finite mixture model, in which each mixture corresponds to a sense of the word w_I. To be specific, suppose that word w_I has N_{w_I} senses and it appears in its h_{w_I}-th sense, i.e., $h_{w_I} \in 1, ..., N_w$ is the index of w_I's sense. Then $P(w_O|w_I)$ is expanded as:

$$P(w_O|w_I) = \Sigma_{i=1}^{N_{w_I}} P(w_O|h_{w_I} = i, w_I)P(h_{w_I} = i|w_I) \tag{2.2}$$

Let W denotes the vocabulary containing all the words. The general idea behind the Multi-Prototype Skip-Gram model is very intuitive: the surrounding words under different senses of the same word are usually different. $P(w_O|h_{w_I} = i, w_I)$ is the ith multinomial distribution and represents the probability of neighboring word be w_O given that the input word w_I occurs in its ith sense. $P(h_{w_I} = i|w_I)$ is the prior probability of word w_I falling into its ith sense. The equation 2.2 states that $P(w_O|w_I)$ is a weighted average of the probabilities of observing w_O conditioned on the appearance of w_I's every sense.

Expectation Maximization(EM) algorithm is used to learn the various embeddings for each sense for word w_I.

We focus on obtaining multiple embeddings for word w_I with N_{w_I} senses. Let $\pi_i = P(h_{w_I} = i|w_I)$.

$X = \{w_1, w_2, ..., w_M\}$ be the neighboring words of w_I in the corpus

Word w_I's multiple embedding vectors are represented by $V_w \in R^{d \times N_{w_I}}$. U refers to output vectors of all the words.

The parameter set we aim to learn is $\Theta = \{\pi_1, ...\pi_{N_{w_I}}; U; V_{w_I}\}$

Let h_m be the index of w_I's sense in the mth pair (w_m, w_I).

$\gamma_{m,i}$ is a hidden indicator variable and indicates if mth presence of word w_I is in its ith sense. The expected complete data log likelihood is

$$\begin{aligned} Q(\theta, \theta^{t-1}) &= \Sigma_{i=1}^{N_{w_I}} \Sigma_{m=1}^{M} \hat{\gamma}_{m,i}(log\pi_i + logP(w_m|h_m = i, w_I)) \\ &= \Sigma_{i=1}^{N_{w_I}} \Sigma_{m=1}^{M} \hat{\gamma}_{m,i}(log\pi_i + log\frac{exp(U_{w_O}^T V_{w_I,i})}{\Sigma_{w \in W} exp(U_w^T V_{w_I,i})}) \end{aligned} \tag{2.3}$$

In the E-step we estimate values of hidden variable $\gamma_{m,i}$ using the conditional expectation

$$\hat{\gamma}_{m,i} = P(\gamma_{m,i} = 1|X, \Theta) \tag{2.4}$$

and go to M step

In the M-step we obtain derivatives of Q w.r.t the parameters and update the embedding vectors and go back to E step.

We repeat this alternating procedure until convergence of the value of function Q. Finally we obtain V_{w_I} which contain the embedding vectors for N_{w_I} senses of w_I.

NonParametric Multi-Sense Skip-gram (MSSG) model [5]

This approach induces the senses by clustering the embeddings of the context words around each token. The vector representation of the context is the average of its context words' vectors. For every word type, clusters of its contexts is maintained and the sense of a word token is predicted as the cluster that is closest to its context representation.

Each word $w \in W$ is associated with sense vectors, context clusters and a global

vector $v_g(w)$. The number of senses for a word is unknown and is learned during training. Initially, the words do not have sense vectors and context clusters. We create the first sense vector and context cluster for each word on its first occurrence in the training data. After creating the first context cluster for a word, a new context cluster and a sense vector are created online during training when the word is observed with a context where the similarity between the vector representation of the context with every existing cluster center of the word is less than λ, where λ is a hyperparameter of the model. Consider the word w_t and let $c_t = w_{t-R_t}, ..., w_{t-1}, w_{t+1}, ..., w_{t+R_t}$ be the set of observed context words. The vector representation of the context is defined as the average of the global vector representation of the words in the context. Let $v_context(c_t) = \frac{1}{2*R_t}\Sigma_{c \in c_t} v_g(c)$ be the vector representation of the context c_t. Let $k(w_t)$ be the number of context clusters or the number of senses currently associated with word w_t. s_t, the sense of word w_t when $k(w_t) > 0$ is given by

$$s_t = \begin{cases} k(w_t) + 1, & \text{if } max_{k=1,2,...,k(w_t)}\{sim(\mu(w_t, k), v_{context}(c_t))\} < \lambda \\ k_{max}, & \text{otherwise} \end{cases}$$

where $\mu(w_t, k)$ is the cluster center of the k^{th} cluster of word w_t and $k_{max} = argmax_{k=1,2,...,k(w_t)} sim(\mu(w_t, k), v_{context}(c_t))$. The cluster center is the average of vector representations of all contexts that belong to that cluster. If $s_t = k_{max} + 1$ a new context cluster and a new sense vector are created for the word w_t.

Problem: These unsupervised approaches might not be able to discover the senses equivalent to traditional senses in a dictionary sense inventory. For this reason, their evaluation is usually more difficult: in order to assess the quality of a sense cluster we should ask humans to look at the members of each cluster and determine the nature of the relationship that they all share (e.g., via questionnaires), or employ the clusters in end-to-end applications, thus measuring the quality of the former based on the performance of the latter.

2.3 Dictionary and Thesaurus methods

2.3.1 Corpus Lesk Algorithm [6]

i)We have some sense labeled data like SemCor. ii)Take all the sentences with the relevant word sense. iii)Now add these to the Dictionary gloss(meaning) + examples for each sense, call it the "signature" of a sense. iv)Choose sense with most word overlap between context and signature.

It is also possible to add weights to each overlapping word. The weight is the inverse document frequency or IDF, a standard information-retrieval measure. Formally the IDF for a word i can be defined as

$idf_i = \log \frac{N_{doc}}{nd_i}$

N_{doc} is the total number of documents and nd_i is the number of these documents containing word i.

Problem: The dictionary definitions are too short to mention all of the collocations(words that are often found in the context of a particular sense of a polysemous word). In addition, dictionaries have much less coverage than one might have expected. With text evolving it might be possible that half of the words occurring in a new text cannot be related to a dictionary entry.Thus the dictionary-based approach is also limited by the knowledge acquisition bottleneck

Chapter 3

Proposed Model for WSD

In this chapter we describe our approach to word sense disambiguation. Our method is motivated by the kind of problem posed by the following sentence.

”Hard rock crushes heavy metal”. In this sentence it is not clear if ”rock” refers to a)stone or b)a genre of popular music originating in the 1950s. But if the preceding sentences and following sentences are given then it will be possible to comprehend the meaning of word ”rock”.

So we need a model that can not only take into account order of words but can also capture dependencies beyond the sentence boundaries. It should keep track of important words in the past and in future too.

First, we briefly introduce LSTM a type of recurrent neural network that can be used to model long term dependencies.

3.1 Background on Bi-directional Long Short Term Memory

Recurrent networks can be used for many sequence processing tasks. It can be used, in principle, to store representations of all past input events in the form of activations. The hidden states at every time step capture a lot of history.

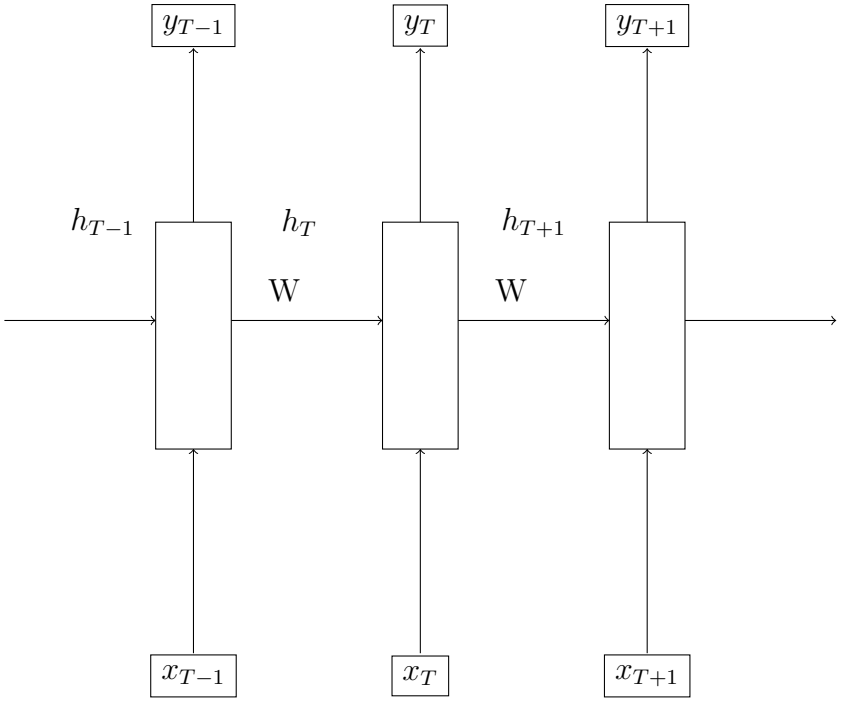


Figure 3.1: Recurrent Net Architecture

Let $x_1, x_2, ...x_T$ be a list of word vectors. x_t is a word vector input at time step t. h_t is the hidden state at time step t and y_t is the output at time step t. The hidden state acts as the memory of the network and it captures information about what happened in all the previous time steps. At a given time step :

$h_t = Wf(h_{t-1}) + W^{hx}x_t$

$y_t = softmax(Uh_t)$

RNN is trained using backpropagation through time(BPTT).

General update rule for Recurrent networks is $\frac{\partial E_t}{\partial W} = \Sigma_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \Pi_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \frac{\partial h_k}{\partial W}$

$\frac{\partial h_j}{\partial h_{j-1}} = W^T diag[f^{'}(h_{j-1})]$

$||\frac{\partial h_j}{\partial h_{j-1}}|| \leq ||W^T|| ||diag[f^{'}(h_{j-1})]|| \leq \beta_W \beta_h$

where $\beta^{'}$s are upper bounds of the norms.

$||\Pi_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}|| \leq (\beta_W \beta_h)^{t-k}$

Algorithms such as recurrent networks, based on the computation of complete gradient suffer from two major drawbacks:

i)Exploding Gradient [7]: If the product becomes very large then the error blows up and can led to oscillating weights and unstable learning.

ii)Vanishing Gradient [8]: If the product becomes very small then the error vanishes and nothing can be learned in acceptable time.

Due to these problems RNN cannot connect the gap between the relevant information and the point where it is needed if the gap between them is large.

To remedy this situation Long Short Term Memory(LSTM) was introduced.

LSTMs [9] are a special kind of RNNs and the way these calculate hidden state is different than RNNs. In LSTM hidden unit (also referred to as LSTM block) consists of:

a)Memory cell: It can keep information intact, unless inputs make it forget or overwrite with the new inputs.

b)input gate: It decides which values in memory cell should be updated.

c)forget gate: It decides which information can be thrown away from the cell.

d)Output gate: It decides what information to output.

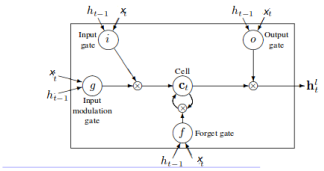


Figure 3.2: A graphical representation of LSTM memory cells **Source:** [10]

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1})$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1})$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1})$$

$$g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1})$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Here i, f, o, c are respectively the input gate, the forget gate, the output gate and the cell activation vectors all of which are the same size as the hidden state h.

For many sequence processing tasks, it is useful to consider the future as well as the past with respect to a given point in the sequence. However, conventional LSTMs are designed to analyze data in one direction only – the past.

Bidirectional LSTMs [11] are used to take advantage of both past and future data in a sequence. State at each time step consists of the state of two LSTMs, one going left and the other going right. For WSD this means that the state has information about both preceding words and succeeding words relative to the target word, which in many cases is absolutely necessary to correctly classify the sense of the word.

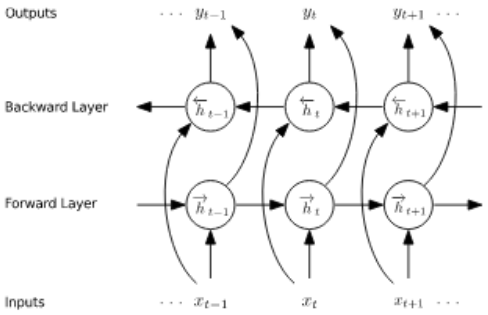


Figure 3.3: Bidirectional LSTM **Source:** [12]

3.2 Background on GloVe Vectors [13]

Word embeddings are real-valued vectors to represent words in semantically meaningful space. These embeddings have been shown to capture syntactic and semantic relations.

One of the methods for learning word embeddings which uses window based co-occurrence statistics information is GloVe.

In this approach first co-occurrence statistics for every word in the corpus is calculated. X_{ij} tabulates the number of times word j occurs in the context of word i. Here context refers to the neighboring words either to the left or right of word i. Let $X_i = \Sigma_k X_{ik}$ be the number of times any word occurs in the context of word i. $P_{ij} = \frac{X_{ij}}{X_i}$ be the probability that word j appear in the context of word i. The cost function used in this approach is $J(\theta) = \frac{1}{2}\Sigma_{i,j=1}^{V} f(X_{ij})(u_i^T v_j - logX_{ij})^2$. V is the size of the vocabulary. $f(X_{ij})$ is the weighing function.

We have used the pre-trained word embeddings made available under the Public Domain Dedication and License v1.0. These embeddings are obtained after training

on Wikipedia 2014 dump with 1.6 billion tokens + Gigaword 5 with 4.3 billion tokens.

3.3 DropOut [14]

Dropout is a regularization technique to prevent overfitting. The key idea is to randomly drop units (along with their connections) from a neural network during training. This prevents the units from co-adapting too much. Dropping units creates thinned networks during training. The number of possible thinned networks is exponential in the number of units in the network. At test time all possible thinned networks are combined using an approximate model averaging procedure. Dropout training followed by this approximate model combination significantly reduces overfitting and gives major improvements over other regularization methods. But conventional dropout does not work well with LSTMs because the recurrence amplifies noise, which in turn hurts learning. The main idea to make this work is to apply the dropout operator only to the non-recurrent connections. Standard dropout perturbs the recurrent connections, which makes it difficult for the LSTM to learn to store information for long periods of time. By not using dropout on the recurrent connections, the LSTM can benefit from dropout regularization without sacrificing its valuable memorization ability.

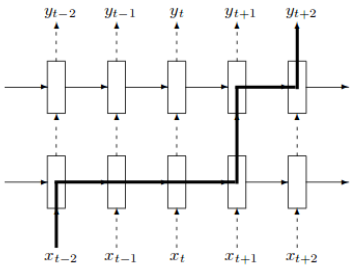


Figure 3.4: Dropout in LSTM **Source:** [10]

It basically changes the following equations:

$i_t = \sigma(W_{xi}x_t + W_{hi}D(h_{t-1}))$

$f_t = \sigma(W_{xf}x_t + W_{hf}D(h_{t-1}))$

$o_t = \sigma(W_{xo}x_t + W_{ho}D(h_{t-1}))$

$g_t = \tanh(W_{xc}x_t + W_{hc}D(h_{t-1}))$

where D is the dropout operator which sets a random subset of its arguments to zero.

3.4 Our Model

Given a sentence $(w_1, w_2, ..., w_N)$ and a target word w_T, i.e. the word to disambiguate, we wish to compute a probability distribution over the possible senses corresponding to that word. Let $|V|$ denote the size of the vocabulary.

3.4.1 Architecture of the Model

The model consists of:

i) Projection layer: The input to the projection layer is $v(w_T)$, the one-hot representation of the word w_i in sentence S. A one-hot vector representation is a vector with dimension $|V|$ consisting of $|V| - 1$ zeros and a single one which indicates the word index. It applies a W_{hx} matrix to the vector $v(w_T)$ to obtain the word embedding x_i. This will have the effect of picking the column from W corresponding to the word w_T. We have initialized W_{hx} using the pre-trained Glove embeddings.

$x_T = W_{hx}v(w_T)$

ii) Bidirectional LSTM layer: The input to the BLSTM is a word embedding x_T. This layer generates two hidden cell states h^L_{T-1} and h^R_{T+1} for word embedding x_T as an output of the right and left traversing LSTMs of the BLSTM at word w_T. The two states are concatenated to form the final BLSTM output $h_T = [h^L_{T-1}; h^R_{T+1}]$.

iii) Softmax Layer: This layer outputs the predicted distribution over senses for the target word w_T.

$s(T) = Uh_T + b$

$y(T) = softmax(s(T)) = \frac{e^{s(T)}}{\Sigma_{i \in S} e^{s(i)}}$

S denotes the total number of senses for word w_T.

The final sense of the word is predicted by taking argmax over all the probabilities

and the corresponding sense index is predicted.

3.4.2 Minimizing the Loss function

Parameters of the model are, $\Omega = \{W_{hx}, W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xc}, W_{hc}, U, b\}$ are fitted by minimizing the softmax cross entropy error with logits.

$$L(\Omega) = -\Sigma_{i \in I} \Sigma_{j \in S(w_i)} t_{i,j} log y_j(i) \tag{3.1}$$

over a set of sense labeled tokens with indices $I \subset \{1, ..., |C|\}$ within a training corpus C, each labelled with a target sense $t_i, \forall i \in I$.

For mini batch gradient training, I refers to index target word, which is to be disambiguated in the training instances present in mini batch. C refers to the training examples in the mini batch.

For our training setting we use mini-batch updates based on an average of the gradients inside each block of B examples:

$$L_t(\Omega_{t-1}) = -\frac{1}{B} \Sigma_{i=B(t-1)+1}^{B(t)} \Sigma_{j \in S(w_i)} t_{i,j} log y_j(i) \tag{3.2}$$

Here B = batch size used in mini batch gradient descent training. $S(w_i)$ refers to senses encoded in WORDNET 3.1 for word w_i

To optimize the loss function 3.2 we use Adam [15], an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

g_t denotes the vector of partial derivative of f_t w.r.t Ω at time step t. m_t denotes the biased first moment estimate. v_t denotes the biased second raw moment estimate. $\hat{m}_t$ denotes the bias-corrected first moment estimate. $\hat{v}_t$ denotes the bias-corrected second raw moment estimate. The hyper-parameters β_1 and $\beta_2 \in [0, 1)$ control the exponential decay rates of m_t and v_t. α is the step size.

The update equations for Adam are as follows:

$g_t = \nabla_\Omega L_t(\Omega_{t-1})$

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\hat{m_t} = \frac{m_t}{1-\beta_1^t}$

$\hat{v_t} = \frac{v_t}{1-\beta_2^t}$

$\Omega_t = \Omega_{t-1} - \alpha \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon}$

3.4.3 Hyperparameters of our model

Our model has several hyper parameters whose values, experience has shown, are important in so far as they have a significant impact on performance. All hyper parameters have been chosen after trying a range of values for each parameter. The best performing value has been chosen for the experiments.

1. Length of the sequence: The length of the sequence to be considered for training is crucial for our learning. The appropriate length was decided by plotting histograms of sequence lengths of datasets. The sequence length was finally choosen by running experiments on all 10 datasets for 100 as well as 200 sequence lengths.

2. Batch size: Batch size to be used in training our network is another important hyperparameter. The value was arrived at by checking performance with different batch sizes.

3. Number of epochs: The number of times we loop through the training data needs to be controlled. We would like to learn high quality representations without overfitting and thereby learning the noise in the data.

4. Hidden size of LSTM layer: The number of LSTM blocks to be used is estimated by repeating experiments on datasets and choosing the best one.

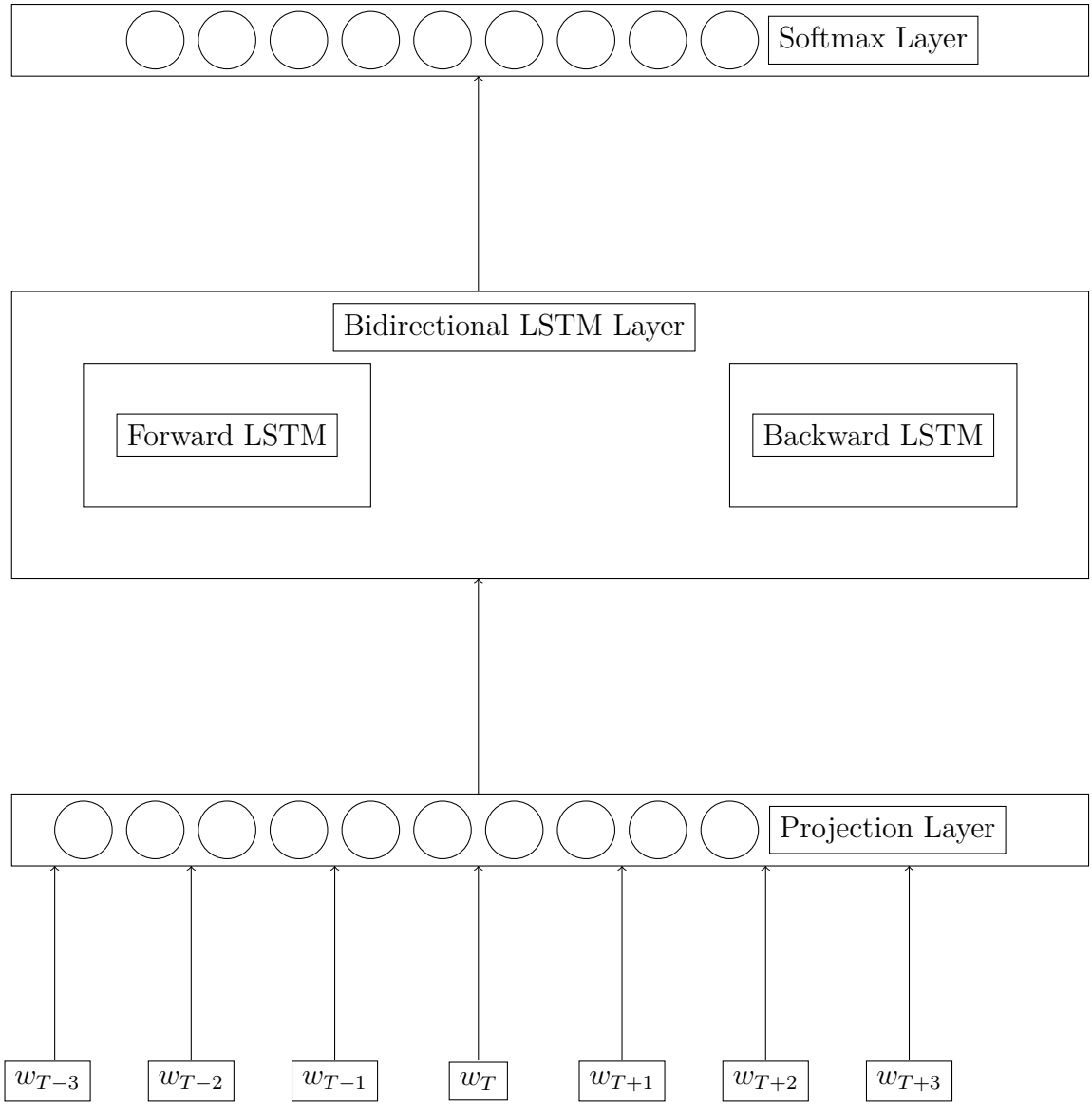


Figure 3.5: Model Architecture

Figure 3.5 gives the model architecture for our BLSTM.

Chapter 4

Performance Evaluation

In this chapter we first introduce the datasets that we use for evaluating the efficacy of our method. We then explain the details of our experimental setup and the different metrics used for performance evaluation.

4.1 Datasets

We experimented on 10 datasets used in previous literature on WSD.

4.1.1 Interest dataset

The interest corpus has been distributed from New Mexico State University contains 2,368 sentences with the nouns ”interest” or ”interests”. These sentences have been automatically extracted from the Penn Treebank Wall Street Journal corpus. Each occurrence of interest is annotated with sense definitions from Longman dictionary of Contemporary English (LDOCE).

LDOCE sense	Meaning	Frequency
interest1	readiness to give attention	361
interest2	quality of causing attention to be given	11
interest3	pastime	66
interest4	for sake of	178
interest5	a share in company, business, etc.	500
interest6	monetary charge	1252

Table 4.1: Distribution of sense tags in Interest Dataset [2]

4.1.2 Hard, Line, Serve datasets

The hard corpus has been extracted from LDC San Jose Mercury News (SJM) corpus and contains 4,333 sentences with the adjective ”hard”. The line and serve corpus has been extracted from the 1987-89 LDC Wall Street Journal corpus and from the American Printing House for the Blind corpus. Line consists of 4,146 sentences with the noun ”line” and serve consists of 4,378 sentences with the verb ”serve”. These datasets were released by Claudia Leacock, Martin Chodorow and George A. Miller

Sense	Meaning	Frequency
Hard1	difficult	3,455
Hard2	dispassionate, not soft(metaphoric)	502
Hard3	resisting weight or pressure	376

Table 4.2: Distribution of sense tags in Hard Dataset [3]

Sense	Meaning	Frequency
Line1	Product	2,217
Line2	Phone	429
Line3	Text	404
Line4	Division	374
Line5	Cord	373
Line6	Formation	349

Table 4.3: Distribution of sense tags in Line Dataset [16]

Sense	Meaning	Frequency
Serve10	Supply with food	1,814
Serve12	Hold an office	1,272
Serve2	Function as something	853
Serve6	provide a service	439

Table 4.4: Distribution of sense tags in Serve Dataset [3]

4.1.3 One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction Dataset

This dataset was released by National University of Singapore. It was extracted using a semi automated process from the MultiUN parallel corpus. WordNet sense tags were assigned to the words. Our aim was to create a balanced dataset for sense disambiguation. So, we experimented with different number of instances for each sense. Finally, we extracted only those words which have at least 3 senses and number of sentences for each sense was at least 500.

Threshold(#instances per sense)	#words	average_#senses(min_cutoff > 3)
50	191	5.47643979058
100	151	5.1059602649
200	105	4.95967741935
300	82	4.74390243902
400	67	4.68656716418
500	47	4.65957446809
550	3	4

Table 4.5: Distribution of average_#sense tags per word in One Million Instance Corpus

The consider dataset consists of 4 verb senses of ‘‘consider’’.

WordNet Sense	Meaning	Frequency
Consider%2:31:00	deem to be	600
Consider%2:32:00	take into account	527
Consider%2:39:00	give careful consideration to	551
Consider%2:32:02	think about carefully; weigh	514

Table 4.6: Distribution of sense tags in Consider Dataset [17]

The follow dataset contains 5 verb senses of word ‘‘follow’’.

WordNet Sense	Meaning	Frequency
Follow%2:38:00	to travel behind, go after, come after	565
Follow%2:38:01	travel along a certain course	520
Follow%2:41:00	act in accordance with someone’s rules, commands, or wishes	518
Follow%2:42:02	be next	512
Follow%2:40:00	choose and follow; as of theories, ideas, policies, strategies or plans	509

Table 4.7: Distribution of sense tags in Follow Dataset [17]

The apply dataset consists of 4 verb senses of the word ”apply”.

WordNet Sense	Meaning	Frequency
Apply%2:34:00	make work or employ for a particular purpose or for its inherent or natural purpose	544
Apply%2:42:00	be pertinent or relevant or applicable	529
Apply%2:32:00	ask(for something)	513
Apply%2:41:06	ensure observance of laws and rules	501

Table 4.8: Distribution of sense tags in Apply Dataset [17]

The hold dataset consists of 5 verb senses of the word ”hold”

WordNet Sense	Meaning	Frequency
Hold%2:36:00	organize or be responsible for	558
Hold%2:40:00	have or possess, either in a concrete or an abstract sense	523
Hold%2:31:01	keep in mind or convey as a conviction or view	520
Hold%2:35:03	to close within bounds, limit or hold back from movement	513
Hold%2:40:04	have rightfully; of rights, titles, and offices	509

Table 4.9: Distribution of sense tags in Hold Dataset [17]

The cause dataset consists of 2 verb senses and 2 noun senses of word ‘‘cause’’.

WordNet Sense	Meaning	Frequency
cause%2:36:00	give rise to	605
cause%2:32:00	cause to act in a specified manner	538
cause%1:11:00	events that provide the generative force that is the origin of something	513
cause%1:10:00	a justification for something existing or happening	509

Table 4.10: Distribution of sense tags in cause Dataset [17]

The ‘‘open’’ dataset consists of 5 senses out of which 3 are verbs and 2 are adjectives.

WordNet Sense	Meaning	Frequency
open_1	cause to open or to become open	564
open_2	begin or set in action, of meetings, speeches, recitals, etc	514
open_3	open to or in view of all	506
open_4	make available	504
open_5	accessible to all	503

Table 4.11: Distribution of sense tags in open Dataset [17]

4.2 Experimental Setup

In this section we describe in detail a) the data preprocessing steps b)Our Model variants c) Evaluation metrics d) Comparison Baseline e) Selecting proper sequence lengths for experiments f) Hyperparameters

4.2.1 Data Preprocessing

All numbers present in the data are replaced with a <number> tag. We remove any kind of URL which occurs in the data. We also remove all kinds of fractional numbers and markups. Multiple newlines are replaced by single newline. For conducting experiments with PoS-tagging we used the Stanford PoS Tagger [18].

4.2.2 Our Model Variants

Our model has 3 variants: i)GloVe pre-initialization: Our input word vectors are initialized using GloVe 100-dimensional word vectors in Projection layer.

ii)POSTagging: All our input words are tagged with their corresponding parts-of-speech.

iii)GloVe + POS: In this variant all words are tagged with their corresponding parts-of-speech and corresponding word vectors are initialized using GloVe 100-dimensional word vectors in Projection layer.

4.2.3 Evaluation Metrics

C_i represents the i-th class. tp_i are true positive counts for C_i. fp_i are false positive counts for C_i. tn_i are true negative counts for C_i. fn_i are false negative counts for C_i. N represents number of classes. μ and M represent micro-averaging and macro-averaging respectively.

Micro-averaging uses sum of counts to obtain cumulative tp, tn, fp, fn and then calculates a performance measure. Macro-averaging is the average of measures calculated for each class $C_1, C_2, ..., C_N$. Since micro-averaging gives equal weightage to all test instances it favors classes with more number of instances. Macro-averaging treats all classes equally.

Metric	Formula	Evaluation Focus
$AverageAccuracy$	$\frac{\Sigma_{i=1}^{N} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{N}$	The average per-class effectiveness of a classifier
$ErrorRate$	$\frac{\Sigma_{i=1}^{N} \frac{fp_i+fn_i}{tp_i+fn_i+fp_i+tn_i}}{N}$	The average per-class classification error
$Precision_{\mu}$	$\frac{\Sigma_{i=1}^{N} tp_i}{\Sigma_{i=1}^{N}(tp_i+fp_i)}$	Agreement of the data class label with those of a classifier calculated from sums of per-instance decisions.
$Recall_{\mu}$	$\frac{\Sigma_{i=1}^{N} tp_i}{\Sigma_{i=1}^{N}(tp_i+fn_i)}$	Effectiveness of a classifier in identifying class labels calculated from sums of per-instance decisions.
$Fscore_{\mu}$	$\frac{(\beta^2+1)*Precision_{\mu}Recall_{\mu}}{\beta^2*(Precision_{\mu})+Recall_{\mu}}$	Relation between positive labels and those given by a classifier based on sums of per-instance decisions.
$Precision_M$	$\frac{\Sigma_{i=1}^{N} \frac{tp_i}{tp_i+fp_i}}{N}$	An average per-class agreement of the data class labels with those of a classifier.
$Recall_M$	$\frac{\Sigma_{i=1}^{N} \frac{tp_i}{tp_i+fn_i}}{N}$	An average per-class effectiveness of a classifier to identify class labels.
$Fscore_M$	$\frac{(\beta^2+1)*Precision_M Recall_M}{\beta^2*(Precision_M)+Recall_M}$	Relation between positive labels and those given by a classifier based on a per-class average.

Table 4.12: Metrics for Multi Sense Classification

Confusion matrix is another kind of metric which is used to visualize the performance of any model. The rows represent the true labels and the columns represent the predicted labels. The diagonal line represents the instances that the model predicts correctly.

Generally β is taken to be 1. $Recall_{\mu}$ measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in micro-averaging.

4.2.4 Baseline

Our baseline performance measure is to determine the most frequently occurring sense in the training set, and to assign this sense to all the test sentences.

4.2.5 Selecting proper sequence length for experiments

We carried out our experiments on all the datasets for sequence lengths of 100 and 200 and choose a final sequence length of 200 based on performance. Table 4.13 , Table 4.14 and Table 4.15 give accuracy results for sequence lengths of 100 and 200.

Word	baseline	Sequence Length=100	Sequence Length=200
Cause	28.46	49.72	55.2
Open	21.63	49.61	57.9
Consider	26.99	47.16	56.94
Follow	22.39	51.90	58.11
Apply	26.99	47.21	54.76
Hold	22.13	65.80	74.86
Interest	52.73	83.35	84.82
Hard	79.96	85.22	86.34
Line	52.76	81.08	83.39
Serve	41.43	85.09	86.17

Table 4.13: GloVe pre-initialization results for different sequence lengths

Word	baseline	Sequence Length=100	Sequence Length=200
Cause	28.46	60.62	58.16
Open	21.63	56.25	61.43
Consider	26.99	44.60	47.85
Follow	22.39	49.67	52.62
Apply	26.99	41.26	46.25
Hold	22.13	61.52	67.83
Interest	52.73	80.02	81
Hard	79.96	84.39	84.78
Line	52.76	76.34	77.75
Serve	42.43	81.6	83.82

Table 4.14: PoS Tagging results for different sequence lengths

Word	baseline	Sequence Length=100	Sequence Length=200
Cause	28.46	58.89	71.04
Open	21.63	54.25	70.37
Consider	26.99	46.4	56.44
Follow	22.39	55.87	58.37
Apply	26.99	49.52	51.63
Hold	22.13	63.81	72.77
Interest	52.73	80.53	82.78
Hard	79.96	84.59	85.52
Line	52.76	79.92	81.2
Serve	42.43	84.27	85.74

Table 4.15: GloVe+PoS results for different sequence lengths

4.2.6 Hyperparameters

We first use the default values of hyperparameters:

i) Number of epochs = 50

ii) Size of BLSTM hidden layer = 64

iii) Initial Learning Rate = 0.01

iv) Learning rate decay = 0.97

v) Batch size (B) = 64

vi) Dropout on word embedding, hidden layer = 0.5

vii) Embedding size = 100

We found that default values for i), iii), iv), v), vi), vii) work well by measuring cross-validation error. We had to change the size of the BLSTM hidden layer to 32 as this was giving better performance.

We initialized the projection layer weights using Glove 100 dimensional vectors. We initializes the hidden layer weights using the uniform initializer $U(-r, r)$ where $r = \frac{6}{\sqrt{(fan\_in+fan\_out)}}$. Here fan_in = number of connections entering the hidden unit and fan_out = number of connections leaving the hidden unit. We initialized the output layer weights from a random normal initializer.

4.3 Results

In this section we present our model's performance in disambigauting different senses on 10 datasets. Final sequence length is 200. In all the experiments 75% of the data was used for training data and the remaining 25% was used as test data.

k-Fold Cross Validation: In k-fold cross-validation, the data is randomly partitioned into k equal sized parts. Of the k parts, a single part is retained as the test data for testing the model, and the remaining k − 1 parts are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k parts used exactly once as the test data. The k results from the folds can then be averaged to produce a single estimation. The advantage of this method is that all sentences are used for both training and testing, and each observation is used for testing exactly once.

We report 4-fold cross validated results on 10 datasets.

4.3.1 Interest dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	94.93	5.05	84.82
With PoS-Tagging	93.6	6.34	81
With GloVe + PoS	94.26	5.73	82.78

Table 4.16: Accuracy and Error Rate measures for Interest Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	63.9	61.5	62.7
With PoS Tagging	61.31	56.85	58.95
With GloVe + PoS	59.81	58.21	58.96

Table 4.17: Macro-averaging Precision, Recall, F1-score measures for Interest Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	84.82	84.82	84.82
With PoS Tagging	81	81	81
With GloVe + PoS	82.78	82.78	82.78

Table 4.18: Micro-averaging Precision, Recall, F1-score measures for Interest Dataset

Model	$Fscore_\mu$
Decomposable Models [2]	78
Exemplar-Based Approach [1]	87
Our Model (GloVe pre-initialization)	84.82

Table 4.19: F1-score measures for Interest Dataset achieved by different approaches

Table 4.19 compares our model's performance with several other models which heavily uses complicated and language specific hand crafted features. However, due to variations in experimental methods, it cannot be concluded that the differences among the most accurate method (Exemplar-Based Approach) are statistically significant. For example, in our approach four-fold cross validation is employed to assess accuracy

while in Exemplar-Based Approach 100 random experiments are conducted and in each experiment, 25% sentences of the data were randomly selected to form the test data and remaining data is used for training.

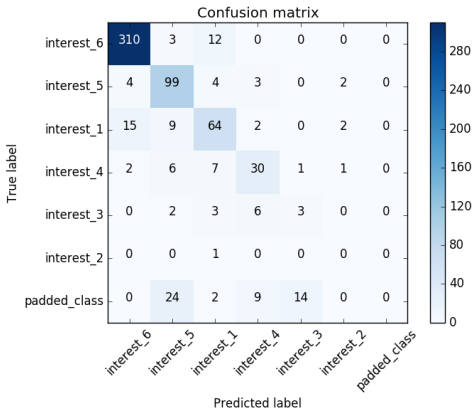


Figure 4.1: Confusion matrix for Interest with GloVe pre-initialization

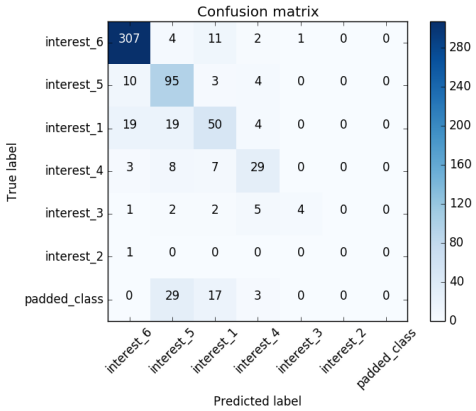


Figure 4.2: Confusion matrix for Interest with PoS tagging

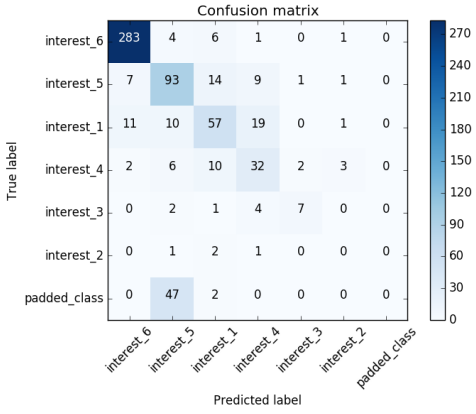


Figure 4.3: Confusion matrix for Interest with GloVe + PoS

4.3.2 Hard dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	90.88	9.11	86.34
With PoS-Tagging	89.85	10.14	84.78
With GloVe + PoS	90.35	9.64	85.52

Table 4.20: Accuracy and Error Rate measures for Hard Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	74.69	68.64	71.51
With PoS-Tagging	71.18	62.44	66.47
With GloVe + PoS	73.10	66.83 69.74	

Table 4.21: Macro-averaging Precision, Recall, F1-score measures for Hard Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	86.34	86.34	86.34
With PoS-Tagging	84.78	84.78	84.78
With GloVe + PoS	85.52	85.52	85.52

Table 4.22: Micro-averaging Precision, Recall, F1-score measures for Hard Dataset

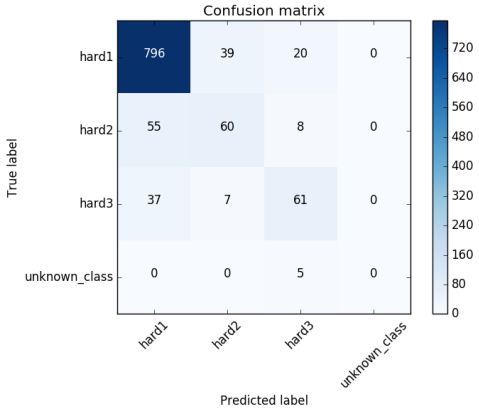


Figure 4.4: Confusion matrix for Hard with GloVe pre-initialization

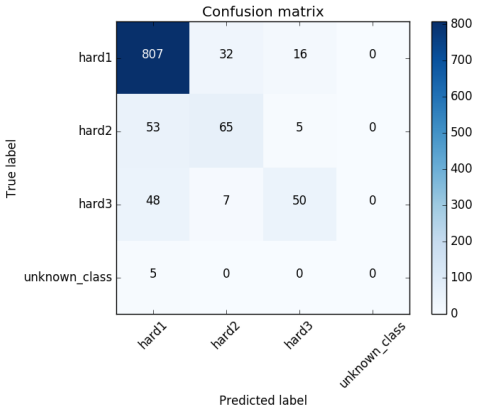


Figure 4.5: Confusion matrix for Hard with PoS Tagging

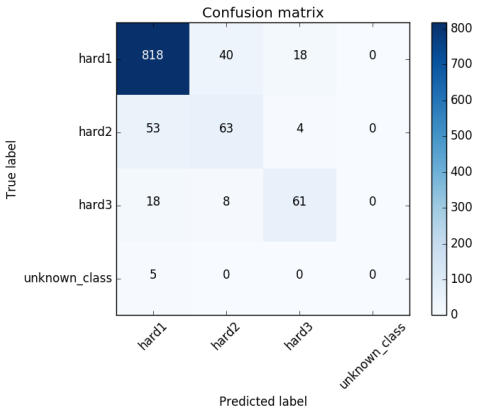


Figure 4.6: Confusion matrix for Hard with GloVe + PoS

Model	$Fscore_{\mu}$
Naive Bayes classifier with bag-of-words context feature set [3]	83
Our Model (GloVe pre-initialization)	86.82

Table 4.23: F1-score measures for Hard Dataset achieved by different approaches

Table 4.23 compares our model's performance with several other models which uses bag-of-words features

4.3.3 Serve dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	93.08	6.91	86.17
With PoS-Tagging	91.91	8.08	83.82
With GloVe + PoS	92.87	7.12	85.74

Table 4.24: Accuracy and Error Rate measures for Serve Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	82.24	82.76	82.5
With PoS-Tagging	79.75	79.63	79.69
With GloVe + PoS	82.63	81.88	82.25

Table 4.25: Macro-averaging Precision, Recall, F1-score measures for Serve Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	86.17	86.17	86.17
With PoS-Tagging	83.82	83.82	83.82
With GloVe + PoS	85.74	85.74	85.74

Table 4.26: Micro-averaging Precision, Recall, F1-score measures for Serve Dataset

Model	$Fscore_\mu$
Naive Bayes classifier with bag-of-words context feature set [3]	78
Our Model (GloVe pre-initialization)	86.17

Table 4.27: F1-score measures for Serve Dataset achieved by different approaches

Table 4.27 compares our model's performance with several other models which uses bag-of-words features.

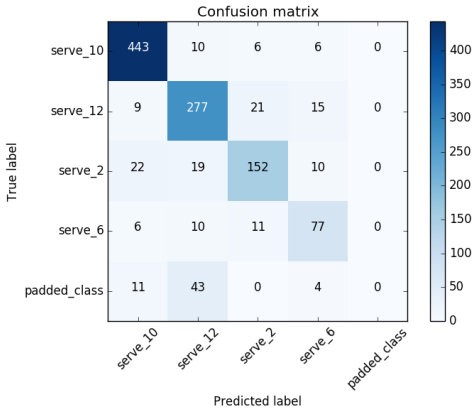


Figure 4.7: Confusion matrix for Serve with GloVe pre-initialization

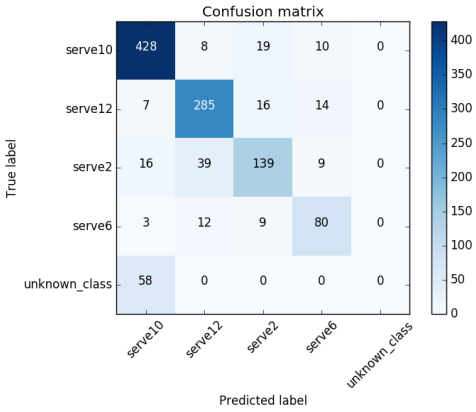


Figure 4.8: Confusion matrix for Serve with PoS Tagging

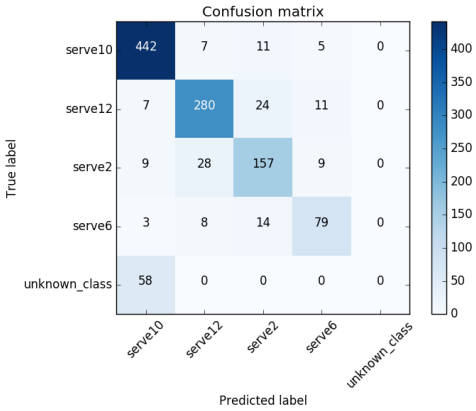


Figure 4.9: Confusion matrix for Serve with GloVe + PoS

4.3.4 Line dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	94.46	5.53	83.39
With PoS-Tagging	92.58	7.41	77.75
With GloVe + PoS	93.73	6.27	81.2

Table 4.28: Accuracy and Error Rate measures for Line Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	76.63	74.74	75.67
With PoS-Tagging	68.35	66.27	67.29
With GloVe + PoS	71.47	71.88	71.67

Table 4.29: Macro-averaging Precision, Recall, F1-score measures for Line Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	83.39	83.39	83.39
With PoS-Tagging	77.75	77.75	77.75
With GloVe + PoS	81.2	81.2	81.2

Table 4.30: Micro-averaging Precision, Recall, F1-score measures for Line Dataset

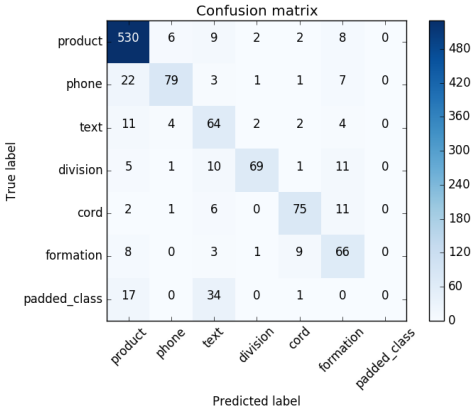


Figure 4.10: Confusion matrix for Line with GloVe pre-initialization

Model	$Fscore_{\mu}$
Naive Bayes Classifier with two separate left and right bag-of-words context feature set [3]	84
Neural network with bag-of-words feature-set [16]	76
Our Model (GloVe pre-initialization)	83.39

Table 4.31: F1-score measures for Line Dataset achieved by different approaches

Table 4.31 compares our model's performance with several other models which uses bag-of-words features.

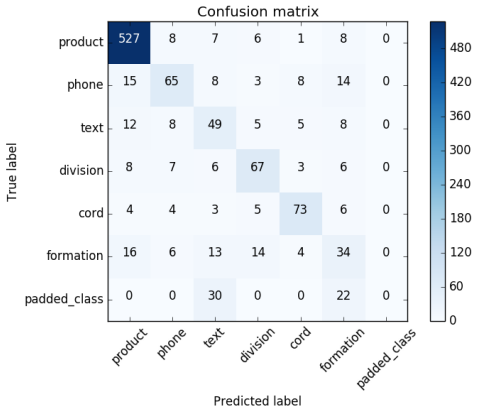


Figure 4.11: Confusion matrix for Line with PoS Tagging

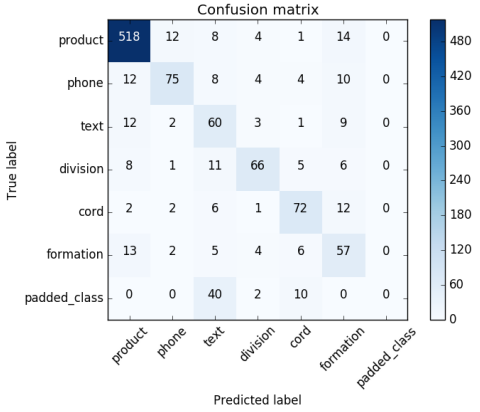


Figure 4.12: Confusion matrix for Line with GloVe + PoS

For the remaining datasets, there is no previous work to compare

our approach, so the only comparison can be made is with the baseline method.

4.3.5 Apply dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	77.38	22.61	54.76
With PoS-Tagging	73.12	26.87	46.25
With GloVe + PoS	75.83	24.16	51.63

Table 4.32: Accuracy and Error Rate measures for Apply Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	55.48	54.64	55.05
With PoS-Tagging	46.43	46.26	46.34
With GloVe + PoS	52.27	51.74	52

Table 4.33: Macro-averaging Precision, Recall, F1-score measures for Apply Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	54.76	54.76	54.76
With PoSTagging	46.25	46.25	46.25
With GloVe + PoS	51.63	51.63	51.63

Table 4.34: Micro-averaging Precision, Recall, F1-score measures for Apply Dataset

4.3.6 Consider dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	78.47	21.52	56.94
With PoS-Tagging	73.93	26.07	47.85
With GloVe + PoS	78.24	21.75	56.44

Table 4.35: Accuracy and Error Rate measures for Consider Dataset

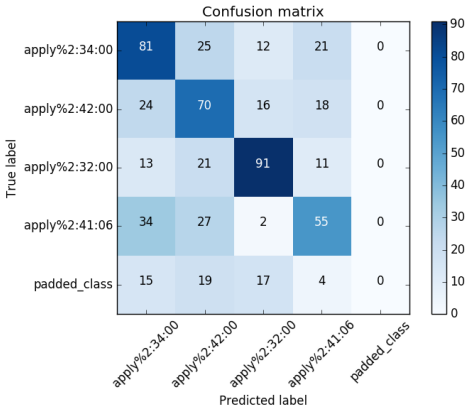


Figure 4.13: Confusion matrix for Apply with GloVe pre-initialization

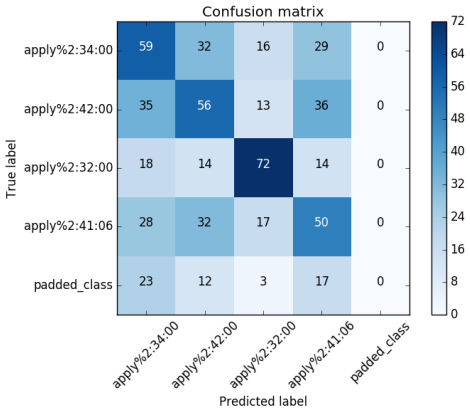


Figure 4.14: Confusion matrix for Apply with PoS Tagging

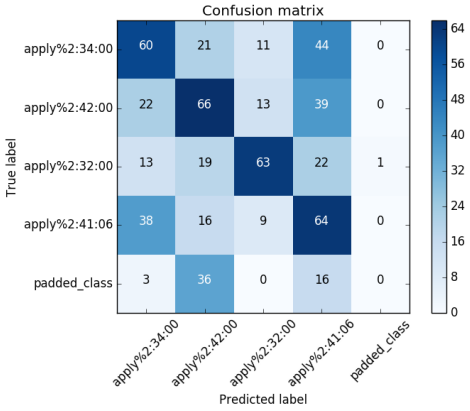


Figure 4.15: Confusion matrix for Apply with GloVe + PoS

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	58.03	56.97	57.49
With PoS-Tagging	48.3	47.99	48.14
With GloVe + PoS	56.81	56.49	56.65

Table 4.36: Macro-averaging Precision, Recall, F1-score measures for Consider Dataset

Model	$Precision_{\mu}$	$Recall_{\mu}$	$Fscore_{\mu}$
With GloVe pre-initialization	56.94	56.94	56.94
With PoS-Tagging	47.85	47.85	47.85
With GloVe + PoS	56.44	56.44	56.44

Table 4.37: Micro-averaging Precision, Recall, F1-score measures for Consider Dataset

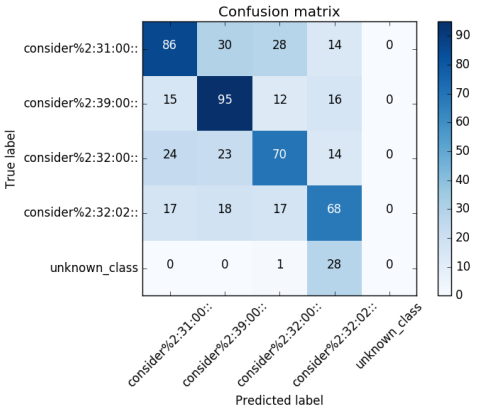


Figure 4.16: Confusion matrix for Consider with GloVe pre-initialization

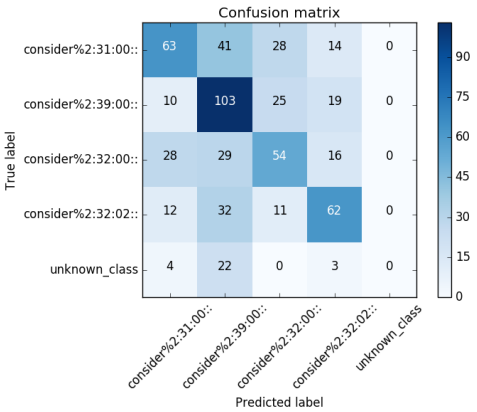


Figure 4.17: Confusion matrix for Consider with PoS Tagging

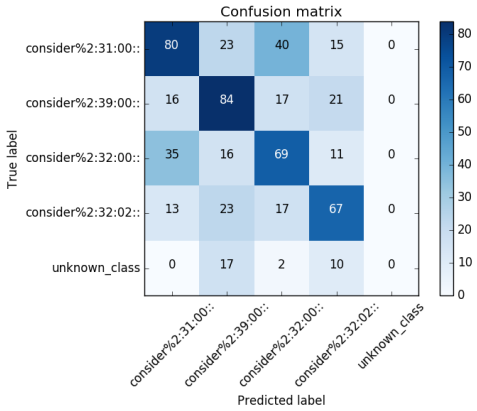


Figure 4.18: Confusion matrix for Consider with GloVe + PoS

4.3.7 Cause Dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	77.60	22.39	55.20
With PoS-Tagging	79.08	20.91	58.16
With GloVe + PoS	85.52	14.47	71.04

Table 4.38: Accuracy and Error Rate measures for Cause Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	55.80	55.93	55.86
With PoS-Tagging	59.42	58.78	59.09
With GloVe + PoS	72.11	71.43	71.77

Table 4.39: Macro-averaging Precision, Recall, F1-score measures for Cause Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	55.20	55.20	55.20
With PoS-Tagging	58.16	58.16	58.16
With GloVe + PoS	71.04	71.04	71.04

Table 4.40: Micro-averaging Precision, Recall, F1-score measures for Cause Dataset

Since ”cause” occurs in 2 senses which are of type ”noun” and 2 senses which are of type ”verb”, the variant POS-Tagging and GloVe + POS

performs better than just using GloVe pre-initialization. This shows that providing syntactic information can help disambiguate better. There are only 11 sentences where model predicted noun sense instead of verb sense and vice-versa

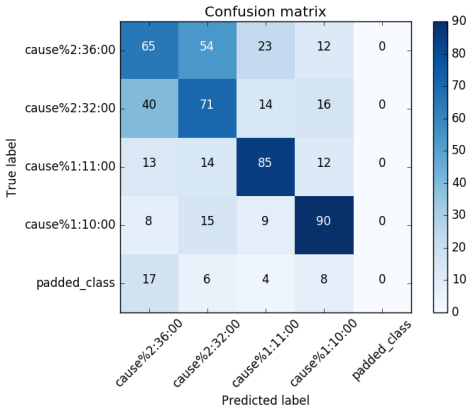


Figure 4.19: Confusion matrix for Cause with GloVe pre-initialization

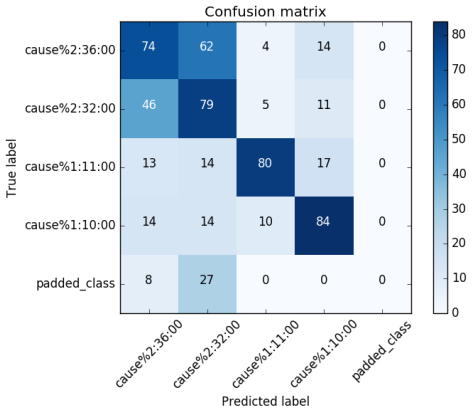


Figure 4.20: Confusion matrix for Cause with PoS Tagging

4.3.8 Open Dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	83.16	16.83	57.90
With PoS-Tagging	84.57	15.42	61.43
With GloVe + PoS	88.15	11.84	70.37

Table 4.41: Accuracy and Error Rate measures for Open Dataset

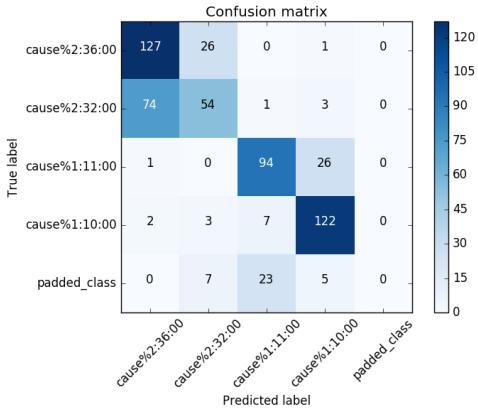


Figure 4.21: Confusion matrix for Cause with GloVe + PoS

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	57.90	58.21	58.02
With PoS-Tagging	62.08	61.49	61.79
With GloVe + PoS	71.54	71.03	71.28

Table 4.42: Macro-averaging Precision, Recall, F1-score measures for Open Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	57.90	57.90	57.90
With PoS-Tagging	61.43	61.43	61.43
With GloVe + PoS	70.37	70.37	70.37

Table 4.43: Micro-averaging Precision, Recall, F1-score measures for Open Dataset

Since ”open” occurs in 3 senses which are of type ”verb” and 2 senses which are of type ”adjectives”, the variant POS-Tagging and GloVe + POS performs better than just using GloVe pre-initialization. This shows that providing syntactic information can help disambiguate better. There are only 13 sentences where model predicted adjective sense instead of verb sense and vice-versa.

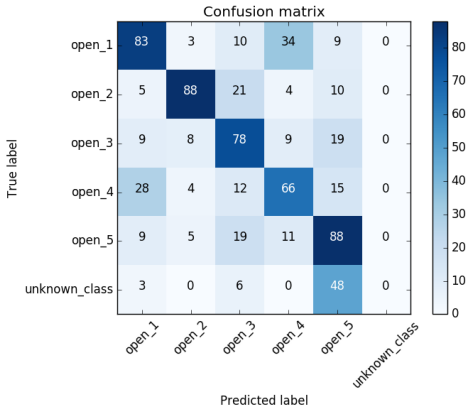


Figure 4.22: Confusion matrix for Open with GloVe pre-initialization

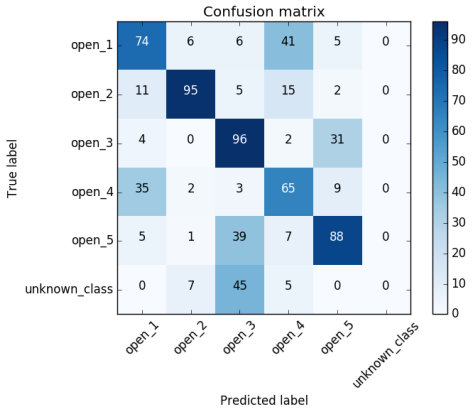


Figure 4.23: Confusion matrix for Open with PoS Tagging

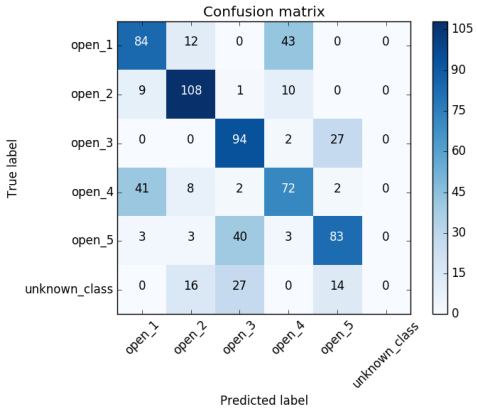


Figure 4.24: Confusion matrix for Open with GloVe + PoS

4.3.9 Hold Dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	89.94	10.05	74.86
With PoS-Tagging	87.13	12.86	67.83
With GloVe + PoS	89.10	10.89	72.77

Table 4.44: Accuracy and Error Rate measures for Hold Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	74.80	74.78	74.79
With PoS-Tagging	68.82	68.23	68.52
With GloVe + PoS	72.85	72.79	72.82

Table 4.45: Macro-averaging Precision, Recall, F1-score measures for Hold Dataset

Model	$Precision_\mu$	$Recall_\mu$	$Fscore_\mu$
With GloVe pre-initialization	74.86	74.86	74.86
With PoS-Tagging	67.83	67.83	67.83
With GloVe + PoS	72.77	72.77	72.77

Table 4.46: Micro-averaging Precision, Recall, F1-score measures for Hold Dataset

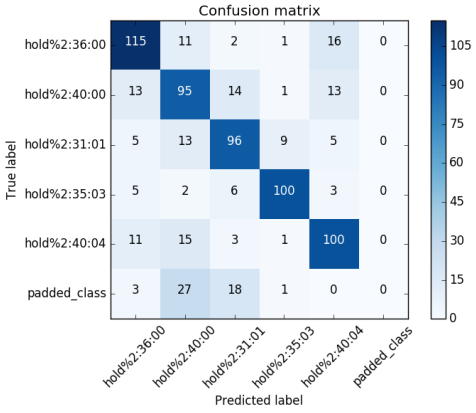


Figure 4.25: Confusion matrix for Hold with GloVe pre-initialization

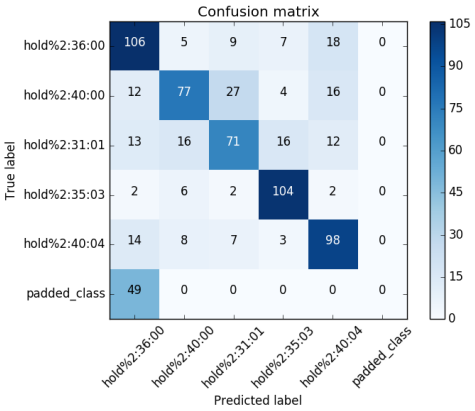


Figure 4.26: Confusion matrix for Hold with PoS Tagging

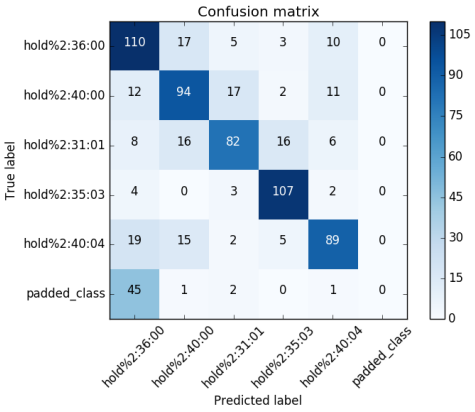


Figure 4.27: Confusion matrix for Hold with GloVe + PoS

4.3.10 Follow dataset

Model	Average Accuracy	Error Rate	Accuracy
With GloVe pre-initialization	83.24	16.75	58.11
With PoS-Tagging	81.04	18.95	52.62
With GloVe + PoS	82.96	17.03	58.37

Table 4.47: Accuracy and Error Rate measures for Follow Dataset

Model	$Precision_M$	$Recall_M$	$Fscore_M$
With GloVe pre-initialization	59.16	57.92	58.54
With PoS-Tagging	53.91	53.10	53.50
With GloVe + PoS	59.48	58.72	59.1

Table 4.48: Macro-averaging Precision, Recall, F1-score measures for Follow Dataset

Model	$Precision_{\mu}$	$Recall_{\mu}$	$Fscore_{\mu}$
With GloVe pre-initialization	58.11	58.11	58.11
With PoS-Tagging	52.62	52.62	52.62
With GloVe + PoS	58.37	58.37	58.37

Table 4.49: Micro-averaging Precision, Recall, F1-score measures for Follow Dataset

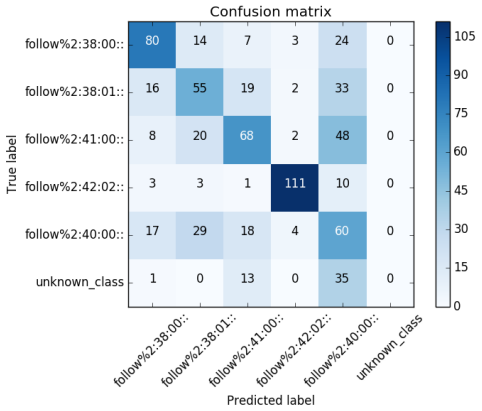


Figure 4.28: Confusion matrix for Follow with GloVe pre-initialization

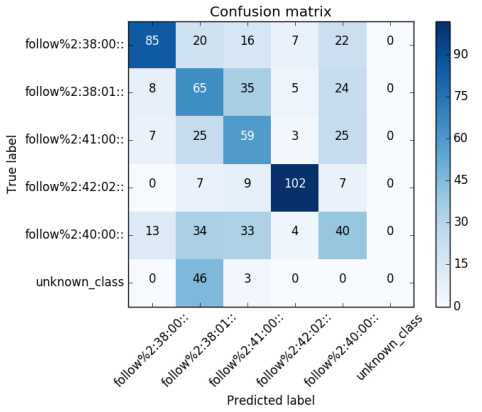


Figure 4.29: Confusion matrix for Follow with PoS Tagging

4.3.11 Observations

i) Our method showed that using Part-of-speech information helps to disambiguate if the target word occurs with more than one part-of-Speech tag as is evident from our model's performance (GloVe + POS) on the Cause dataset (3 nouns and 2 verbs) and Open dataset (3 verbs and 2 adjectives).

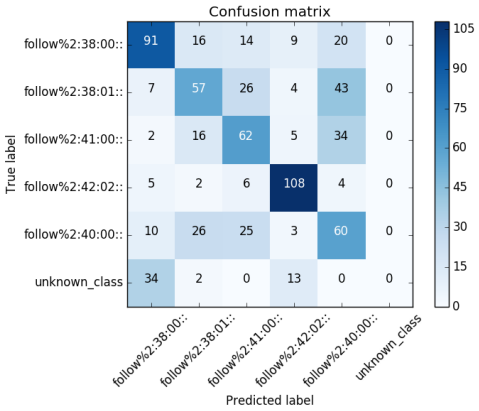


Figure 4.30: Confusion matrix for Follow with GloVe + PoS

ii) Our method is able to use longer dependencies to resolve disambiguties. For example, the following sentence was correctly classified as having correct sense of ‘‘line’’ as a cord.

Human muscle and skill had to take the place of a gun, and if the **line** *were badly thrown, if it fell short and had to be recoiled and re-thrown, five minutes would be wasted, and, with many ships to fuel, a similar mistake at each would make a total of many minutes lost, and in so many minutes ... The warrant bosun did not know that Nelson had once said that at sea five minutes can make the difference between victory and defeat, but he felt that truth in his bones after twenty years in the Navy. The weight shot off into the darkness, trailing the line behind it, and the warrant bosun uttered a grunt of morose satisfaction at feeling the line catch and hold.*

The same sentence truncated upto 100 sequence length:

Human muscle and skill had to take the place of a gun , and if the **line** *were badly thrown, if it fell short and had to be recoiled and re-thrown, five minutes would be wasted, and, with many ships to fuel, a similar mistake at each would make a total of many minutes lost, and in so many minutes ... The warrant bosun did not know that Nelson had once said that at sea five minutes can make the difference between victory and defeat, but he felt that truth in his bones.*

Since longer sentence have more words like **shot off**, the model was able to identify the right sense. It appears that very long range dependencies help in disambiguation.

Chapter 5

Conclusions and Future Work

We presented a BLSTM based model for WSD that was able to effectively exploit word order and capture dependencies spanning beyond sentence boundaries. We overcame some of the problems associated with the bag-of-words model as is evident from the performance on Hard and Serve datasets. We were also able to attain performances on Interest and Line datasets that were comparable with the methods using hand-crafted features and language specific resources.

5.1 Scope for Future work

Below we outline some of the possible future directions we would like to pursue.

5.1.1 Using multiple embeddings for a target word

We have used GloVe pre-initialization to initialize word vectors for words in the sentence. Since a single vector conflates all the senses of a word into a single vector, it misses out some of the sense-specific features which can easily distinguish one sense from another. Moreover, the huge data requirement for BLSTM gives poor result on targets with less data. To alleviate these problems, we would like to see whether initializing a word with separate embeddings depending on the word sense will help. These word embeddings are learned from large amount of untagged corpus(such as Wikipedia snapshot) using unsupervised methods. Since we have already identified

multiple senses using unsupervised method, we can use these better sense embeddings to initialize the word vectors in projection layer. This approach can utilize large amounts of unsupervised data to assist the model in case there is less supervised data.

5.1.2 Using different Recurrent networks

We have used Bidirectional-LSTM as our model which scans a sentence from left to right direction and is thus in many cases able to correctly predict sense of a target word. The model showed better performance when distinguishing between target words which occur in more than one kind of Part-of-Speech sense if sentences are pos-tagged. This indicates that giving more syntactic information can help the model learn better. We would like to use more syntactic information to correctly predict senses. We would like to use recurrent networks like Tree-Structured LSTMs [19] where each LSTM unit is able to capture information from multiple child units. It allows one to preserve information from child words which may be more helpful in capturing the sense of the target word.

5.1.3 Better Loss function

We have used the softmax entropy loss function in our experiments. Better loss functions which take account of meaning structure of a sentence like hypernym/hyponym relations in WORDNET graph which are more suitable for WSD can be formulated and these may yield better results.

References

[1] Hwee Tou Ng and Hian Beng Lee. ‘‘Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-based Approach’’. In: *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. ACL ’96. Santa Cruz, California: Association for Computational Linguistics, 1996, pp. 40–47. DOI: 10.3115/981863.981869. URL: http://dx.doi.org/10.3115/981863.981869.

[2] Rebecca Bruce and Janyce Wiebe. ‘‘Word-sense Disambiguation Using Decomposable Models’’. In: *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*. ACL ’94. Las Cruces, New Mexico: Association for Computational Linguistics, 1994, pp. 139–146. DOI: 10.3115/981732.981752. URL: http://dx.doi.org/10.3115/981732.981752.

[3] Claudia Leacock, Martin Chodorow, and George A. Miller. ‘‘Using Corpus Statistics and WordNet Relations for Sense Identification’’. In: *Computational Linguistics* 24.1 (1998), pp. 147–165. URL: citeseer.ist.psu.edu/leacock98using.html.

[4] Fei Tian et al. ‘‘A Probabilistic Model for Learning Multi-Prototype Word Embeddings’’. In: Aug. 2014. URL: https://www.microsoft.com/en-us/research/publication/a-probabilistic-model-for-learning-multi-prototype-word-embeddings/.

[5] Arvind Neelakantan et al. ‘‘Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space’’. In: *CoRR* abs/1504.06654 (2015). URL: http://arxiv.org/abs/1504.06654.

[6] Michael Lesk. ‘‘Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone’’. In: *Proceedings of the 5th Annual International Conference on Systems Documentation*. SIGDOC ’86. Toronto, Ontario, Canada: ACM, 1986, pp. 24–26. ISBN: 0-89791-224-1. DOI: 10.1145/318723.318728. URL: http://doi.acm.org/10.1145/318723.318728.

[7] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. ‘‘Understanding the exploding gradient problem’’. In: *CoRR* abs/1211.5063 (2012). URL: http://arxiv.org/abs/1211.5063.

[8] Sepp Hochreiter et al. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. 2001.

[9] Sepp Hochreiter and Jürgen Schmidhuber. ‘‘Long Short-Term Memory’’. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[10] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. ‘‘Recurrent Neural Network Regularization’’. In: *CoRR* abs/1409.2329 (2014). URL: http://arxiv.org/abs/1409.2329.

[11] Alex Graves and Jürgen Schmidhuber. ‘‘Framewise phoneme classification with bidirectional LSTM and other neural network architectures.’’ In: *Neural Networks* 18.5-6 (Nov. 10, 2005), pp. 602–610. URL: http://dblp.uni-trier.de/db/journals/nn/nn18.html#GravesS05.

[12] Yuchen Fan et al. ‘‘TTS synthesis with bidirectional LSTM based recurrent neural networks’’. In: *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*. 2014, pp. 1964–1968. URL: http://www.isca-speech.org/archive/interspeech_2014/i14_1964.html.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. ‘‘GloVe: Global Vectors for Word Representation’’. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[14] Nitish Srivastava et al. ‘‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’’. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[15] Diederik P. Kingma and Jimmy Ba. ‘‘Adam: A Method for Stochastic Optimization’’. In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[16] Claudia Leacock, Geoffrey Towell, and Ellen Voorhees. ‘‘Corpus-based Statistical Sense Resolution’’. In: *Proceedings of the Workshop on Human Language Technology*. HLT ’93. Princeton, New Jersey: Association for Computational Linguistics, 1993, pp. 260–265. ISBN: 1-55860-324-7. DOI: 10.3115/1075671.1075730. URL: http://dx.doi.org/10.3115/1075671.1075730.

[17] Kaveh Taghipour and Hwee Tou Ng. ‘‘One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction’’. In: *CoNLL*. 2015.

[18] Kristina Toutanova et al. ‘‘Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network’’. In: *In Proceedings of HLT-NAACL 2003*. 2003, pp. 252–259.

[19] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. ‘‘Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks’’. In: *CoRR* abs/1503.00075 (2015). URL: http://arxiv.org/abs/1503.00075.

[20] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[21] Roberto Navigli. ‘‘Word Sense Disambiguation: A Survey’’. In: *ACM Comput. Surv.* 41.2 (Feb. 2009), 10:1–10:69. ISSN: 0360-0300. DOI: 10.1145/1459352.1459355. URL: http://doi.acm.org/10.1145/1459352.1459355.

[22] Yoshua Bengio. ‘‘Practical recommendations for gradient-based training of deep architectures’’. In: *CoRR* abs/1206.5533 (2012). URL: http://arxiv.org/abs/1206.5533.

[23] Ted Pedersen, Rebecca Bruce, and Janyce Wiebe. ‘‘Sequential Model Selection for Word Sense Disambiguation’’. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*. ANLC ’97. Washington, DC: Association for Computational Linguistics, 1997, pp. 388–395. DOI: 10.3115/974557.974613. URL: http://dx.doi.org/10.3115/974557.974613.

[24] Rada Mihalcea, Paul Tarau, and Elizabeth Figa. ‘‘PageRank on Semantic Networks, with Application to Word Sense Disambiguation’’. In: *Proceedings of the 20th International Conference on Computational Linguistics*. COLING ’04. Geneva, Switzerland: Association for Computational Linguistics, 2004. DOI: 10.3115/1220355.1220517. URL: https://doi.org/10.3115/1220355.1220517.