# An Analysis of Word Sense Disambiguation in Bangla and English Using Supervised Learning and a Deep Neural Network Classifier


Department of Computer Science and Engineering

BRAC University

Supervisor: Mr. Moin Mostakim

Maroof Ur Rahman Pasha – 12301004

BRAC University

August 2016

# An Analysis of Word Sense Disambiguation in Bangla and English Using Supervised Learning and a Deep Neural Network Classifier

A Thesis submitted to the School of Computer Science and Engineering, BRAC University, Bangladesh.

In partial fulfillment of the requirements for the Bachelor's degree in Computer Science and Engineering

Signature of Author

_____

Maroof Ur Rahman Pasha, 12301004

Signature of Supervisor

_____

Moin Mostakim

Department of Computer Science and Engineering

BRAC University.

# DEDICATION

I would like to dedicate this thesis to my parents and to my grandparents. Their support, love and care motivates and inspires me to work harder. I would also like to dedicate this thesis to the martyred intellectuals and freedom fighters of Bangladesh.

This thesis is also dedicated to those that are struggling with speech disorders, visual disabilities and hearing impairments.

# ACKNOWLEDGEMENT

This thesis could not be completed without the wish and blessings of the almighty Allah, the most Exalted. I would not have been able to complete this thesis if it were not for my supervisor, Mr. Moin Mostakim. My parents have immensely supported me during the writing of this thesis. They have guided me, encouraged me and informed me in every step of the path to completing my thesis.

I would like to express my most sincere gratitude to my supervisor, Mr. Moin Mostakim. He has supported and guided me with great diligence and patience.

Additionally, I would like acknowledge all my friends who encouraged and supported me to complete my thesis.

**Table of Contents**

## List of Figures

## Abstract

Word sense disambiguation is a significant task in natural language processing and addresses an old problem in the field of computational linguistics. Word sense disambiguation facilitates tasks like machine translation, information retrieval, text-to-speech and other application systems. Sense ambiguity is introduced because certain words for a given language can have multiple meanings. Word sense disambiguation involves identifying the correct sense of a word for a given context. Bangla language has a few cases of word sense ambiguity. Many machine learning algorithms have already been applied to disambiguate word sense including few implementations of neural networks. However, many existing word sense disambiguation systems using supervised learning and neural networks were focused only on English data. And an evaluation of their implementation on Bangla data is necessary. This thesis attempts to analyze and present results when using an updated deep neural network classifier and supervised learning algorithm for word sense disambiguation on a Bangla dataset and on an English dataset. A pre-processor for the dataset was constructed to appropriately extract features from the context sentences to build sets of refined feature vectors which are fed into the neural network.

# CHAPTER 1

# Introduction

## 1.1 Overview

Word Sense Disambiguation for most systems involves choosing or classifying the correct sense for a given word in a sentence. Existence of homonyms in languages introduces ambiguity for word senses. Since certain words can have multiple meanings depending on the context of the sentence, the syntactic structure of that given word does not provide sufficient clues to the word's correct sense. Therefore, the context words that occur before and after the target word has to be taken into account for accurate sense classification. There are many existing word sense disambiguation algorithms and systems. Word sense disambiguation has many applications and is an important task that helps and facilitates other natural language processing tasks like machine translation, question-answering, information retrieval and text-to-speech systems.

Machine learning techniques have been applied on tackling word sense disambiguation exercises before. Supervised learning algorithms have shown to be very effective at disambiguating word senses when it comes to the lexical sample task [1]. Most supervised machine learning algorithms for lexical sample tasks learn from the training corpora, by taking in the features and labels to build an appropriate classifier. This classifier can then be evaluated by using a test data set. The classifier is able to label the senses after fed with test features. The tagged senses are then evaluated for accuracy. Deep learning and artificial neural networks have shown to be effective at dealing with complex problems and tasks. Neural(s). Therefore, deep neural

network classifier for disambiguating word sense is an approach that is justifiable. There are existing models of word sense disambiguation based on neural networks. An analysis of the sense tagging task with an updated neural network on Bangla language as well English data will provide further insights into how effective they are on natural language processing.

## 1.2 Objective and Goals

Neural networks have many applications and have been applied to word sense disambiguation. However, there is scope for further application and a revision of newer neural network models. This thesis attempts to dive deeper into the performance and efficiency of deep neural networks when used for disambiguating word senses on a standard SENSEVAL-2 English dataset and also on a small hand labelled Bangla corpus.

### 1.3 Motivation

Natural language processing in Bangla language has advanced and evolved in recent years. However, there is room for improvement and further evaluation of natural language processing systems for Bangla is important to strive towards perfection and faster systems. Neural networks and supervised machine learning houses strong models to use for word sense disambiguation of Bangla data. Motivation to work for Bangla language, neural networks and machine learning was always present. This thesis paper is the manifestation and combination of that motivation and desire to improve the computational linguistic field across all languages.

# CHAPTER 2

## Literature Review

Word sense disambiguation systems have been developed using many supervised learning techniques and shown to be effective however finding a large corpora to train the systems is still difficult [1]. They have been constructed with LSTM neural networks and show promise in classification accuracy [2]. Only few word sense disambiguation has been done on Bangla using Bangla word net. A word sense disambiguation (WSD) system for Bengali language has been developed by Ayan Das and Sudeshna Sarkar; and applied the system to get correct lexical choice for the task in Bengali-Hindi machine translation [3]. Many systems have been developed with different results and models which permits an in-depth comparative analysis and review of the systems.

## 2.1 Word Sense Disambiguation

Most supervised machine learning algorithms for lexical sample tasks learn from the training corpora, by taking in the features and labels to build an appropriate classifier. This classifier can then be evaluated by using a test data set. The classifier is able to label the senses after fed with test features. The tagged senses are then evaluated for accuracy. Word sense Disambiguation algorithms, usually, provide as output the context word selected from the sense inventory. The structure of the input and the inventory of context senses is dependent on the application: for machine translations from English to Bangla, for example, the sense inventory might be the set of Bangla translations for a given English word; for speech synthesis, the inventory might be homographs pronounced differently, such as *Bass* or *Bow* [4].

There are two forms of the word sense disambiguation tasks: the lexical sample task and the all-words task. A set of target labels is chosen, along with a set of senses for each word from some lexical corpora. Supervised machine learning approaches are most used to handle lexical sample tasks because the set of words and the set of senses are not very large. Corpus instances can be selected and hand-labelled with the correct senses for each lexical word. Classifier systems and learning algorithms can then be trained with these tagged data. Early work in word sense disambiguation mainly focused on lexical sample tasks, building word-specific systems and algorithms for disambiguating single words [4].

On the other hand, the all-words task involves systems being given entire texts and a vocabulary with a set of senses for each example and are required to disambiguate every word in the given text. As a result, there is an occurrence of serious data sparseness problem due to the

large data sample and entries; there is likely to be adequate training data for every word in the test data set. Additionally, due to the number of polysemous words in large dictionaries and lexicons, supervised learning on training data is not always practical for all-words task [4].

Work has been with unsupervised techniques for word sense disambiguation. Yarowsky, in his paper, *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods*, presents an unsupervised algorithm that rivals supervised techniques; the algorithm follows two constraints, one sense per discourse and one sense per collocation [5].

In Yarowsky's work, the one-sense-per-discourse hypothesis and model was tested on 37,232 example sense. The applicability and accuracy of the test words were recorded and showed an average of 99.8% accuracy and 50.1% applicability. Therefore, Yarowsky claimed that the hypothesis does hold confidence to be exploited and used for sense tagging tasks [5].

Yarowsky presents that their algorithm works by harnessing effective and powerful properties and modelling a rich diversity of collocation relationship [5].

Yoong Keok Lee and Hwee Tou Ng, in their work present an evaluation of knowledge sources and look at corpus-based machine learning algorithms on word sense disambiguation on the SENSEVAL-1 and SENSEVAL-2 dataset [6].

Therefore, it is safe to present that other learning algorithms, unsupervised methods and knowledge based techniques have significant and justified solutions to word sense disambiguation just like Neural network and supervised learning models do. It is important to acknowledge that the work presented using these different models are also effective and show promise in practicality.

## 2.2 Neural Networks

Word Sense Disambiguation with Neural Language Models have been presented in previous works. Yuan, Doherty, Richardson, Evans and Altendorf present, in their paper, different WSD algorithms to achieve state-of-the-art precision using neural network language models. The methods and models uses a set of word senses, a text corpus and a few example instances for each sense [7]. The classification for neural network model for their paper, is based on cosine similarity of vectors from labeled example instances and unlabeled text data for the senses. Using the WordNet and the New Oxford American Dictionary as sense inventory, there was significant performance results. They claim that the LSTM language model produced the best results [7]. Long-short term memory is a type of recurrent neural network model. The paper discussed the use of semi-supervised methods since obtaining large data is not easy. Vector representation of words and context was built using a neural network based language model (NNLM) and two NNLM were discussed [7].

The CBOW computes a context vector from word contexts and then predicts a target sense while the LSTM is trained to predict held out words from a sentence [7].

Kågebäck and Salomonsson present a bidirectional long short-term memory network model to disambiguate word sense in their paper [2]. The bidirectional version of the LSTM takes into account two LSTMs in each state, one for words before and one for word after. The model consists of a softmax layer, hidden layer and a bi-directional LSTM. The bidirectional LSTM and hidden layer for all sense and type, share the parameters, while the softmax is parameterized by word type which decides the bias vector and weight matrix for each word type [2].

The proposed BLSTM model by Kågebäck and Salomonsson performed better than previous state-of-the-art score for the Senseval-2 dataset [2].

Work has also been done for word sense disambiguation using neural networks with refined concept co-occurrence information, which is automated and does not require hand coding large data, as presented by Chung, Kang, Moon, and Lee in their paper [8].

As described before, many word sense disambiguation is done using neural network with varying precision and efficiency. They are powerful models to use and their effectiveness on Bangla language processing will provide further insight of their practical usefulness on different languages.

## 2.3 Supervised Machine Learning

Lexical sample word sense disambiguation task is commonly done using supervised learning algorithms to detect the correct sense. Firstly, the classifier has to be built using the training dataset. Important and relevant features from the dataset needs to be extracted. Which features are chosen and how they are refined tend to differ based on different models and systems. Therefore, some pre-processing of the raw text data instances has to be done. A feature vector is built encoded as numerical values for corresponding lexical data. Collocational and Bag-of-words features are two types of feature classes. Collocational features hold information of the specific positions to the left and right of the target word [4].

The collocational feature vector used for supervised word sense disambiguation, as described in [4], with a two word window to the left and to the right of the target word follows the following structure:

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

This feature vector is fed into the supervised learning model with the corresponding labelled sense tag as a unit of training data. The algorithm or model then decides the correct sense when fed with a test data of context sentences. The output is then compared with the test sense label to determine the accuracy score of the model. Many models and algorithms have been implemented, including decision lists, decision trees, Naïve Bayes, Exemplar based learning, support vector machines, Ensemble methods and many semi-supervised methods [1].

# CHAPTER 3

## Experimental Setup

For initial experiment evaluation the following tool kits and libraries were used: CUDA 7.5 toolkit, NLTK, Tensorflow, scikit-learn and Anaconda.

The experiments were setup on a personal computer with the following hardware and system configuration:

Operating System: Linux Ubuntu 16.04

CPU: Intel Core i5 650 @3.20 GHz

RAM: 4GB

GPU: Nvidia GT740

## 3.1 CUDA Toolkit:

Presented as new features in [9] for CUDA toolkit 7.5, compared to previous iterations of CUDA toolkit version, the following were introduced:

16-bit floating point (FP16) data format

- Store up to 2x larger datasets in GPU memory
- Reduce memory bandwidth requirements by up to 2x
- New mixed precision cublasSgemmEX() routine supports 2x larger matrices

New cuSPARSE GEMVI routines

- Optimized dense matrix x sparse vector routines - ideal for Natural Language Processing

Instruction-level profiling helps pinpoint performance bottlenecks

- Quickly identify the specific lines of source code limiting the performance of GPU code
- Apply advanced performance optimizations more easily

NVIDIA GPU accelerated CUDA compute platform provides acceleration across many different domains and fields including Bioinformatics, Computational chemistry, computational fluid dynamics, computational structural mechanics, Data science, Defense, Electronic Design automation, Computational finance and many more [10].

Deep learning and neural networks are relatively new software models where billions of software-neurons and connections are built and trained, in parallel instead of sequentially. Running deep neural network algorithms and learning from examples, the computer is essentially writing its own software and GPU's are ideal for such parallel computation [11].

Therefore, since the word sense disambiguation lexical sample task is done using neural network, the CUDA toolkit greatly accelerated the training speed of the dataset and allowed a faster query and test response. GPU accelerated process greatly improved the efficiency of the system.

## 3.2 Tensorflow:

As stated in the official website and documentation, "TensorFlow is an open source software library for numerical computation. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API" [12].

TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well [12].

TensorFlow provides the necessary library while keeping many complex implementations hidden, and thus does not require re-writing code by hand. Tensorflow was used to allow faster deployment of the system and to evaluate the complex model used for word sense disambiguation.

## 3.3 Scikit-learn:

Scikit-learn is an open source package of efficient tools for data mining, machine learning and data analysis. It is built on NumPy, SciPy, and matplotlib [13]. Scikit-learn python module made it easy to implement the machine learning techniques and to easily evaluate results greatly facilitating the initial experiments which was coded using the python programming language. NLTK for python was used to import and read the SENSEVAL -2 dataset.

### 3.4 Dataset:

For English, the SENSEVAL - 2 Lexical Sample dataset were used for initial experiments. For Bangla, a small hand labelled set of lexical sample examples were used for the initial experiment evaluation of the word sense disambiguation models.

### 3.5 Algorithm and Model:

**Nearest Neighbors classification:**

The nearest neighbors classification model is described on the scikit-learn website:

"Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point." [13] [14].

In general, a larger valued k suppresses noise, but makes the classification boundaries less distinct. The best choice of the value k is dependent on the data used. [14].

"The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the weights keyword. The default value, *weights* = 'uniform', assigns uniform weights to each neighbor. *weights* = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied which is used to compute the weights" [13], [14].

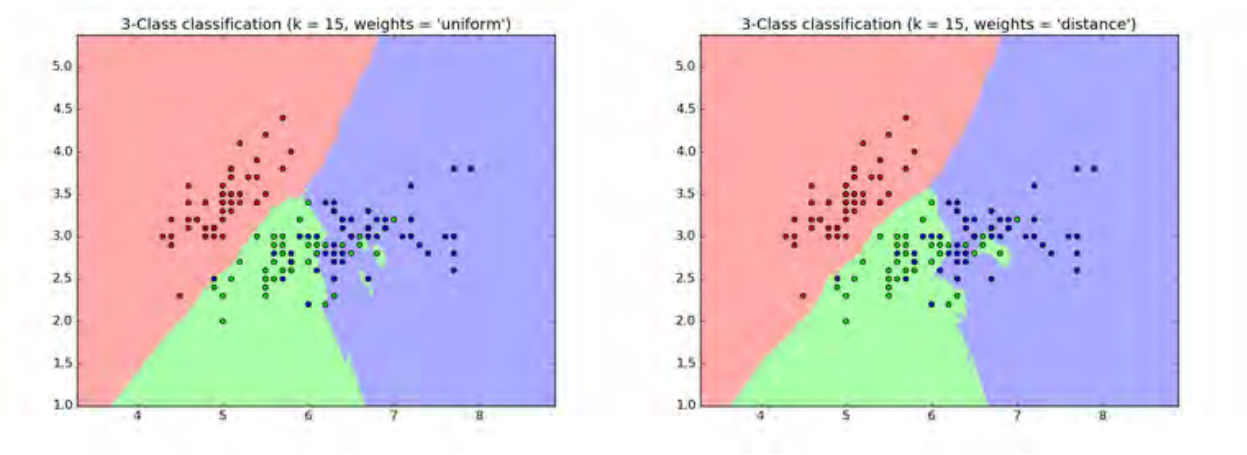



**FIGURE 1 CLASS CLASSIFICATION WITH NEAREST NEIGHBORS**

As described in the scikit-learn website, the optimal choice of algorithm, given a dataset, is complicated, and depends on a number of factors [14]:

- Number of samples N (i.e. n_samples) and dimensionality D (i.e. n_features).
    - Brute force query time grows as O[D N]
    - Ball tree query time grows as approximately O[D \log(N)]

    - KD tree query time changes with D in a way that is difficult to precisely characterize. For small D (less than 20 or so) the cost is approximately O[D\log(N)], and the KD tree query can be very efficient. For larger D, the cost increases to nearly O[DN], and the overhead due to the tree structure can lead to queries which are slower than brute force. [14]

- Data structure: intrinsic dimensionality of the data and/or sparsity of the data. Intrinsic dimensionality refers to the dimension d \le D of a manifold on which the data lies, which can be linearly or non-linearly embedded in the parameter space. Sparsity refers to the degree to which the data fills the parameter space (this is to be distinguished from the concept as used in "sparse" matrices. The data matrix may have no zero entries, but the structure can still be "sparse" in this sense).
    - Brute force query time is unchanged by data structure.
    - Ball tree and KD tree query times can be greatly influenced by data structure. [14]

- Number of neighbors k requested for a query point.
    - Brute force query time is largely unaffected by the value of k
    - Ball tree and KD tree query time will become slower as k increases. This is due to two effects: first, a larger k leads to the necessity to search a larger portion of the parameter space. Second, using k > 1 requires internal queueing of results as the tree is traversed.
- Number of query points. Both the ball tree and the KD Tree require a construction phase. The cost of this construction becomes negligible when amortized over many queries. If only a small number of queries will be performed, however, the construction can make up

a significant fraction of the total cost. If very few query points will be required, brute force is better than a tree-based method. [14]

“Currently, algorithm = 'auto' selects 'kd_tree' if k < N/2 and the 'effective_metric_' is in the 'VALID_METRICS' list of 'kd_tree'. It selects 'ball_tree' if k < N/2 and the 'effective_metric_' is not in the 'VALID_METRICS' list of 'kd_tree'. It selects 'brute' if k >= N/2. This choice is based on the assumption that the number of query points is at least the same order as the number of training points, and that leaf_size is close to its default value of 30” as stated in [13], [14].

**Neural Network Classifier:**

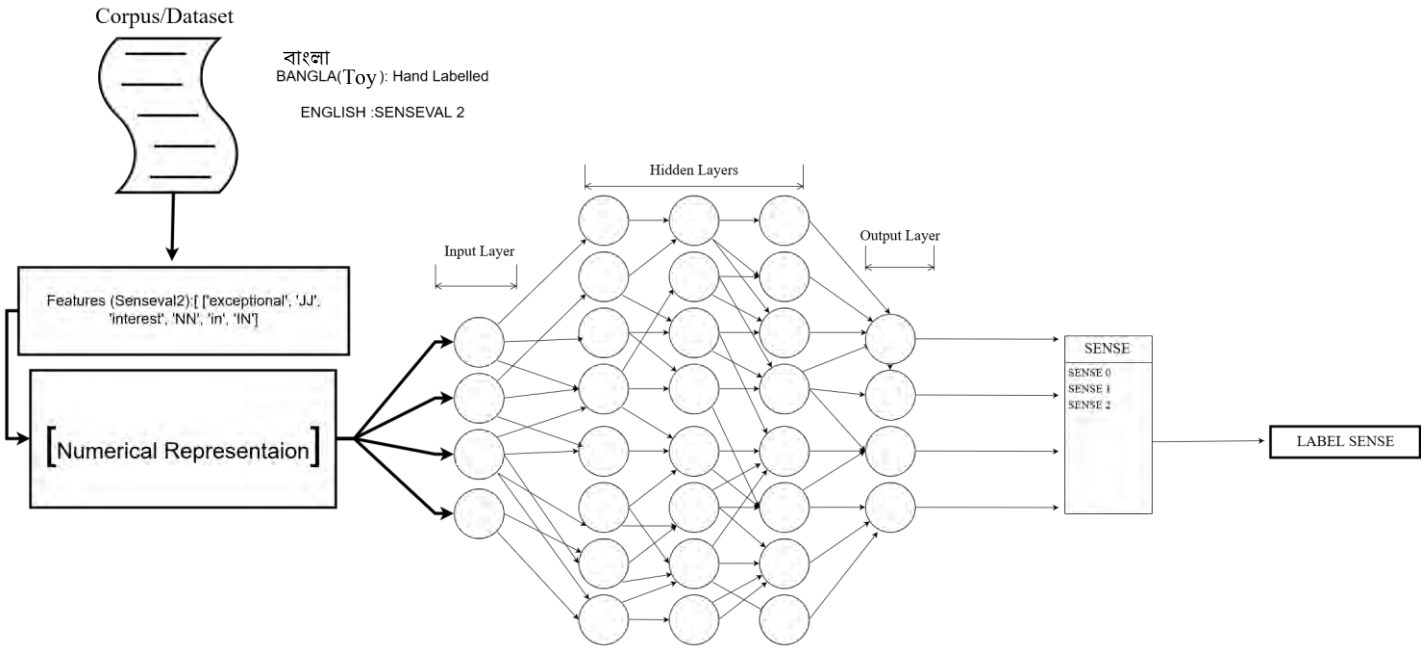



**FIGURE 2 WSD MODEL USING NEURAL NETWORK**

The Neural network classifier is initialized and formed upon the input training data which is refined through pre-processing. The collocational feature vector is formed from the words at different positions and the parts-of-speech tag of those words. This collocation vector is then fed into the neural network as a numerical matrix representation. The neural network calculates weights for links and computes activation value in each neuron or unit.

Each unit in the neural network usually does two major computation components:

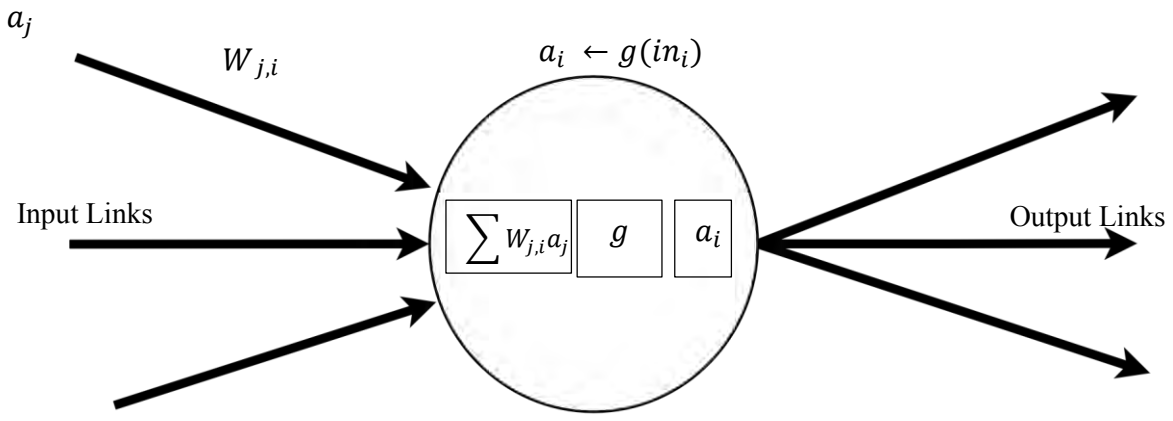
1) A linear calculation on the summation of incoming activation values $a_j$ and the weights on the links $W_{j,i}$.

2) Another non-linear activation function, $g$, computation.

$$in_i = \sum_j W_{j,i} a_j = W_i \cdot a_i$$

| | |
|---|---|
| $W_{j,i}$ | Weight on the link from unity to unit *i* |
| $a_j$ | Activation value of unit *i* (also the output of the unit) |
| $in_i$ | Weighted sum of inputs to unit |

$$a_i \leftarrow g(in_i) \leftarrow g\left(\sum_j W_{j,i} a_j\right)$$




Input Function | Activation Function | Output

As stated by [15], it is important to note that if the selected network is too large, it will be able to memorize all the examples by building a large lookup table. However, it will not generalize well to inputs that have not been seen before. Like all statistical models, neural networks are subject to overfitting when there are too many parameters (i.e., weights) in the model [15].

## 3.7 Hardware Specification:

### CPU:

Processor:

| | |
|---|---|
| Name | Intel Core i5 650 |
| Max TDP | 73.0 W |
| Package | Socket 1156 LGA |
| Technology | 32 nm |
| Core voltage | 0.912 V ~ 0.95 V |
| Family | 6 |
| Ext. Family | 6 |
| Model | 5 |
| Ext. Model | 25 |
| Stepping | 2 |
| Revision | C2 |

**TABLE 1: PROCESSOR SPECIFICATION**

Clocks (Core #0):

| | |
|---|---|
| Core Speed | 3.20 GHz (max) |
| Bus Speed | 133.34 MHz |
| QPI Link | 3200.11 MHz |
| | |

**TABLE 2: PROCESSOR CLOCK**

Caches:

| | | | |
|---|---|---|---|
| L1 Data Cache | Size | 32 Kbytes | x 2 |
| | Descriptor | 8-way set associative | 64-byte line size |

| | | | |
|---|---|---|---|
| L1 Instruction Cache | Size | 32 Kbytes | x 2 |
| | Descriptor | 4-way set associative | 64-byte line size |

| | | | |
|---|---|---|---|
| L2 Cache | Size | 256 Kbytes | x 2 |
| | Descriptor | 8-way set associative | 64-byte line size |

| | | | |
|---|---|---|---|
| L3 Cache | Size | 4 Mbytes | |
| | Descriptor | 16-way set associative | 64-byte line size |

**Memory:**

---

| | |
|---|---|
| Memory Type | DDR3 |
| Memory Size | 4 Gigabytes |
| Channels | Dual, (Symmetric) |
| Memory Frequency | 666.7 MHz (4:20) |
| CAS# latency (CL) | 9.0 |
| RAS# to CAS# delay (tRCD) | 9 |
| RAS# Precharge (tRP) | 9 |
| Cycle Time (tRAS) | 24 |
| Row Refresh Cycle Time (tRFC) | 74 |
| Command Rate (CR) | 1T |
| Uncore Frequency | 2400.1 MHz |

Chipset:

---

| | |
|---|---|
| Northbridge | Intel Havendale/Clarkdale Host Bridge rev. 12 |
| Southbridge | Intel H55 rev. 06 |
| Graphic Interface | PCI-Express |
| PCI-E Link Width | x4 |
| PCI-E Max Link Width | x16 |

**GPU:**

| | |
|---|---|
| Name | NVIDIA GeForce GT 740 |
| Revision | A2 |

GPU Engine Specification:

| | |
|---|---|
| CUDA cores | 384 |
| Base Clock (MHz) | 993 |
| | |

**TABLE 3.1: GPU ENGINE SPECIFICATION**

GPU Memory Specification:

| | |
|---|---|
| Memory Clock | 1.8 Gbps |
| Standard Memory Configuration | 2048 MB |
| Memory Interface | DDR3 |
| Memory Interface Width | 128-bit |
| Memory Bandwidth (GB/sec) | 28.8 |

**TABLE 4: GPU MEMORY SPECIFICATION**

GT 740 Feature Support:

| | |
|---|---|
| Microsoft DirectX | 12 API |
| OpenGL | 4.4 |
| Bus Support | PCI Express 3.0 |
| Supported Technologies | 3D Vision, DirectX 12, TXAA, FXAA |
| PhysX | Yes |
| NVIDIA surround | Yes |
| Adaptive VSync | Yes |
| 3d Vision Ready | Yes |

**TABLE 5: GPU FEATURE SUPPORT**

Display Support:

| | |
|---|---|
| Multi Monitor | 3 Displays |
| Maximum Digital Resolution | 3840x2160*<br>4096x2160 |
| Maximum VGA Resolution | 2048x1536 |
| HDCP | Yes |
| HDMI | Yes |
| Standard Display Connectors | Dual Link DVI-I, Dual Link DVI-D, mini HDMI |
| Audio Input for HDMI | Internal |

*3840x2160 at 30Hz or 4096x2160 at 24Hz supported over HDMI. 3840x2160 at 60Hz supported over Display Port.

**TABLE 6: GPU DISPLAY SUPPORT**

Graphics Card Dimensions:

| | |
|---|---|
| Height | 4.38 inches |
| Length | 5.7 inches |
| Width | Dual Width |

**TABLE 7: GRAPHICS CARD DIMENSIONS**

Thermal and Power Specification:

| | |
|---|---|
| Maximum GPU Temperature (in C) | 98 |
| Graphics Card Power (W) | 64 |
| Minimum Recommended System Power(W) | 400 |

**TABLE 8: THERMAL AND POWER SPECIFICATION**

# CHAPTER 4

## Experimental Result Analysis

Senseval – 2 Data set output result Using K-Near Neighbors:

| Sense Words | Average Accuracy Score (%) |
|---|---|
| "Interest" | 73.81 |
| …….. | ……. |
| "Serve" | 82.64 |

**TABLE 9: ACCURACY OF KNN ON SENSEVAL 2**

| Total Average Accuracy Score | 78.23 % |
|---|---|

Senseval – 2 Data set output result Using Deep Neural Network Classifier:

| Sense Words | Average Accuracy Score (%) |
|---|---|
| "Interest" | 53.8 |
| …….. | ……. |
| "Serve" | 54.3 |

**TABLE 10: ACCURACY OF DNNC ON SENSEVAL 2**

| Total Average Accuracy Score | 54.05% |
|---|---|

Hand-labelled Bangla dataset output result:

| Sense Words | Average Accuracy Score (%) |
|---|---|
| "কাল" | 63.4% |
| "উত্তর" | 77.3% |
| "ফল" | 84.3% |

**TABLE 11: ACCURACY ON BANGLA DATASET**

| Total Average Accuracy Score | 75% |
|---|---|

Comparison of ‘Serve.pos’ sense output values (Prediction versus Y  test):

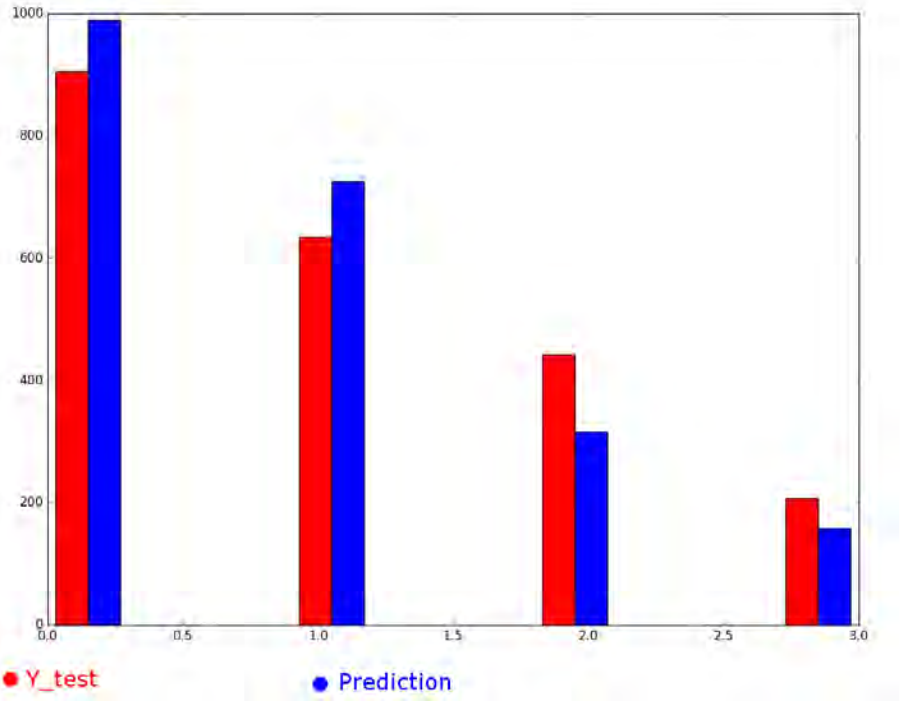



FIGURE 3 COMPARISON OF ‘SERVE.POS’ SENSE OUTPUT VALUES

Comparison of ‘Interest.pos’ sense output values (Prediction versus Y_test):

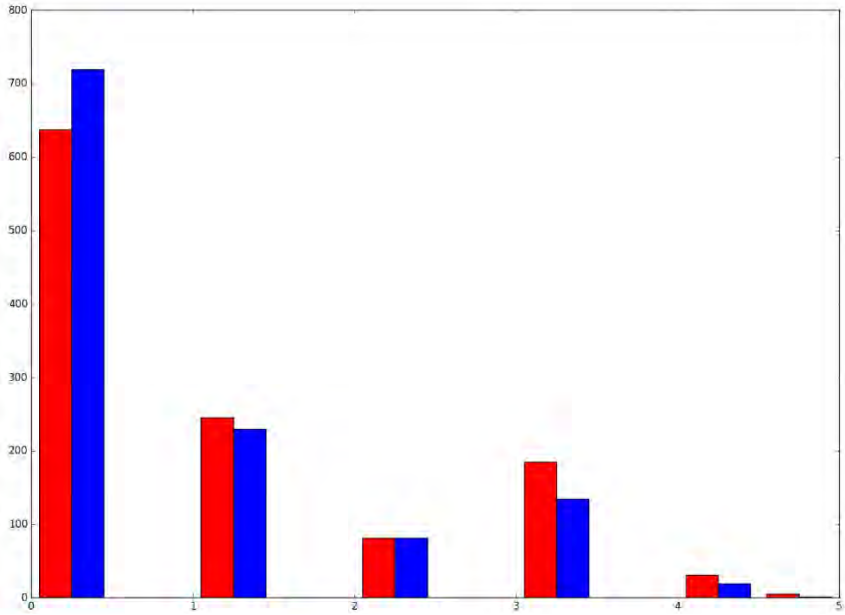



FIGURE 4 COMPARISON OF ‘INTEREST.POS’ SENSE OUTPUT VALUES

After running and training the system on the training data, the built classifier was used to predict sense label for the input feature vectors of new test data and the results were recorded.

The neural network classifier was initialized and fitted with 4 hidden layers with 100,70,40,30 neurons on each layer respectively and on test had an accuracy score of 52.3% on initial experiment on the senseval-2 English lexical sample dataset split into training and testing data. However, fitting the training data on a 3-layer neural network with 10, 60 and 10 neurons on each layer had a better accuracy score of 54.2% on initial experiment. Iterative steps to train the model beyond 200 did not make significant difference to the output classification result. A possible explanation for this difference might be over fitting, however further analysis and iterations on more different dataset and feature vector is necessary to pinpoint the factors causing this.

For, the K Near Neighbors algorithm classifier, a better accuracy score on average was achieved. The classifier was able to even hit 82% for one of the sense instances for the senseval-2 data. A large variance was still present in output result for different sense context words. However, a satisfactory result was still reached.

The hand-labelled sense Bangla dataset had a decent accuracy score on initial experiments. However, a further analysis and experiments on large standard sense-tagged data are necessary to make any significant claims on the models' accuracy in disambiguating Bangla word senses for lexical sample tasks. With more improvements, there is very good potential and possibility for these models to be effective on Bangla language.

# CHAPTER 5

# Conclusion and Future Work

## 5.1 Conclusion

This thesis presented and discussed the effectiveness of word sense disambiguation using supervised learning and a deep neural network classifier on both English and Bangla data. After collecting the SENSEVAL 2 lexical sample data set and applying the individual models, it was evident that the K near neighbors supervised classifier performed with satisfactory results. The accuracy score turned out to be much better than anticipated. On the other hand, the Neural network classifier still has room for improvement. The neural network classifier had significant noise and major variance in accuracy. The neural network implementation requires even further experiments and tweaking in terms of layers, pre-processing and neuron unit count. In both models, the English dataset resulted in more standard and accurate results, since the dataset used, SENSEVAL 2, was a large and standard one. However, the Bangla dataset results are yet to be more accurate, since the data were hand labelled and was relatively small. A larger dataset of Bangla sense tagged data would work even better and would result in a better accuracy score. In general, both the systems were able to perform fairly efficiently and effectively with very fast response time. The systems and implementations presented in this thesis definitely hold practical value in natural language processing and with more experiments, better results can be achieved in the future.

## 5.2 Future Works:

There were limitations in analyzing the word sense disambiguation models on Bangla language due to the lack of sense tagged resources. In the future, I would like to further investigate and improve the systems on Bangla using a larger dataset and perhaps newer models. Additionally, I would like to improve the deep learning system and implement it with speech to build a stable and efficient intelligent system that can have a fluid and clear conversation with a human in multiple languages.

## References


[1] R. Navigli, "Word Sense Disambiguation: A Survey," *ACM Computing Surveys (CSUR),* vol. 41, no. 2, p. 61, February 2009.

[2] M. Kågebäck and H. Salomonsson, "Word Sense Disambiguation using a Bidirectional LSTM," Goteborg, 2016.

[3] A. Das and S. Sarkar, "Word Sense Disambiguation in Bengali applied to Bengali-Hindi Machine," 2013.

[4] D. Jurafsky and J. H. Martin, Speech and Language Processing, 2 ed., Prentice Hall, Pearson Education International, 2014.

[5] D. Yarowsky, "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," Philadelphia.

[6] Y. K. Lee and H. T. Ng, "An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation," in *Conference of Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, 2002.

[7] D. Yuan, R. Doherty, J. Richardson, C. Evans and E. Altendorf, "Word Sense Disambiguation with Neural Language Models," 2016.

[8] Y.-J. Chung, S.-J. Kang, K.-H. Moon and J.-H. Lee, "Word Sense Disambiguation Using Neural Networks with Concept Co-occurrence Information," in *Sixth Natural Language Processing Pacific Rim Symposium*, Tokyo, Japan, 2001.

[9] NVIDIA Corporation, "CUDA Toolkit | NVIDIA Developer," NVIDIA Corporation, [Online]. Available: https://developer.nvidia.com/cuda-toolkit. [Accessed 2016].

[10] NVIDIA Corporation, "About CUDA | NVIDIA Developer," NVIDIA Corporation, 2016. [Online]. Available: https://developer.nvidia.com/about-cuda. [Accessed August 2016].

[11] J.-H. Huang, "Accelerating AI with GPUs: A New Computing Model," January 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/. [Accessed August 2016].

[12] "Tensorflow," 2016. [Online]. Available: www.tensorflow.org. [Accessed August 2016].

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher and M. Perrot, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* pp. 2825-2830, 2011.

[14] J. Vanderplas, "scikit-learn|Nearest Neighbors," [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html. [Accessed August 2016].

[15] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, New Jersey: Prentice Hall, 2009.

[16] N. M. IDE and J. VERONIS, "Very Large Neural Networks for Word Sense Disambiguation," in *European Conference on Arificial Intelligence*, Stockholm, 1990.

[17] B. Pang, L. Lee and S. Vaithyanathan, "Sentiment Classiflcation using Machine Learning," in *Conference on Empirical Methods in Natural Language Processing(EMNLP)*, Philadelphia, 2002.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, Vanhoucke, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow:Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2015.