ROBUST WORD PREDICTIONS

A Thesis

by

S VENKATA SATISH KUMAR PASUMARTHI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2019

Major Subject: Computer Engineering

ABSTRACT

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI) that allows machines to process and comprehend human languages in order to bring machines nearer to language understanding. In older days, statistical methods were predominant where the rules were written / calculated manually. Recent advances in Machine Learning and Deep Learning have led to many breakthroughs in various sub-fields of NLP which include language modeling, machine translation, speech recognition etc. Language Model (LM) forms a building block of all the NLP applications where the task is to predict the next word given previous words. The main drawback of the language model is that it is limited to predicting next word and the training requires significant amount of vocabulary.

This research aims for the development of generalized language model in which the prediction is not just limited to next word but to any word in the future text. The research tried to expand the horizons of language modeling problem and make it more generalized in terms of understanding context as well as making prediction. This work proposes a Neural generalized language modeling technique and tested on two kinds of databases i.e., BBC News articles as well as Wikipedia exploring the use of word embeddings, attention mechanism to understand the context and making context based word predictions. The main focus lies on predicting non-stop words and meaningful words. Also, this work explored the memory capabilities and Noise tolerance of the developed model.

DEDICATION

To my Parents, Teachers and the Almighty.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Anxiao (Andrew) Jiang, Department of Computer Science and Engineering, for his constant supervision, guidance, suggestions, and invaluable encouragement through out my career as a graduate student at Texas A&M University. I would like to thank Dr. Peng Li and Dr. Weiping Shi, for willing to be a part of my thesis committee and review panel and providing thoughtful insights.

I am grateful to Prafulla Choubey for sharing his knowledge and involvement in discussion with me on deep learning and Natural Language Processing which improved my knowledge and understanding of the subject.

I am grateful to the Department of Electrical and Computer Engineering at Texas A&M University for providing me financial support in the form of one time graduate merit scholarship, and Student Grader. I am also thankful to my academic advisors Katie Bryan and Melissa Sheldon for all their help throughout my master's degree.

Finally, I would like to thank my family and friends for their encouragement and moral support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Anxiao (Andrew) Jiang of the Department of Computer Science and Engineering, Professor Weiping Shi, and Professor Peng Li from the department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work did not receive any external funding.

NOMENCLATURE

ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
LM	Language Model
NLM	Neural Language Model
NLP	Natural Language Processing
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
TF	Term Frequency
IDF	Inverted Document Frequency
GUI	Graphical User Interface

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW

Language Models (LM) are statistical models that assign a probability for next possible word in a word sequence. Language modeling forms the backbone for a wide variety of application in the domain of Natural Language Processing (NLP) [2]. For applications where language understanding is of utmost importance, the language models are usually employed at either front-end or back-end of a much more sophisticated model. Language models can be categorized into two : Statistical based and Neural based. The traditional statistical methods involve making an Markov assumption of n-th order and estimating the n-gram probabilities using subsequent smoothing and counting [3]. The LM literature abounds with successful approaches for learning the count based LM like modified Kneser-Ney smoothing, Jelinek-Mercer smoothing [4] etc.

With the advent of GPU and their ability to handle parallelism, Neural methods have been adopted to language modeling tasks. The use of Neural networks in building the language model is termed as Neural Language Modeling. The deep learning models i.e., nonlinear neural network models can overcome the limitations of the traditional language models. These models eliminate the need for hand written rules and are capable to generalize to various contexts. They are able to condition on progressively large sizes of context with linear growth of weights in the network. Moreover, the Neural Language Models (NLM) were able to solve the data sparsity problem in the n-gram model, by using vector representation for words called as Word Embeddings.[5] These embeddings are passed to NLM as an input. During the training of the network, the weights are updated and learned. Word embeddings generated through Neural LMs have the capability of representing semantically close words as nearby vectors in the derived vector space.

Also, NLMs have the ability to capture and understand the context at various levels i.e., subword, sentence, and corpus. In Natural language Processing, Language Models are evaluated using Perplexity. It is the inverse probability of held-out test set, normalized by the number of words. In mathematical terms, Perplexity is derived by taking exponent of the entropy. Entropy is the average number of bits to encode the information contained in a random variable[6], so the exponent of the entropy gives us the aggregate sum of all conceivable information. It is the weighted average of considerable number of decisions a random variable can make. Mathematically, it can be represented as $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-n+1}..w_{i-1})}}$ where n is the n-gram model and N is the number of words in vocabulary (test set). Lower the perplexity, the better is the model. The standard datasets used for LM tasks are PennTreeBank (PTB)[7], WikiText-2 [7] and WikiText-103 [7] in the increasing order of their complexity. The State of the art models like FRAGE[8] achieved a perplexity of 46.54 on PTB, 39.14 on wikiText-2, and Transformer[9] based Models [10] achieved 16.4 on wikiText-103 respectively. Since the focus of LM is on predicting next word, a unidirectional RNN (left to right) will suffice for the purpose.

In the work by Yoon Kim et al.,[11] a simple neural language model was built that depends on character level inputs but the outputs predictions are word level. They employed a Convolutional Neural Network (CNN) and a highway network over characters, whose output is given to a Long Short Term Memory (LSTM) recurrent neural network language model (RNN-LM). Rafal Jozefowicz et al.,[12] extended current models to address two main difficulties. One is vocabulary and corpus size, and other is complex, long-term language structure. On the One Billion Word Benchmark, they conducted an exhaustive survey of methods such as character Convolutional Neural Networks or Long-Short Term Memory.

Sungjin Ahn et al., [13] propose a Neural Knowledge Language Model (NKLM) which combines symbolic knowledge provided by the knowledge graph with the RNN language model. By foreseeing whether the word to create has an fundamental reality or not, the model can produce such knowledge-related words by replicating from the depiction of the anticipated truth.

This work employed a bidirectional RNN to make the model understand the flow of text from both directions. Also, techniques like structured self-attention [1] have been proposed in literature for extracting an interpretable embedding which uses a matrix instead of vector to represent the embeddings. These techniques were proven to be working well for problems like Question Answering, sentiment classification etc. where the model has to focus on specific words. We explored employing self-attention to see if it helps the language model.

2. DATA PREPARATION AND AN ATTEMPT AT VISUALIZATION

Since we want our model to learn about the text in a generic sense and not specific to a particular area, the best possible data source would be Wikipedia and BBC News[14]. Text where the prior sentence is related to the following one which acts as very good training source for our model. The Wikipedia dataset can be downloaded from *here*. The data is downloaded in the XML format and later converted to .txt format by processing the XML files. There are around 3.8M articles in the Wikipedia dataset which cover around 170K wide variety of areas. We also chose BBC News articles dataset[14] as it is a smaller dataset and the articles are organized into 5 categories contributing a total of 2225 articles.

Category	**articles**
technology	401
politics	417
sports	511
business	510
entertainment	386

Table 2.1: Table showing the number of articles per category in the BBC News Dataset

The raw data contains lot of unnecessary information for the model like stop words, numbers etc. This raw data is then processed to be made available for the DNN model. Since we want to predict words in the future sentences, we consider first half of the document as source text and the tokens from rest of the document as target text. We ignore the article if the number of sentences are less than 10. The data has undergone many preprocessing steps before being fed to the model. Each of the step and its importance are explained below:

2.1 Data Preparation Pipeline

The first half of the document is undisturbed while the second half of the text undergoes the below processing steps to become meaningful targets.

2.1.1 Word Tokenization

The first step is to split each sentence into separate words or tokens. This is called word tokenization. For example if we have a sentence “cat sat on the couch”, after applying word tokenization we have “cat” , “sat”, “on”, “the”, “couch”. Tokenization in English is done by splitting the sentence into words whenever a space is encountered.

2.1.2 Lemmatization

In languages especially, words may appear in various forms. Consider the two sentences below:
I had a toy
I had two toys
Even though both the sentences are mentioning the noun “toy”, they have different inflections. In language, having knowledge about the base form of a word helps us understand that both the sentences are discussing the same thing. If we do not have the knowledge of the base form (lemma), the words “toy”, “toys” look completely different to a machine.

In NLP, identifying the basic form or lemma of the words in a given sentence is known as lemmatization.

2.1.3 Stop words

The words that appear very frequently in text are called stop words. English language has a lot of stop words that appear very frequently like “an”, “and”, and “is” etc. When textual analysis is done, these words present a part of clamor since they show up way more regularly than other words. We don’t want the model to predict such words and hence these kind of words are removed and won’t be present in target text.

2.1.4 Smaller words

There are many words of length < 3 in the corpus which are not stop words. For example the words like *be*, *me* are neither stop words nor useful for prediction. So we remove the words which have length less than 3

2.1.5 Numerical data

In the articles we consider for training our model, there will be many numerical words like years, currency etc. It won't be useful to learn the numerical data and hence we remove all such words.

2.1.6 Term Frequency

A word's term frequency (TF) is defined as the number of occurrences in a particular document. Since we want the model to predict words from the second half of the document, we remove the occurrence of any duplicates and make their term frequencies of all the words equal to one.

2.1.7 Inverted Document Frequency

The Inverted Document Frequency (IDF) of a word is given by inverse function of the number of documents in which it occurs. It is a measure of importance of words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score. IDF of a word w is given by $idf(w) = log(\frac{N}{df_w})$ where
N represent the total number of documents in the corpus
df_w represents the number of documents which has the word

2.1.8 TF-IDF

In the attempt to find the important words for predictions, we used TF-IDF score as metric. Since TF for every word is one, we focus on IDF score. We consider words only present in less than 100 documents and more than 5 documents. This basically filters out very rare words which doesn't have enough data points to generalize and also the words which are not important.

We do not want our model to be sensitive to the capitalization of the words and hence we covert every word to lower case and even remove the punctuation. Ideally in our problem statement the words "Cricket" and "cricket" does not make any difference.

There are 1.5M data sequences generated from the Wiki corpus.[15] We used 10K and 25K samples from the dataset. Below is a sample data text generated. The Wikipedia article is about famous cartoonist Arnold Roth.[15]

Example 1. *Source text represents the input text, Target represents the target text*

Source text : *Arnold Roth (born february 25, 1929) is an American freelance cartoonist and illustrator for advertisements, album covers, books, magazines, and newspapers. Novelist John Updike wrote, "All cartoonists are geniuses, but Arnold Roth is especially so." He was born in philadelphia, pennsylvania. Roth graduated in from the Philadelphia museum school of industrial art. He began freelancing in the following year. Roth's art is in the collections of the Philadelphia Museum of Art, the museum of cartoon art (San Francisco), Philadelphia's Rosenbach Museum and Library and the Karikature and Cartoon Museum (Basel, Switzerland), plus many private collections. Roth has done covers for the New Yorker and his artwork has appeared in tv guide sports illustrated and esquire. His cartoons and illustrations were contributions to the satirical magazines edited by his friend Harvey Kurtzman: Trump (1957), Humbug (1957 - 58) and Help! (1960-65). Roth's cartoons began appearing in Playboy in the late 1950s.*

Target text : *Roth was a political cartoonist for The Progressive from 1981 to 1987. He drew the comic strip Poor Arnold's Almanac as a Sunday strip from 1959 to 1961. He received the National Cartoonists Society Advertising and Illustration Award; Magazine and Book Illustration Award; Sports Cartoon Award; Reuben Award; and their Gold Key Award. On June 25, 2009, Roth was inducted into the Society of Illustrators Hall of Fame which honors artists for their "distinguished achievement in the art of illustration." Past Society presidents select inductees based on their body of work and the impact on the field of illustration.*

Yellow implies stop words, words with less than 3 characters, too frequent words, less frequent words. Magenta implies words which are common between source and target texts. The words without any color background are the target words to be predicted.

In this data sample, source text is talking a cartoonist. The target text mentions keywords like artists, achievement, induct, honors, comic etc. are of interest to us. We assume equal probabilities for all these words and expect our model to predict the target words (with out color background).

2.2 Visualizing text as Path

Visualization is an important tool for analyzing natural languages and the knowledge they contain. Word-level embedding tools, including word2vec and GloVe, represent the adjacency relations between words by embedding words to a high-dimensional space, where nearby words in the embedding space are more likely to be near each other in natural languages. So the word embedding can be seen as a "summary" of natural languages, and a text (such as a sentence, a paragraph or an article) can be seen as a path in the embedding space. We observer, however, that the path corresponding to a text is typically not smooth at all in the embedding space. In fact, it zigzags wildly. Let us look at an example sentence from Wikipedia [15] for visualization.

Example 2. *"Robot is a machine, especially one programmable by a computer capable of carrying out a complex series of actions automatically"[16]*

The visualization of above sentence using word2vec embedding is shown in Figure 2.1. First, the word embeddings of words in the sentence which are d-dimensional are projected onto 2-D or 3-D space using t-SNE [17]. We do this for every non-stop word in the sentence (which is a node in the path), we also show its three nearest neighboring words.

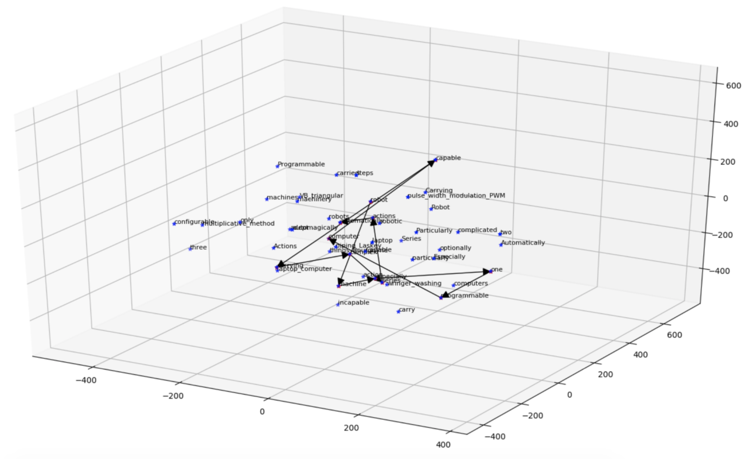
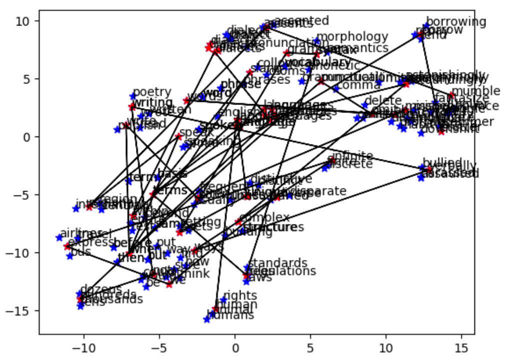
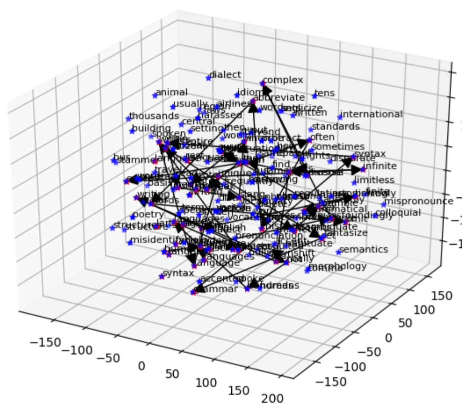


Figure 2.1: The path for a sentence in 3-d space, using word2vec



(a)



(b)

Figure 2.2: Visualizing path for a paragraph (a) In 2-D space, using GloVe of 100 dimensions, & (b) In 3-D space, using GloVe of 100 dimensions

Although the path for a text zigzags wildly, it is actually not zigzagging completely randomly. The reason is that the word embedding still captures the adjacency of words in texts. But that adjacency is for all texts as a whole, so when it comes to an individual text, its variance can be very large, which corresponds to the zigzagging. That means the path for a text actually has some smoothness (namely, locality), but the smoothness is not enough yet. The smoothness of a path is strongly related to the predictability of the path. That is, given a sequence of words in a text, if we can predict what the next words will be, then the text is said to be predictable, and the path for the text should be relatively smooth. We illustrate this relationship with the following example.

We show two curves, one smoother [Fig 2.3a] and one less smooth [Fig 2.3b]. Usually for a smoother curve, it is easier to predict what the future values will be. We can use the same intuition for natural languages. We can think of a text as a sequence of words, like a discrete function. Every word is a vector in an embedding space. If we can predict future words in a text well based on its past words, then it means we have discovered some "smooth structure" in the natural language. We can then visualize and analyze the smooth structure in the embedding space.

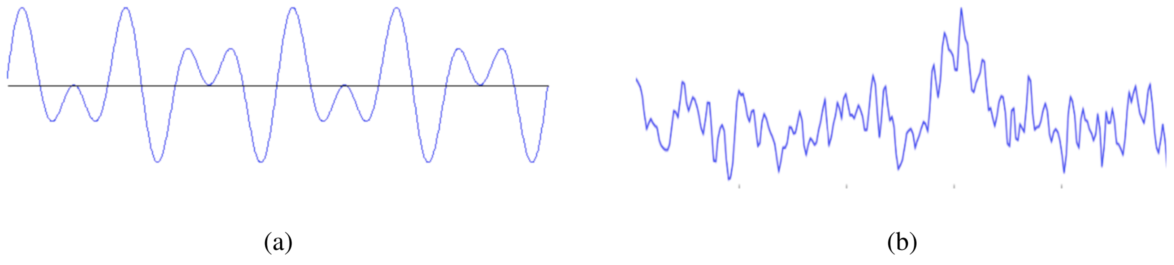

(a) (b)

Figure 2.3: Predicting future values using visualization (a) Easier for smoother curve, & (b) Difficult for random curve

Using deep learning we attempt to study the high complex flow in the language text. As we have discussed, it is closely related to finding a smooth structure in the language. If the Deep Learning models can understand their structure in high dimensional space, we can successfully predict the future words in the text.

3. ROBUST WORD PREDICTIONS USING DEEP LEARNING

3.1 Problem Statement

In the field of computation linguistics and probability, an n-gram is a contiguous sequence of n items from a specified speech or text. These items can be characters or words depending on the application. Language model tries to predict the next word given an n-gram which can be modelled as $P(W_{n+1}|W_nW_{n-1}W_{n-2}....W_1)$.

For example if we are asked to predict the next word in the sentence *Please turn your homework ...*, we can easily guess the word would be *in*, or possibly *over*. This is what language model does, it assign probabilities to all the words in the corpus and the word with the highest probability will be considered as the prediction. In this research, we worked on a novel problem of predicting second half of the document. We worked to solve this problem by building a generalized language model using neural methods where the prediction is not limited to next word instead predict all possible words in the rest of the sentence or document, as shown below. To help understand the problem statement better, let's consider a small example text about robot from Wikipedia[16]:

Example 3. *"A robot is a machine especially one programmable by a computer capable of carrying out a complex series of actions automatically. Robots can be guided by an external control device or the control may be embedded within."*

The text has two sentences so we divide the first sentence to input and the second to output. When we feed in the first sentence (input) to the network, it should be able to predict the words in the second sentence. The sentence "A robot is a machine especially one programmable by a computer capable of carrying out a complex series of actions automatically." is passed as **input** and the words *{"robots", "guided" , "external", "control", "device", "embedded"}* will be the **expected outputs**.

Given a document, we feed in the first half of the document to the model and let the model guess words in the second half. This problem of predicting the rest of the words in the documents belongs to the category of Multi Label Classification problem in literature. This is a very challenging task as the model first has to understand the context and predict words related to the context and the possibilities could be enormous. At times, there could be multiple contexts in the given sequence and it becomes even more challenging. The number of output labels the model has to guess for a given data sample depends on the number of non-stop words in the second half of the document. We measure the model's performance using F1 score which is a harmonic mean of precision and recall which are explained in 3.4.

3.2 Network

We solve the problem of predicting future words based on 3 components which are described in the following subsections.

3.2.1 Architecture

The deep learning architecture we propose contains 3 sections : Embedding, Hidden Representation and Fully Connected. The Embedding layers converts the input words to vector of numbers that can be fed to the model. As the words in the sentence have temporal relation between them, Recurrent neural network (RNN) is a perfect choice for Neural LM. The Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) variants resolve vanilla RNN inadequacies in long-range context modeling. Bidirectional RNNs have the ability to read input from left to right (past to future) as well as right to left (future to past). This lies as main difference between unidirectional RNN where the input is read only from left to right (past to future). In bidirectional RNN, every unit will have two hidden states and by combining these states we are prepared to maintain past and future data at any time step. Usage of LSTM vs GRU is a pure architectural choice. GRU performance is equivalent to LSTM while being computationally and memory effective. Also, in literature the GRUs are preferred over LSTM for language modeling tasks. Hence our choice of RNN is GRU for all the models.

We pass the output of the encoder section which has the representation of inputs in the form of hidden states to next section in the model which is a series of Fully Connected Layers.

In between the fully connected layer, we use tanh activation layer. Tanh activation function has a range from -1 to 1. The advantage of tanh activation function is that the negative inputs will be strongly mapped to negative and the zero inputs are mapped near zero. The tanh activation is monotonic while its derivative is not. For classification problems like these tanh or ReLu will be first choices. We chose tanh over ReLu as it helped better avoid the vanishing gradient problem. For a given input, the number of targets words are very less compared to the target vocab. We want the model push the predictions of non target words to as negative as possible to reduce False Positives.

The output layer will have a size equal to the target vocab. So, the model used for BBC News dataset and Wikipedia dataset will have different number of units in the output layer. Each neuron in the output layer will give a real number as output which are then passed to Sigmoid unit(s) in the MultiLabelSoftMargin loss. During inference, the positive output is treated as 1 and negative output as 0.

The high level architecture along with each layer's input and output dimensions is shown using the block diagram below:

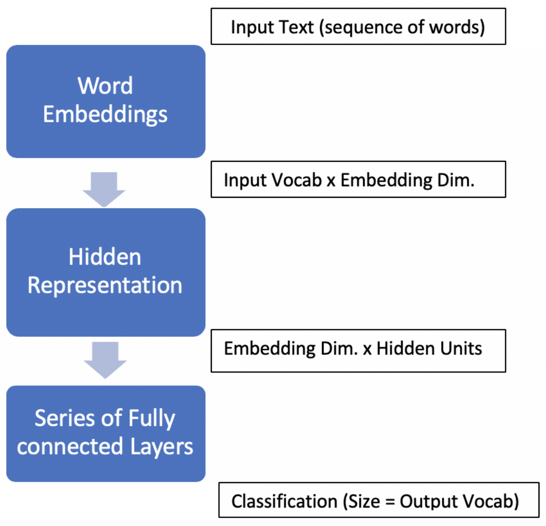

Input Text (sequence of words)
Word Embeddings
Input Vocab x Embedding Dim.
Hidden Representation
Embedding Dim. x Hidden Units
Series of Fully connected Layers
Classification (Size = Output Vocab)

Figure 3.1: Block diagram representation of model

Following this generalized structure, the models we experimented are shown below along the input and output of each layer.

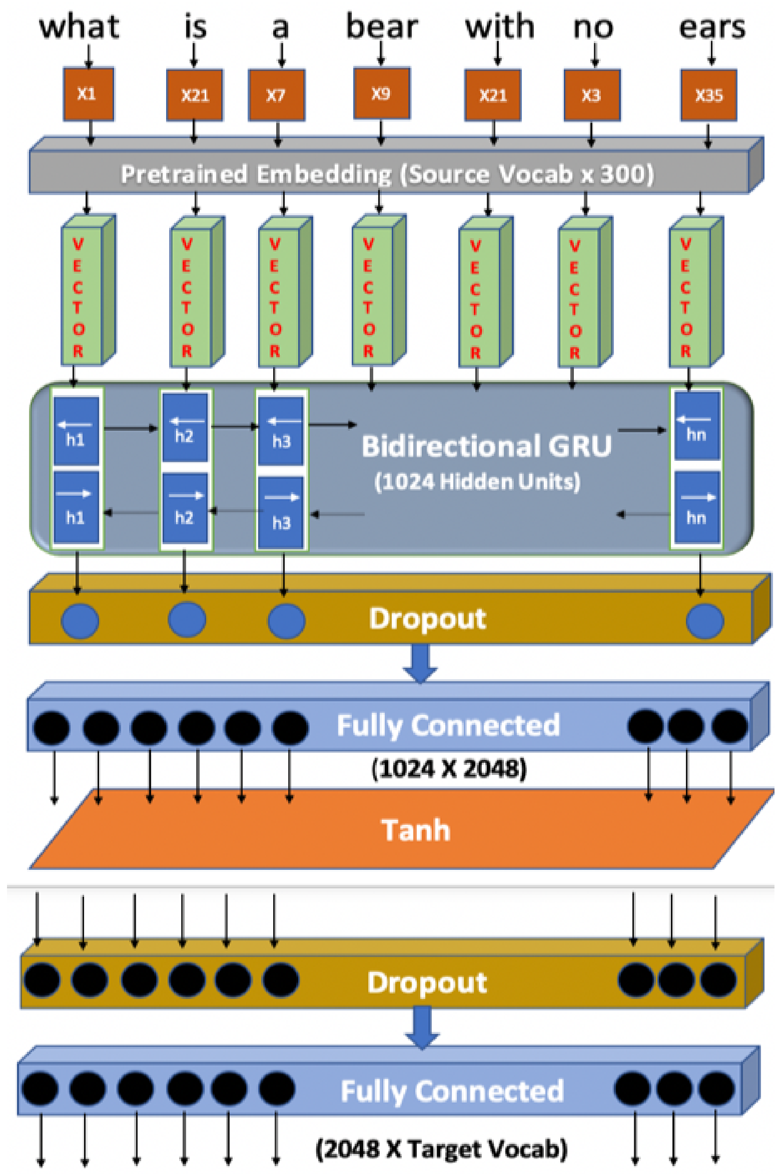
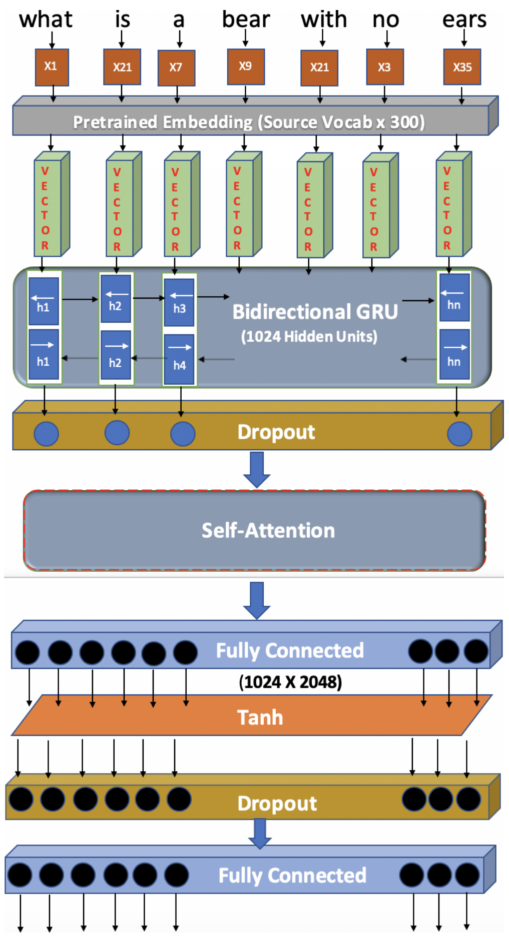


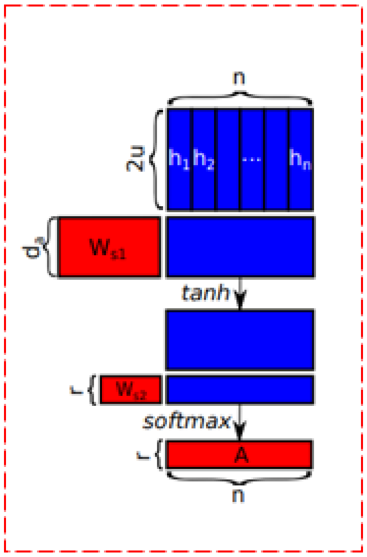
Figure 3.2: Generalized Neural Language Model

3.2.2 Structured Self-Attention

In most of the models the embedding is represented using a vector, in structured self-attention work by Lin et al., [1] a much better representation for embedding an input sentence, which is a 2-Dimensional matrix, is used. Each row (called **hop**) of the matrix attends to a different part of the input. This mechanism if added to our existing model, the new model is supposed to perform better than earlier as we are trying to embed the input data in multiple ways making the model to make predictions based on multiple words.



(a)



(b)

Figure 3.3: Structured Self-Attention Model (a) Complete Model with Self-Attention layer (highlighted with dotted lines) & (b) *Representation of Self-Attention layer. Reprint from [1]

As opposed from previous attention approaches, this self-attention mechanism allows extracting different aspects of the sentence into multiple vector representations.[1] This self-attention model has two important blocks. a bidirectional GRU, and self-attention mechanism. The attention mechanism provides a weighted vectors for the GRU encoding ie., the hidden states. The GRU hidden states are dotted with the generated weighted vector to generate the input sentence embedding. The attention mechanism takes the whole GRU hidden states H as input, and outputs a vector of weights **a** given by [1]

$$\mathbf{a} = softmax(\mathbf{w_{s2}} tanh(W_{s1}H^T)) \tag{3.1}$$

Here W_{s1} is weight matrix with shape d_aX2u, w_{s2} is a vector of parameters with size d_a, d_a is a hyperparameter. Since H is sized $nX2u$, the annotation vector a will have a size n, the $softmax()$ is to make the weights sum to 1. The input sentence embedding is thus generated by dotting the GRU hidden states with the weighted vector. But in our work, the input is not just single sentence instead a sequence of sentences. Thus having a single hop of attention wouldn't suffice. Hence instead of generating annotation vector, an annotation matrix A is generated by extending w_{s2} into rXd_a matrix, making it W_{s2}. This can be written as [1] :

$$A = softmax(W_{s2} tanh(W_{s1}H^T)) \tag{3.2}$$

The above equation can be viewed as a 2-layer MLP without bias, with parameters W_{s2}, W_{s1}. The resulting sentence embedding is given as :

$$M = AH \tag{3.3}$$

A penalizing term is introduced to make the weights diverse otherwise this mechanism could gives similar weighted vectors for every hop. The penalization term is given by [1]:

$$P = ||(AA^T - I)||_F{}^2 \tag{3.4}$$

where $||.||$ represents Frobenius norm of a matrix.

We also tried ensemble model which is called a 2 pass model. In the first pass, the prediction is done using Model1 as usual where as for the second pass (for Model2) the expected output will be filtered i.e., the predictions made in the first pass will be excluded in the second pass. This type of ensemble is used when we want the second model to learn what the first model didn't. Any of the above represented models can be used in the place of the blocks represented by "Model1/Model2"

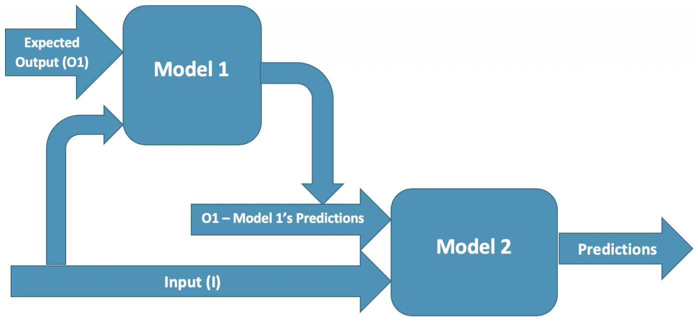


Figure 3.4: Ensemble (2 pass) Model

3.3 Loss Function

Since we are trying to do multiLabel classification, **MultiLabel SoftMargin Loss** from pytorch[18] is our default choice. MultiLabelSoftMargin loss optimizes a multi-label one-versus-all loss based on max-entropy, between input x and target y of size (N, C) where N, C represent number of samples and number of labels respectively[18].

$$loss(x, y) = -\sum_i y[i] * log(\frac{1}{1 + exp(-x[i])}) + (1 - y[i]) * log(\frac{exp(-x[i])}{(1 + exp(-x[i]))}) \tag{3.5}$$

where i = {0 .. size(x)-1}, y[i] in {0,1}

The MultiLabelSoftMargin Loss is nothing but a layer of sigmoids followed by Binary Cross Entropy (BCE) for each unit. Unlike, Categorical Cross Entropy loss here the loss tries to increase the confidence of all the positive labels simultaneously as we have multiple labels are outputs. We are using Pytorch [18] implementation of MultiLabelSoftMargin loss utility in our models. The loss function can be visualization as shown below:

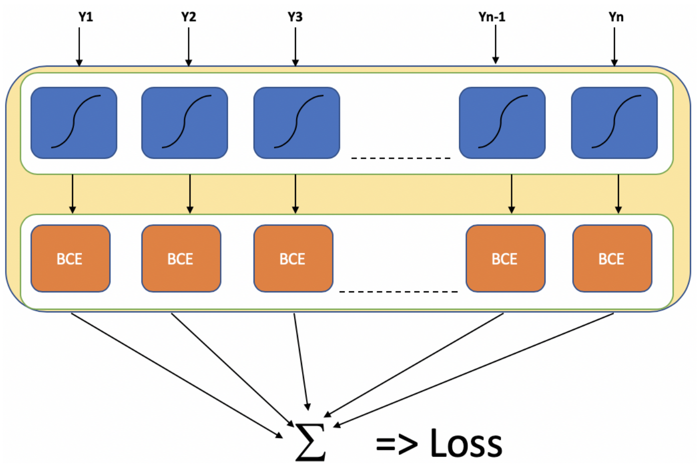


Figure 3.5: Visualization of MultiLabel SoftMargin Loss. The first layer is layer of sigmoids which gets its input from the output layer of the network. The second layer is Binary Cross Entropy (BCE) calculation for each sigmoid output. Finally all the binary cross entropy outputs are added to get the accumulated loss.

Let's assume we have 3 labels namely "A", "B" and "C" $(i = 3)$. For each label we define some actual prediction (y) which tells whether the label is associated with the input. Similarly, x represents the model output. Here y is a binary value while x is a real number. We will present two cases i.e., correct and incorrect predictions and analyze how loss is computed during both the cases.

Case 1. When the model makes incorrect predictions

	label “A”	label “B”	label “C”
y (Actual)	1	0	1
x (Prediction)	-10	2	9

Table 3.1: Table depicting a scenario when the predictions are incorrect for label A and label B, correct for label C. The second row y (Actual) represents a binary value whether the label is correct prediction for the input. 1 represents correct whereas 0 represents incorrect. The last row x (prediction) shows the real output values which are later passed to softmax layer which is included in the loss function.

For label A, the loss is calculated as $loss_A = -1 * log(1 + exp(10))^{-1} = -4.3429$

For label B, the loss is calculated as $loss_B = -1 * log(\frac{exp(-2)}{1+exp(-2)}) = -0.9237$

For label C, the loss is calculated as $loss_C = -1 * log(1 + exp(-10))^{-1} = 0$

The total loss for this example is $\sum_{i=A,B,C}(loss_i) = -5.2666$

The loss function penalizes the model heavily for labelA since it predicted negative number (-10) instead of positive number, moderately for label B since it predicted a small positive number (2) instead of negative number for incorrect label.

Case 2: When model makes correct predictions

	label ’A’	label ’B’	label ’C’
y (Actual)	1	0	1
x (Prediction)	10	-10	10

Table 3.2: Table depicting a scenario when the predictions are correct for all labels. The second row y (Actual) represents a binary value whether the label is correct prediction for the input. 1 represents correct whereas 0 represents incorrect. The last row x (prediction) shows the real output values which are later passed to softmax layer which is included in the loss function.

For label A, the loss is calculated as $loss_A = -1 * log(1 + exp(-10))^{-1} = 0$

For label B, the loss is calculated as $loss_B = -1 * log(1 + exp(-10))^{-1} = 0$

For label C, the loss is calculated as $loss_C = -1 * log(1 + exp(-10))^{-1} = 0$

The total loss for this example is $\sum_{i=A,B,C}(loss_i) = 0$

In this case, the loss function does not penalize the model as the predictions are correct i.e., Positive values are predicted for correct labels and negative numbers for incorrect labels.

3.4 Performance

As mentioned earlier, the output labels are highly imbalanced and hence using *accuracy* is not a meaningful metric for this kind of problem. While dealing with highly imbalanced labels in the output, F1 score is a much better metric. F1 score tries to balance Precision and Recall by calculating Harmonic mean of them. The minimum possible value for F1 score is 0 whilst the maximum possible is 1.

	Actual (+ve)	Actual (-ve)
Model (+ve)	True Positive (TP)	False Positive (FP)
Model (-ve)	False Negative (FN)	True Negative (TN)

Table 3.3: Confusion Matrix showing how the values of True Positive, False Positive, False Negative and True Negative are calculated. +ve meaning predicting as 1 and -ve meaning predicting 0.

The precision and recall values can be calculated using the below formulae:

$$Precision = TP/(TP + FP) \tag{3.6}$$

$$Recall = TP/(TP + FN) \tag{3.7}$$

F1 score is the harmonic mean of precision and recall. F1 balances the precision and recall equally and the possible range of F1 score is [0,1] with 0 being poor performance and 1 being very good performance. It is calculated as

$$F1 = 2 * precision * recall/(precision + recall) \tag{3.8}$$

In our case, the model outputs real values for possible words. Since the loss function inherently applies softmax we don't need to specifically include softmax activation at the network output. But for accuracy calculation we need to convert them to binary values. In order to achieve this we use 0 as threshold and make all positive output values as **1** and all negative values as **0**. This will be our model prediction. An example of binary classified model output is shown below:

$$y_{pred} = [0\,1\,0\,1\,1\,0\,.........\,1] \tag{3.9}$$

Our ground truth vector y_{true} contains the indices of the words in the vocab as shown below:

$$y_{true} = [1,\ 4,\ 13,\ .....\,23,\ 45] \tag{3.10}$$

Since y_{pred} contains the binary values corresponding to each word and y_{true} contains the indices of the true words, we convert the y_{pred} to an array of indices using array operations. Now we have two arrays which contains the array indices which enables us to proceed with precision and recall calculation.

3.5 Training and Prediction

The model was trained and experimented on two different datasets namely BBC News articles and portion of Wikipedia dataset. The best performance results are presented here in the following subsections.

3.5.1 BBC News Dataset

BBC news articles dataset which contains 2225 articles that are categorized into 5 topics namely politics, business, entertainment, sport and technology. The documents are almost balanced i.e., more or less the number of documents in each category is nearly same. We made random 333 articles as validation and random 333 articles as testing dataset. Below are some of the details of the dataset. During training we limited the source and target text lengths to 15.

Now using this dataset, the model is trained on documents from 5 distributions and the train, test and validation distributions are not mutually exclusive. For the model, the words in the target vocab are made sure their frequency is 5 and maximum frequency is 100 so that the output vocab size is not too huge.

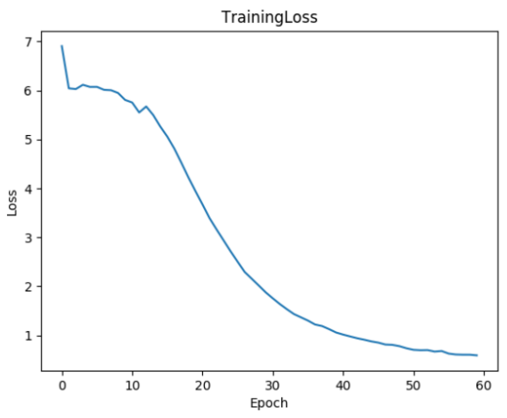
Model Training and Optimization

The network contains a 1024 hidden unit Bidirectional GRU for encoding the input text. The model is trained for 60 epochs on Nvidia 1080 Ti GPU.

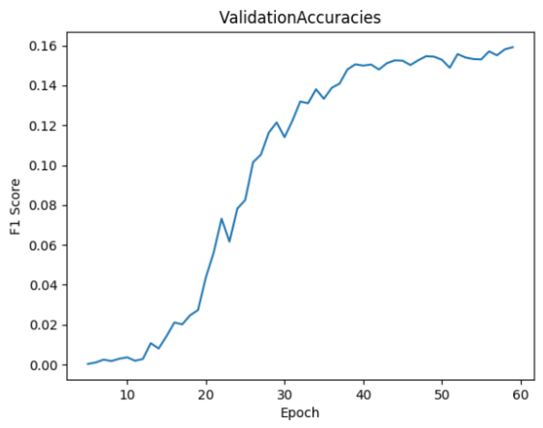
The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM[20] optimizer with momentum 0.9 and batch size 4. The learning rate is set to 5e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

Performance Plots and Results

Below are the training loss and validation accuracies during the training of the model.



(a)



(b)

Figure 3.6: Training Statistics for BBC News Dataset (a) Training Loss vs epochs, & (b) validation Accuracy vs epochs

The performance metrics Precision, Recall and F1 score calculated on the test set are tabulated below.

Precision	Recall	F1 score
0.7431	0.1403	0.2360

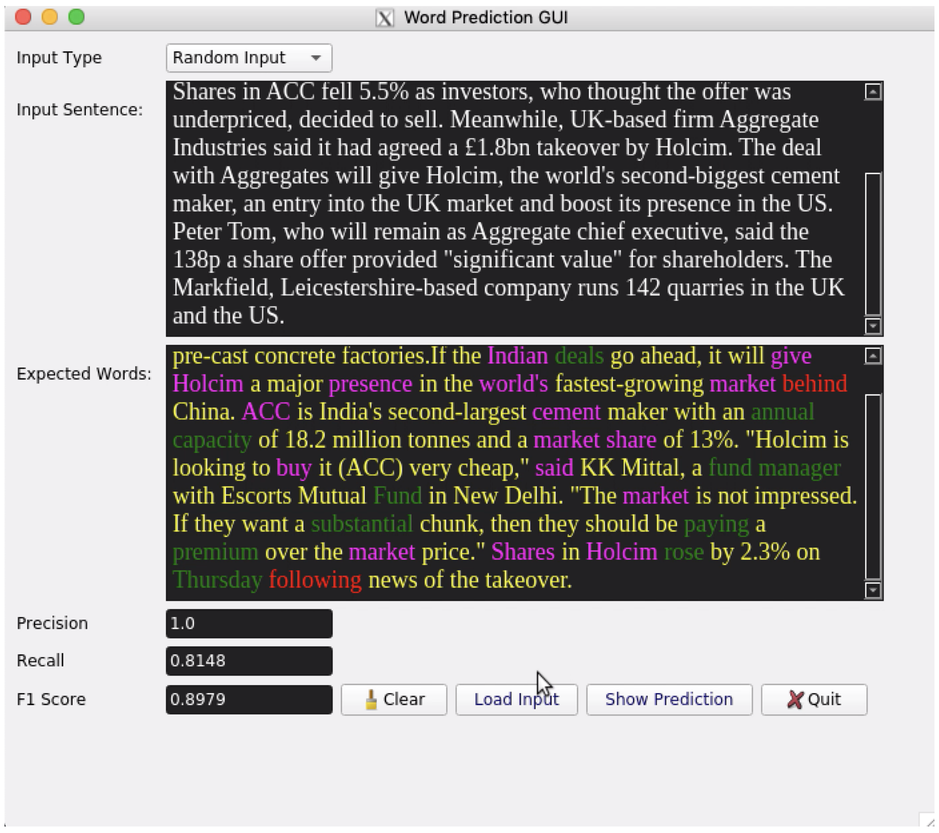
Table 3.4: Word Prediction results summary on BBC News dataset: The first column represent what proportion of positive predictions were actually correct i.e., precision [3.6] and second column represents what proportion of actual positives were identified correctly i.e., recall [3.7]. Third column represents the harmonic mean of precision and recall

Example 4. *Source text represents the input text, Target represents the target text. Below is a business article about Swiss cement firm from BBC News articles dataset[14].*

Source Text : *Swiss cement firm Holcim has bid $800m (£429m) to buy two Indian cement firms and a holding company in the country. It plans to buy Associated Cement Companies (ACC), Ambuja Cement Eastern and the holding firm, Ambuja Cement India Ltd, a Holcim statement said. Shares in ACC fell 5.5% as investors, who thought the offer was underpriced, decided to sell. Meanwhile, UK-based firm Aggregate Industries said it had agreed a £1.8bn takeover by Holcim.Âă The deal with Aggregates will give Holcim, the world's second-biggest cement maker, an entry into the UK market and boost its presence in the US. Peter Tom, who will remain as Aggregate chief executive, said the 138p a share offer provided "significant value" for shareholders. The Markfield, Leicestershire-based company runs 142 quarries in the UK and the US.*

Target Text : *It also has 164 ready-mixed concrete plants , 90 asphalt plants and 32 pre - cast concrete factories . If the Indian deals go ahead , it will give Holcim a major presence in the world's fastest - growing market behind China . ACC is India's second-largest cement maker with an annual capacity of 18.2 million tonnes and a market share of 13% . "Holcim is looking to buy it (ACC) very cheap ," said KK Mittal , a fund manager with Escorts Mutual Fund in New Delhi . The* *market is not impressed. If they want a substantial chunk, then they should be paying a premium over the market price. Shares in Holcim rose by 2.3% on Thursday following news of the takeover.*

Here the model achieves an F1 score of 0.8979 with Precision of 1.0 and Recall of 0.8148.



Input Type: Random Input

Input Sentence: Shares in ACC fell 5.5% as investors, who thought the offer was underpriced, decided to sell. Meanwhile, UK-based firm Aggregate Industries said it had agreed a £1.8bn takeover by Holcim. The deal with Aggregates will give Holcim, the world's second-biggest cement maker, an entry into the UK market and boost its presence in the US. Peter Tom, who will remain as Aggregate chief executive, said the 138p a share offer provided "significant value" for shareholders. The Markfield, Leicestershire-based company runs 142 quarries in the UK and the US.

Expected Words: pre-cast concrete factories.If the Indian deals go ahead, it will give Holcim a major presence in the world's fastest-growing market behind China. ACC is India's second-largest cement maker with an annual capacity of 18.2 million tonnes and a market share of 13%. "Holcim is looking to buy it (ACC) very cheap," said KK Mittal, a fund manager with Escorts Mutual Fund in New Delhi. "The market is not impressed. If they want a substantial chunk, then they should be paying a premium over the market price." Shares in Holcim rose by 2.3% on Thursday following news of the takeover.

Precision: 1.0

Recall: 0.8148

F1 Score: 0.8979

Figure 3.7: GUI Demo showing the word predictions for above example.

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction. Even though the model's prediction are good, the number of such predictions on the test set are less resulting in low recall. As a result of this low recall, the model's F1 score is still 0.2360 even though the Precision is good.

3.5.2 Wikipedia Dataset

Unlike BBC News articles, the Wikipedia dataset articles span across a variety of genres meaning the distributions are not uniform. It was highly challenging to train the Wikipedia dataset. We randomly sampled 25000 articles from Wikipedia and made random 2500 articles as validation, and random 2500 articles as test set. We used 2048 hidden units of Bidirectional GRU of the encoder to get a better representations of the input text. The number of sentences in the source and target text are limited at 15 sentences each. Also, for each word in the target we made sure they have minimum frequency of 25 and maximum frequency of 100. This is to make sure that the model has enough samples to learn from as Wikipedia is a highly diversified dataset.

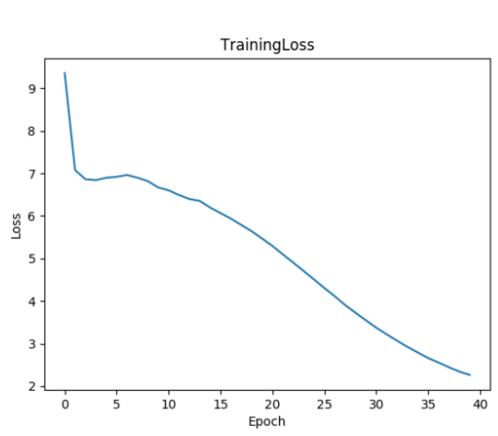
Model Training and Optimization

The network contains a 2048 hidden unit Bidirectional GRU for encoding the input text. The model is trained for 40 epochs on Nvidia 1080 Ti GPU.

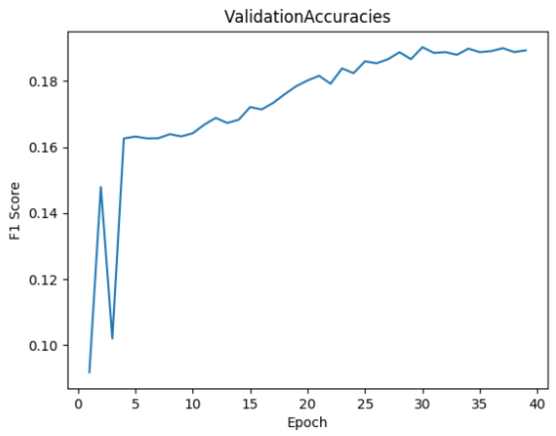
The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM[20] optimizer with momentum 0.9 and batch size 16. The learning rate is set to 3e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

Performance Plots and Results

Below are the training loss and validation accuracies during the training of the model.



(a)



(b)

Figure 3.8: Training Statistics for Wikipedia Dataset Training (a) Training Loss vs epochs, & (b) validation Accuracy vs epochs

We can notice that the training loss starts at a higher value for Wikipedia and ends at lower value for BBC news articles due to difference in dataset sizes. The performance metrics Precision, Recall and F1 score calculated on the test set are tabulated below.

Precision	Recall	F1 score
0.7969	0.3417	0.4783

Table 3.5: Word Prediction result summary on Wikipedia dataset: The first column represent what proportion of positive predictions were actually correct i.e., precision [3.6] and second column represents what proportion of actual positives were identified correctly i.e., recall [3.7]. Third column represents the harmonic mean of precision and recall

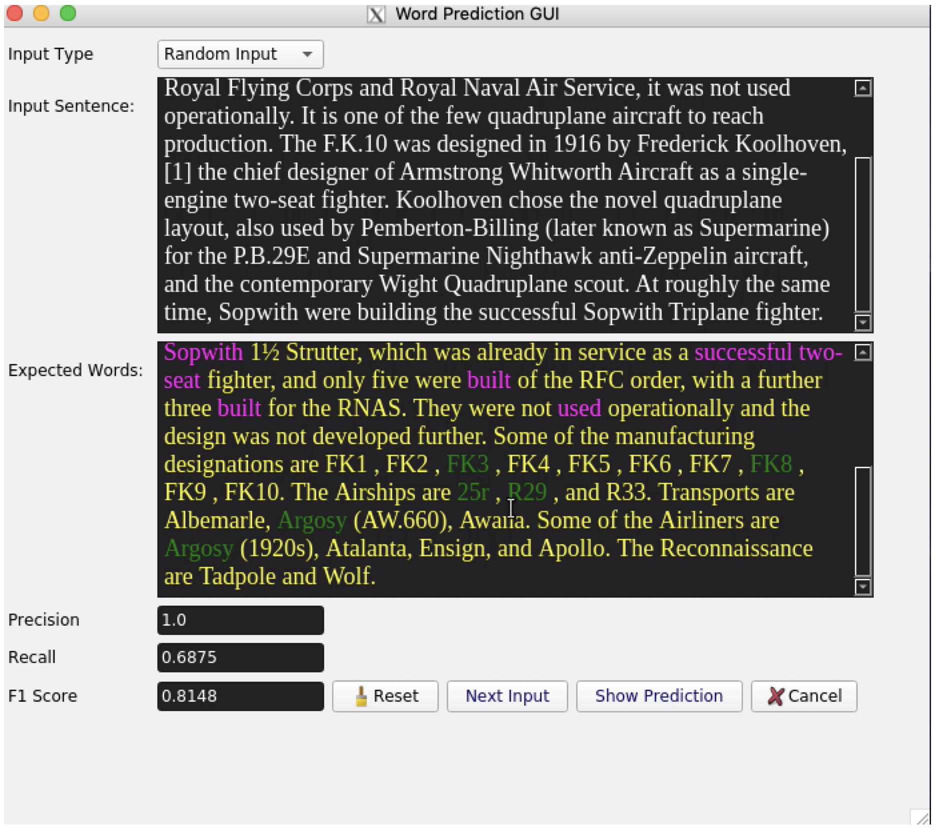
Example 5. *Below is an article on fighter plane from Wikipedia dataset [15]*

Source text : *The Armstrong Whitworth FK10 was a British two-seat quadruplane (i.e., four wing) fighter aircraft built by Armstrong Whitworth during the First World War. While it was ordered in small numbers for the Royal Flying Corps and Royal Naval Air Service, it was not used oper-*

ationally. It is one of the few quadruplane aircraft to reach production. The FK10 was designed in 1916 by Frederick Koolhoven, the chief designer of Armstrong Whitworth Aircraft as a single-engine two-seat fighter. Koolhoven chose the novel quadruplane layout, also used by Pemberton-Billing (later known as Supermarine) for the P.B.29E and Supermarine Nighthawk anti-Zeppelin aircraft, and the contemporary Wight Quadruplane scout. At roughly the same time, Sopwith were building the successful Sopwith Triplane fighter.

Target text: *The first prototype , the FK9 was built and first flown in the summer of 1916 , powered by a 110 hp (80 kW) Clerget 9Z engine . It had a shallow fuselage , with the wings joined by plank-like interplane struts , similar to those used by the Sopwith Triplane . In late 1916 , a production order for 50 was placed by the RFC for a modified version , the FK10 . The production FK10 had a new , deeper fuselage , and a new tail , but retained the wing planform of the FK9 . The FK10 showed inferior performance to the Sopwith 1Â¡ Strutter , which was already in service as a successful two-seat fighter , and only five were built of the RFC order , with a further three built for the RNAS . They were not used operationally and the design was not developed further . Some of the manufacturing designations are FK1 , FK2 , FK3 , FK4 , FK5 , FK6 , FK7 , FK8 , FK9 , FK10 . The Airships are 25r , R29 , and R33 . Transports are Albemarle , Argosy (AW.660), Awana . Some of the Airliners are Argosy (1920s), Atalanta , Ensign , and Apollo . The Reconnaissance are Tadpole and Wolf .*

Here the model achieves an F1 score of 0.8148 with Precision of 1.0 and Recall of 0.6875.


Word Prediction GUI

Input Type: Random Input

Input Sentence: Royal Flying Corps and Royal Naval Air Service, it was not used operationally. It is one of the few quadruplane aircraft to reach production. The F.K.10 was designed in 1916 by Frederick Koolhoven, [1] the chief designer of Armstrong Whitworth Aircraft as a single-engine two-seat fighter. Koolhoven chose the novel quadruplane layout, also used by Pemberton-Billing (later known as Supermarine) for the P.B.29E and Supermarine Nighthawk anti-Zeppelin aircraft, and the contemporary Wight Quadruplane scout. At roughly the same time, Sopwith were building the successful Sopwith Triplane fighter.

Expected Words: Sopwith 1½ Strutter, which was already in service as a successful two-seat fighter, and only five were built of the RFC order, with a further three built for the RNAS. They were not used operationally and the design was not developed further. Some of the manufacturing designations are FK1 , FK2 , FK3 , FK4 , FK5 , FK6 , FK7 , FK8 , FK9 , FK10. The Airships are 25r , R29 , and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.

Precision: 1.0

Recall: 0.6875

F1 Score: 0.8148

Figure 3.9: GUI Demo showing the word predictions for above example.

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction. Because of the dataset pruning (increased minimum frequency for target words), the number of target words for each input are relatively less compared to BBC News dataset. This along with increase in hidden unit size gave an overall better recall, which helped achieve better F1 score than BBC News dataset.

3.6 Memory capability

Every neural network has its own limitations on the memory capability. This is analogous to how human brain functions, we learn a lot of things but we don't remember all of them. Sometimes we remember the whole thing we learnt, sometimes only important things while sometimes nothing. In this part of research, we explored how well the network can remember the data we used to train i.e., doing predictions on seen data.

For a given neural network, lesser the data used to train higher the remembrance power. But to what extent can the network remember data? This is what we have explored.
When the data samples increase the network can not remember all of them and slowly the performance is expected to degrade. After a certain point, the network cannot remember or generalize and hence the prediction accuracy will tend towards zero. During these experiments the training, validation and testing datasets are made to be exactly identical as we want to test how well the network can remember. We modelled this problem as sequence to multi-label classification as earlier. In the output, we are not worried about the order of the words and hence the metrics Precision [3.6], recall [3.7] and F1 [3.8] Scores are our default choice. All these metrics can have a minimum value of 0 and maximum value of 1.

We employed a completely different strategy for training the neural network for this purpose. Firstly, the training , validation and testing data are same. Secondly, the network is trained for a very large number of epochs or until training loss becomes zero which ever is achieved first. We trained one model for a given number of training samples and hence we have multiple models for comparison on a given dataset. The experiments were carried on two types of datasets i.e., BBC News as well as Wikipedia datasets. The network training details and results are explained in the sections below:

3.6.1 BBC News Dataset

Model Training and Optimization

Different models are trained with training data of sizes 1, 2, 5, 10, 25, 50, 100, 200, 250, 500, 750, 1500, 2000 and 2225. Each model is trained for 100 epochs so that the training loss becomes almost 0. The Performance measured is averaged over all the which were samples used to train the model.

The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM optimizer with momentum 0.9 and batch size 4. The learning rate is set to 5e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

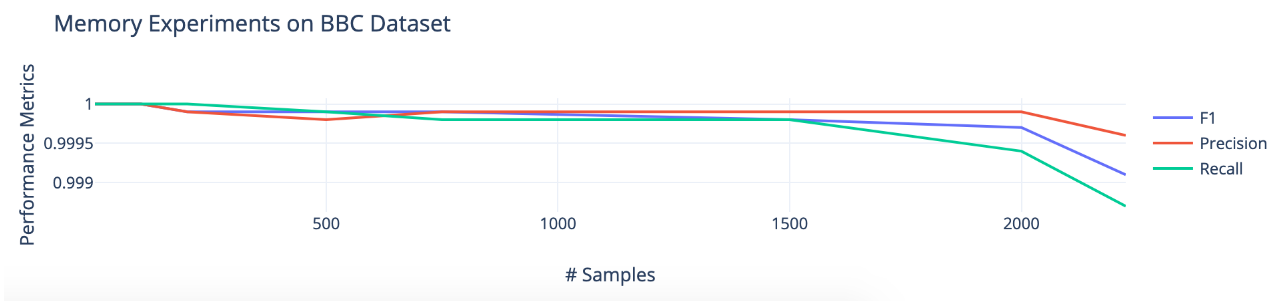


Figure 3.10: Effect of increasing data samples (BBC dataset) on the Performance Metrics. Y-axis represent the metrics Precision, Recall and F1-Score (= Harmonic Mean(Precision, Recall)) while X-axis represent the number of samples used to train the network

Since the size BBC[14] dataset is small, we can clearly understand that the model trained with 2225 articles achieves an F1 score of 0.9996. The model certainly has the capability of remembering the entire training data of 2225 articles.

Example 6. *The remembrance capability of the model (trained with 2225 articles) on a business article titled "India power shares jump on debut" from BBC News articles dataset[14]*

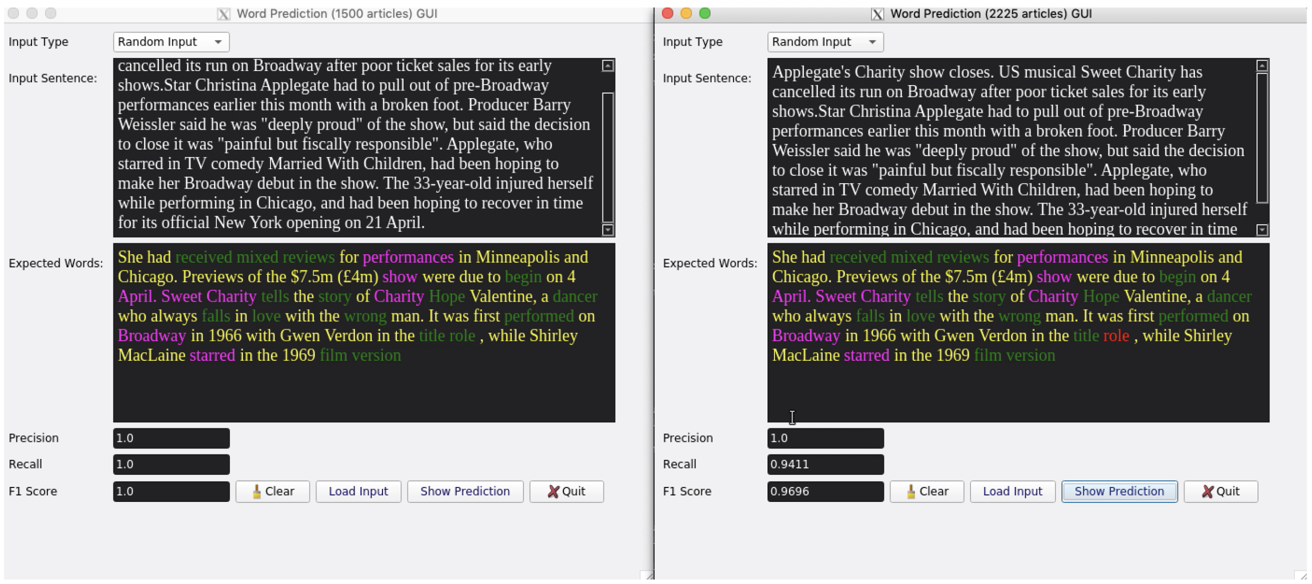
Source text : *US musical Sweet Charity has cancelled its run on Broadway after poor ticket sales for its early shows.*
Star Christina Applegate had to pull out of pre-Broadway performances earlier this month with a broken foot. Producer Barry Weissler said he was "deeply proud" of the show, but said the decision to close it was "painful but fiscally responsible". Applegate, who starred in TV comedy Married With Children, had been hoping to make her Broadway debut in the show. The 33-year-old injured herself while performing in Chicago, and had been hoping to recover in time for its official New York opening on 21 April.

Predictions of model trained with 1500 articles : *She had received mixed reviews for performances in Minneapolis and Chicago. Previews of the $7.5m (£4m) show were due to begin on 4 April. Sweet Charity tells the story of Charity Hope Valentine, a dancer who always falls in love with the wrong man. It was first performed on Broadway in 1966 with Gwen Verdon in the title role, while Shirley MacLaine starred in the 1969 film version*

The model achieved an F1 score of 1.0 with Precision of 1.0 and Recall of 1.0

Predictions of model trained with 2225 articles : *She had received mixed reviews for performances in Minneapolis and Chicago. Previews of the $7.5m (£4m) show were due to begin on 4 April. Sweet Charity tells the story of Charity Hope Valentine, a dancer who always falls in love with the wrong man. It was first performed on Broadway in 1966 with Gwen Verdon in the title role, while Shirley MacLaine starred in the 1969 film version*

The model achieved an F1 score of 0.9696 with Precision of 1.0 and Recall of 0.9411


Word Prediction (1500 articles) GUI

Input Type: Random Input

Input Sentence:

cancelled its run on Broadway after poor ticket sales for its early shows.Star Christina Applegate had to pull out of pre-Broadway performances earlier this month with a broken foot. Producer Barry Weissler said he was "deeply proud" of the show, but said the decision to close it was "painful but fiscally responsible". Applegate, who starred in TV comedy Married With Children, had been hoping to make her Broadway debut in the show. The 33-year-old injured herself while performing in Chicago, and had been hoping to recover in time for its official New York opening on 21 April.

Expected Words:

She had received mixed reviews for performances in Minneapolis and Chicago. Previews of the $7.5m (£4m) show were due to begin on 4 April. Sweet Charity tells the story of Charity Hope Valentine, a dancer who always falls in love with the wrong man. It was first performed on Broadway in 1966 with Gwen Verdon in the title role , while Shirley MacLaine starred in the 1969 film version

Precision: 1.0

Recall: 1.0

F1 Score: 1.0

Word Prediction (2225 articles) GUI

Input Type: Random Input

Input Sentence:

Applegate's Charity show closes. US musical Sweet Charity has cancelled its run on Broadway after poor ticket sales for its early shows.Star Christina Applegate had to pull out of pre-Broadway performances earlier this month with a broken foot. Producer Barry Weissler said he was "deeply proud" of the show, but said the decision to close it was "painful but fiscally responsible". Applegate, who starred in TV comedy Married With Children, had been hoping to make her Broadway debut in the show. The 33-year-old injured herself while performing in Chicago, and had been hoping to recover in time

Expected Words:

She had received mixed reviews for performances in Minneapolis and Chicago. Previews of the $7.5m (£4m) show were due to begin on 4 April. Sweet Charity tells the story of Charity Hope Valentine, a dancer who always falls in love with the wrong man. It was first performed on Broadway in 1966 with Gwen Verdon in the title role , while Shirley MacLaine starred in the 1969 film version

Precision: 1.0

Recall: 0.9411

F1 Score: 0.9696

Figure 3.11: GUI comparison showing the word predictions for different models. The GUI on the left shows the model predictions when trained with 1500 articles and the right shows the model predictions when trained with 2225 articles

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction.

3.6.2 Wikipedia Dataset

Model Training and Optimization

Different models are trained with training data of sizes 1, 2, 5, 25, 100, 500, 1000, 5000, 10000, 15000, 25000, 40000 and 50000. Each model is trained for 150 epochs so that the training loss becomes almost 0. The Performance measured is averaged over all the which were samples used to train the model. The training of model with 50000 samples took 4 days. We couldn't experiment further after 50000 training data due to memory and runtime constraints.

The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM optimizer with momentum 0.9 and batch size 8. The learning rate is set to 1e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

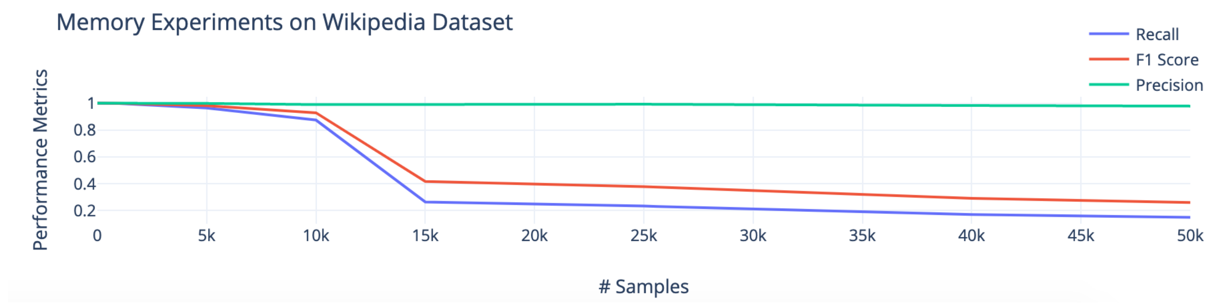


Figure 3.12: Effect of increasing data samples (Wikipedia dataset) on the Performance Metrics. Y-axis represent the metrics Precision, Recall and F1-Score (= Harmonic Mean(Precision, Recall) while the X-axis represent number of samples used to train the network

From the above result, we can see that after 10000 samples the recall metric starts to degrade and hence the F1 score. The recall score becomes 15% when the number of samples increases to 50K.

Example 7. *Below is an example article to illustrate the performance of memory capability of the model when trained with 1000, 15000 samples and 50000 samples. Below is an article about bomber aircraft Armstrong Whitworth from Wikipedia dataset[15].*

Source text : *The Armstrong Whitworth AW.23 was a prototype bomber/transport aircraft produced to specification C.26/31 for the British Air Ministry by Armstrong Whitworth Aircraft. While it was not selected to meet this specification, it did form the basis of the later Armstrong Whitworth Whitley aircraft. Specification C.26/31 required a dual-purpose bomber/transport air-*

craft for service with the Royal Air Force (RAF), with the specification stressing the transport part of its role. The AW.23 was designed by John Lloyd, chief designer of Armstrong Whitworth to meet this specification, competing with the Handley Page HP.51 and the Bristol Bombay. The AW.23 was a low-wing twin-engine monoplane, powered by two Armstrong Siddeley Tiger engines. The aircraft's wings used a novel structure, patented by Armstrong Whitworth, which used a massive light alloy box-sparbraced internally with steel tubes. This structure was extremely strong but required a thick wing section, increasing drag.

Predictions of model trained with 1K articles : *This wing structure was re-used in Armstrong Whitworth's Whitley bomber. The AW.23 was the first Armstrong Whitworth Aircraft to be fitted with a retractable undercarriage. A single prototype, K3585, was built first flying on 4 June 1935. Owing to its unreliable Tiger engines, its delivery to the RAF for testing was delayed, with the Bombay being declared the winner of the specification. The prototype was given the civil registration G-AFRX in May 1939 being used for inflight refuelling development by Flight Refuelling Ltd who used it with the Short Empire flying boat. Data from The British Bomber since 1914[2] General characteristics Performance Armament . FK1, FK2, FK3 , FK4 , FK5, FK6, FK7, FK8, FK9, FK10 . The Airships are 25r, R29, and R33 . Transports are Albemarle, Argosy (AW.660), Awana . Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf .*

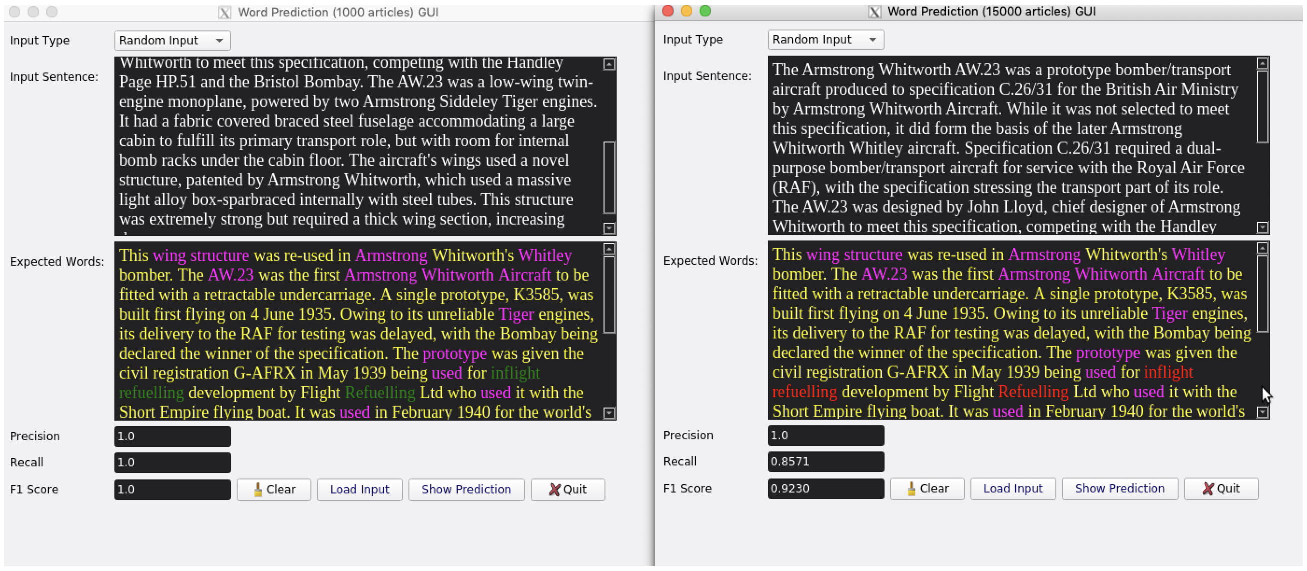
The model achieved an F1 score of 1.0 with Precision of 1.0 and Recall of 1.0

Predictions of model trained with 15K articles : *This wing structure was re-used in Armstrong Whitworth's Whitley bomber. The AW.23 was the first Armstrong Whitworth Aircraft to be fitted with a retractable undercarriage. A single prototype, K3585, was built first flying on 4 June 1935. Owing to its unreliable Tiger engines, its delivery to the RAF for testing was delayed, with the Bombay being declared the winner of the specification. The prototype was given the civil registration G-AFRX* *in May 1939 being used for inflight refuelling development by Flight Refuelling Ltd who used it with the Short Empire flying boat. Data from The British Bomber since 1914[2] General characteristics Performance Armament. FK1, FK2, FK3, FK4, FK5, FK6, FK7, FK8, FK9, FK10. The Airships are 25r, R29, and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.*

The model achieved an F1 score of 0.9230 with Precision of 1.0 and Recall of 0.8571

Predictions of model trained with 50K articles : *This wing structure was re-used in Armstrong Whitworth's Whitley bomber. The AW.23 was the first Armstrong Whitworth Aircraft to be fitted with a retractable undercarriage. A single prototype, K3585, was built first flying on 4 June 1935. Owing to its unreliable Tiger engines, its delivery to the RAF for testing was delayed, with the Bombay being declared the winner of the specification. The prototype was given the civil registration G-AFRX in May 1939 being used for inflight refuelling development by Flight Refuelling Ltd who used it with the Short Empire flying boat. Data from The British Bomber since 1914[2] General characteristics Performance Armament. FK1, FK2, FK3, FK4, FK5, FK6, FK7, FK8, FK9, FK10. The Airships are 25r, R29, and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.*

The model achieved an F1 score of 0.1308 with Precision of 1.0 and Recall of 0.07


Word Prediction (1000 articles) GUI
Clear
Load Input
Show Prediction
Quit
Word Prediction (15000 articles) GUI
Clear
Load Input
Show Prediction
Quit

Input Type: Random Input

Input Sentence: Whitworth to meet this specification, competing with the Handley Page HP.51 and the Bristol Bombay. The AW.23 was a low-wing twin-engine monoplane, powered by two Armstrong Siddeley Tiger engines. It had a fabric covered braced steel fuselage accommodating a large cabin to fulfill its primary transport role, but with room for internal bomb racks under the cabin floor. The aircraft's wings used a novel structure, patented by Armstrong Whitworth, which used a massive light alloy box-sparbraced internally with steel tubes. This structure was extremely strong but required a thick wing section, increasing

Expected Words: This wing structure was re-used in Armstrong Whitworth's Whitley bomber. The AW.23 was the first Armstrong Whitworth Aircraft to be fitted with a retractable undercarriage. A single prototype, K3585, was built first flying on 4 June 1935. Owing to its unreliable Tiger engines, its delivery to the RAF for testing was delayed, with the Bombay being declared the winner of the specification. The prototype was given the civil registration G-AFRX in May 1939 being used for inflight refuelling development by Flight Refuelling Ltd who used it with the Short Empire flying boat. It was used in February 1940 for the world's

Precision: 1.0

Recall: 1.0

F1 Score: 1.0

Input Type: Random Input

Input Sentence: The Armstrong Whitworth AW.23 was a prototype bomber/transport aircraft produced to specification C.26/31 for the British Air Ministry by Armstrong Whitworth Aircraft. While it was not selected to meet this specification, it did form the basis of the later Armstrong Whitworth Whitley aircraft. Specification C.26/31 required a dual-purpose bomber/transport aircraft for service with the Royal Air Force (RAF), with the specification stressing the transport part of its role. The AW.23 was designed by John Lloyd, chief designer of Armstrong Whitworth to meet this specification, competing with the Handley

Expected Words: This wing structure was re-used in Armstrong Whitworth's Whitley bomber. The AW.23 was the first Armstrong Whitworth Aircraft to be fitted with a retractable undercarriage. A single prototype, K3585, was built first flying on 4 June 1935. Owing to its unreliable Tiger engines, its delivery to the RAF for testing was delayed, with the Bombay being declared the winner of the specification. The prototype was given the civil registration G-AFRX in May 1939 being used for inflight refuelling development by Flight Refuelling Ltd who used it with the Short Empire flying boat. It was used in February 1940 for the world's

Precision: 1.0

Recall: 0.8571

F1 Score: 0.9230

Figure 3.13: GUI comparison showing the word predictions for different models on wikipedia dataset. The GUI on the left shows the model predictions when trained with 1K articles and the right shows the model predictions when trained with 15K articles

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction.

As we can see from above comparison, when the model was trained on more samples the model looses its generalization ability and results in poor recall. The model trained with 50000 examples was not able to make more predictions leading to poor recall which implies that the network is not able to remember.

3.7 Noise Tolerance

The technique of addition of Noise while training the model has proven improved generalization in the literature[21]. Infact the noise added will help improve back propagation training [22] as it acts like a regularization technique[23]. However, one has multiple options for introducing noise into the model like adding noise to inputs, weights or to the output. Study has shown adding noise at the input better generalization ability [21] compared to others. However, addition of any level of noise does not help the model. Right amount of noise is again a hyperparameter to choose from. In this part of research, we analyzed the effect of various levels of noise on Model's performance.

As all the training, validation and testing datasets we use while developing a model doesn't contain any noise. For building such robust model, noise has to be introduced artificially so that the model is fit for much more practical applications where input can possibly have noise. The Noise layer is responsible for inducing noise sampled randomly from a normal distribution. We changed the model architecture slightly by introducing an additional Noise layer at the input end after the Embedding layer. The purpose of introducing this layer immediately after the input embedding layer is that the pre-trained embedding we get from open source models like Glove[24] or Word2Vec[5] are not perfect and we want the model to be robust to noises at the input level. The noise layer randomly adds values from a normal distribution of 0 mean and a given standard deviation. The more the standard deviation, the higher the impact of noise on the embeddings. Usually lower level of noises act as a level of regularization but higher levels distort the performance of model and infact the model needs much more training to achieve the results same as the one without noise. We studied the effect of noise with 3 levels of standard deviation i.e., 0.1 , 0.2 and 0.3.

Implementation of Noise Layer

The Noise layer is nothing enabled only during the training phase. This Noise layer is implemented using *normal_* function from Pytorch[18]. It adds random samples, of size equal to the input, from the normal distribution with 0 mean and given standard deviation. The pytorch implementation of the noise layer is as shown below:

$$sampled\_noise = torch.zeros(*input.size()).normal\_(mean = 0, std = sigma) \tag{3.11}$$

$$return\ \ input + sampled\_noise \tag{3.12}$$

Model Architecture

The updated model with Noise layer is as shown in the below figure:

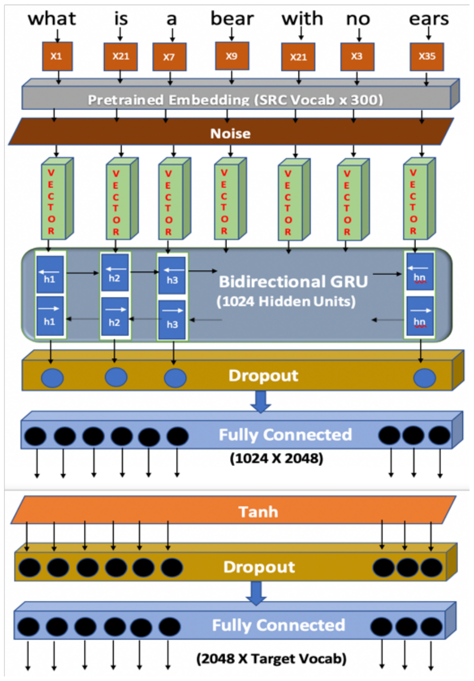


Figure 3.14: Model Architecture with Additional Noise Layer after Pretrained Embedding layer

Below is an example to illustrate how different levels of noise effect the word embeddings by computing Euclidean (L2) distance between new and old embeddings.

Example 8. *For simplicity, let us assume that the word "cat" is represented using 3 dimensional embedding (Vector) as follows* $[1.0755, -2.1031, -0.2598]$. *Let us examine what happens when noise of different standard deviations are added to the vector:*

Case 1: Mean = 0 , Standard deviation = 0.1

New Vector: $[1.1051, -2.1006, -0.2907]$

L2 Distance between the vectors (old and new) is 0.0429

Case 2: Mean = 0 , Standard deviation = 0.2

New Vector: $[0.8570, -1.9747, -0.3511]$

L2 Distance between the vectors (old and new) is 0.2693

Case 3: Mean = 0 , Standard deviation = 0.3

New Vector: $[1.2537, -2.3850, -0.5183]$

L2 Distance between the vectors (old and new) is 0.4219

The vector in 3-Dimensional space has moved much farther as the standard deviation of noise increases. Since our embedding dimension in actual training is 300, the vector could move much further. Also, one interesting aspect to note here is that as we have embedded the noise layer into our model the word embedding of any word being passed to Encoder layer will be different in each epoch. As the noise generated is random, the new embedding vector will also be different. This kind of acts as data augmentation when the noise levels are low and hence the fact that low level of noise acts as regularizer[23] for network.

3.7.1 BBC News Dataset

Model Training and Optimization

Since the noise generated is random, we trained 2 models at each noise level and the metrics are averaged. Each model is trained for 40 epochs.

The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM[20] optimizer with momentum 0.9 and batch size 4. The learning rate is set to 5e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

Performance Plots

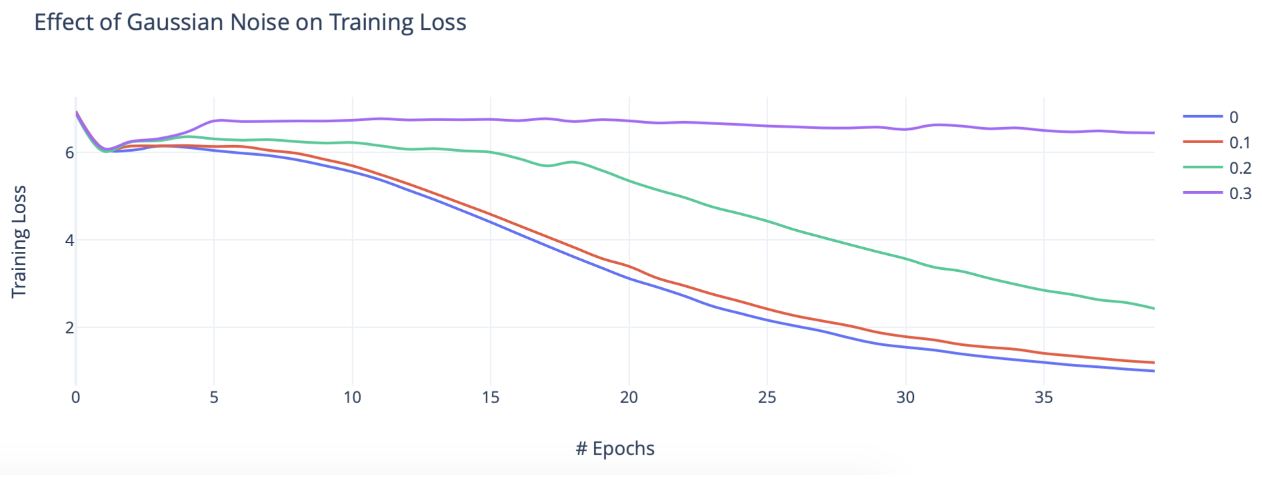


Figure 3.15: Effect of Noise on Training Loss (BBC dataset) : Y-axis showing the training loss and X-axis is number of epochs.

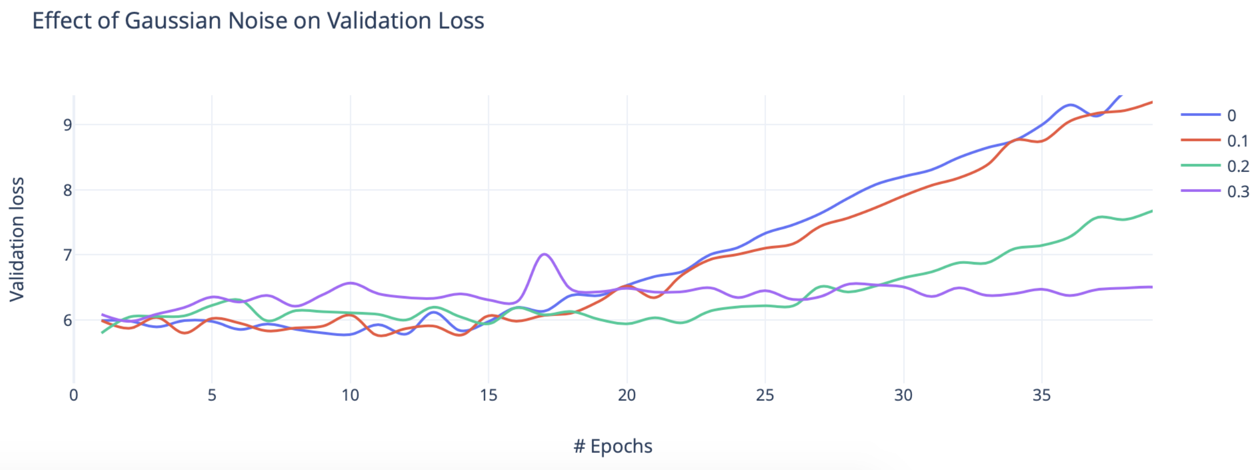


Figure 3.16: Effect of Noise on Validation Loss (BBC dataset) : Y-axis showing the validation loss and X-axis is number of epochs.

As the level of noise increases, the training loss decreases very slowly. At the end of 40 epochs, the model with 0 and 0.1 noise have achieved a loss close to 0 where as the model with 0.2 and 0.3 noise levels have higher levels of loss.

As the level of noise increases, the time taken for the model to overfit increases. With noise, the model sees more data samples as compared to no noise case as the embeddings keep changing at each epoch.

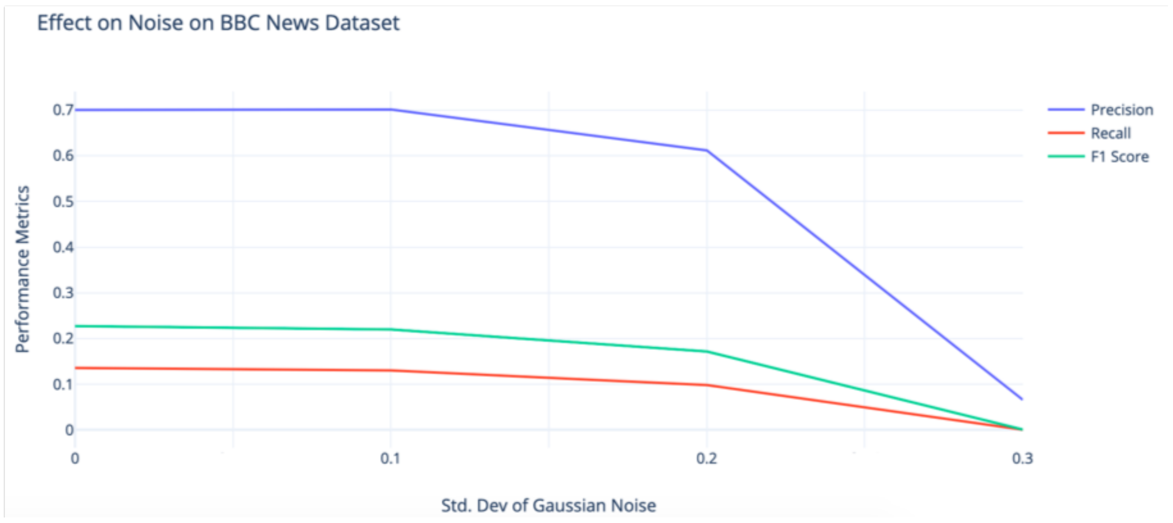


Figure 3.17: Performance Metrics over various Noise levels (BBC dataset): Y-axis shows the Precision, recall and F1 score, X-axis represents the various levels of noise added to the model. Precision represent what proportion of positive predictions were actually correct [3.6] and recall represents what proportion of actual positives were identified correctly [3.7]. F1 score is the harmonic mean of precision and recall

For the noise levels 0 and 0.1, the model's performance is almost similar where as the performance deteriorates in the case 0.2 and 0.3. For the extreme case, the model's performance is worse because it was just trained for 40 epochs (same as other models). In the case of noise, the model needs to be trained with more number of epochs as it sees more samples which implies more data and hence the larger training times.

Below is an example to illustrate the effect of noise of Model's predictions.

Example 9. *Source text represents the input text, Target represents the target text and Prediction1, Prediction2, Prediction3 represent the predictions of the model with 0.1, 0.2 and 0.3 noise levels respectively. The article is about a England team's manager Glenn Hoddle from BBC News articles dataset[14].*

Source text : *Glenn Hoddle will be unveiled as the new Wolves manager on Tuesday. The club have confirmed that the former England coach will be unveiled as the successor to Dave Jones at a news conference at Molineux at 1100 GMT. Hoddle has been linked with a return to former club Southampton but Wolves have won the race for his services. He has been out of the game since being sacked at Spurs in September 2003 and worked alongside Wolves caretaker boss Stuart Gray at Southampton. Hoddle began his managerial career as player-boss with Swindon before moving on to Chelsea and then taking up the England job. His spell in charge of the national side came to an end after the 1998 World Cup when he made controversial remarks about the disabled in a newspaper interview.*

Model predictions when trained with 0.1 standard deviation of noise : *The 47-year-old later returned to management with Southampton, where he again succeeded Jones - as he has now done at Wolves. He engineered an upturn in Saints' fortunes before being lured to White Hart Lane by Tottenham - the club where he made his name as a player. That relationship turned sour at the start of the last campaign and he left the London club early last season. Since then he has applied unsuccessfully for the post of France manager and had also been linked with a return to Southampton. Wolves are currently 17th in the Championship and have a home game against Millwall on Tuesday.*

The model achieved an F1 score of 0.3414 with Precision of 0.3333 and Recall of 0.35

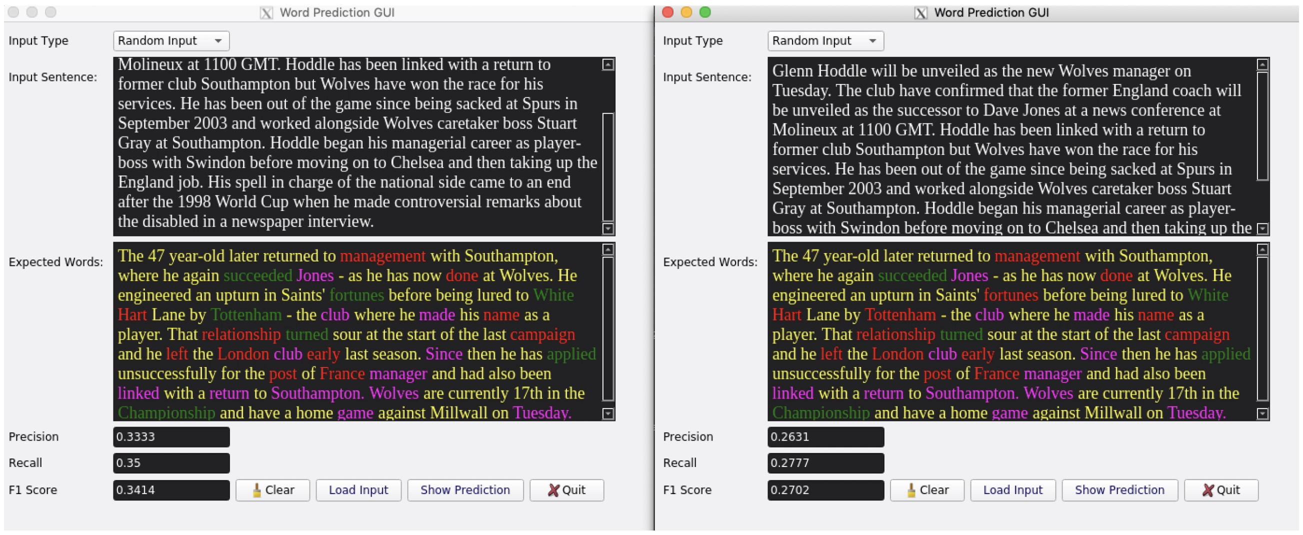
Model predictions when trained with 0.2 standard deviation of noise : *The 47-year-old later returned to management with Southampton, where he again succeeded Jones*

- *as he has now done at Wolves. He engineered an upturn in Saints' fortunes before being lured to White Hart Lane by Tottenham - the club where he made his name as a player. That relationship turned sour at the start of the last campaign and he left the London club early last season. Since then he has applied unsuccessfully for the post of France manager and had also been linked with a return to Southampton. Wolves are currently 17th in the Championship and have a home game against Millwall on Tuesday.*

The model achieved an F1 score of 0.2702 with Precision of 0.2631 and Recall of 0.2777

Model predictions when trained with 0.3 standard deviation of noise : *The 47-year-old later returned to management with Southampton, where he again succeeded Jones - as he has now done at Wolves. He engineered an upturn in Saints' fortunes before being lured to White Hart Lane by Tottenham - the club where he made his name as a player. That relationship turned sour at the start of the last campaign and he left the London club early last season. Since then he has applied unsuccessfully for the post of France manager and had also been linked with a return to Southampton. Wolves are currently 17th in the Championship and have a home game against Millwall on Tuesday.*

The model achieved an F1 score of 0.0 with Precision of 0.0 and Recall of 0.0



Input Type: Random Input

Input Sentence: Molineux at 1100 GMT. Hoddle has been linked with a return to former club Southampton but Wolves have won the race for his services. He has been out of the game since being sacked at Spurs in September 2003 and worked alongside Wolves caretaker boss Stuart Gray at Southampton. Hoddle began his managerial career as player-boss with Swindon before moving on to Chelsea and then taking up the England job. His spell in charge of the national side came to an end after the 1998 World Cup when he made controversial remarks about the disabled in a newspaper interview.

Expected Words: The 47 year-old later returned to management with Southampton, where he again succeeded Jones - as he has now done at Wolves. He engineered an upturn in Saints' fortunes before being lured to White Hart Lane by Tottenham - the club where he made his name as a player. That relationship turned sour at the start of the last campaign and he left the London club early last season. Since then he has applied unsuccessfully for the post of France manager and had also been linked with a return to Southampton. Wolves are currently 17th in the Championship and have a home game against Millwall on Tuesday.

Precision: 0.3333

Recall: 0.35

F1 Score: 0.3414

Input Type: Random Input

Input Sentence: Glenn Hoddle will be unveiled as the new Wolves manager on Tuesday. The club have confirmed that the former England coach will be unveiled as the successor to Dave Jones at a news conference at Molineux at 1100 GMT. Hoddle has been linked with a return to former club Southampton but Wolves have won the race for his services. He has been out of the game since being sacked at Spurs in September 2003 and worked alongside Wolves caretaker boss Stuart Gray at Southampton. Hoddle began his managerial career as player-boss with Swindon before moving on to Chelsea and then taking up the

Expected Words: The 47 year-old later returned to management with Southampton, where he again succeeded Jones - as he has now done at Wolves. He engineered an upturn in Saints' fortunes before being lured to White Hart Lane by Tottenham - the club where he made his name as a player. That relationship turned sour at the start of the last campaign and he left the London club early last season. Since then he has applied unsuccessfully for the post of France manager and had also been linked with a return to Southampton. Wolves are currently 17th in the Championship and have a home game against Millwall on Tuesday.

Precision: 0.2631

Recall: 0.2777

F1 Score: 0.2702

Figure 3.18: GUI comparison showing the word predictions for different models on BBC dataset. The GUI on the left shows the model predictions when induced noise level was 0.1 and the right shows the model predictions when induced noise level was 0.2

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction. From the above example, we can notice that model did not make any correct predictions when the noise level is too high. The model's recall follows a decreasing trend clearly with increase in noise level.

3.7.2 Wikipedia Dataset

Model Training and Optimization

Since the noise generated is random, we trained 2 models at each noise level and the performance metrics are averaged. Each model is trained for 50 epochs.

The MultiLabelSoftMargin loss function for the training is taken as shown in section 3.3. The network is trained end to end, with weights initialized with Xavier initialization [19]. We trained the model with ADAM[20] optimizer with momentum 0.9 and batch size 8. The learning rate is set to 3e-4. The output layer returns the floating values, these and the loss values are required for optimization during training. The whole model is implemented in PyTorch library [18].

Performance Plots

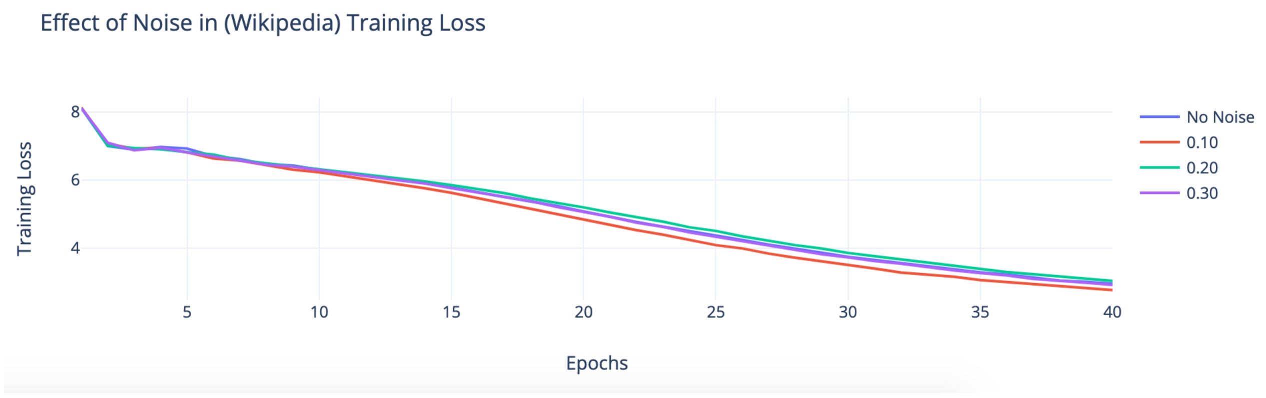


Figure 3.19: Effect of Noise on Training Loss (Wikipedia dataset) : Y-axis showing the validation loss and X-axis is number of epochs.

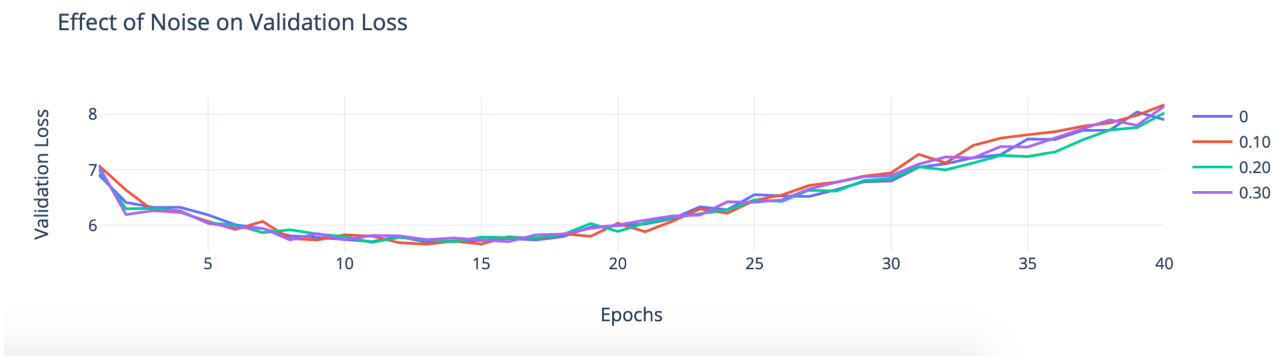


Figure 3.20: Effect of Noise on Validation Loss (Wikipedia dataset) : Y-axis showing the validation loss and X-axis is number of epochs.

As the level of noise increases, the training loss decreases very slowly. These experiments are carried out for 40 epochs to illustrate the effect of noise on the loss. As the level of noise increases, the time taken for the model to overfit increases.

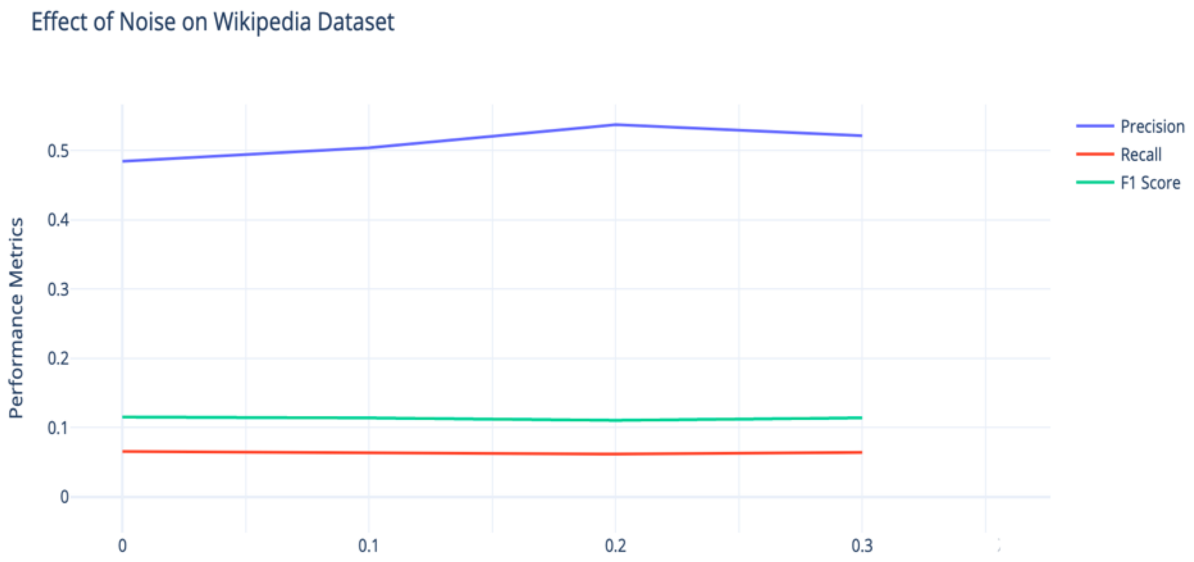


Figure 3.21: Performance Metrics over various Noise levels (Wikipedia dataset): Y-axis shows the Precision, recall and F1 score, X-axis represents the various levels of noise added to the model. Precision represent what proportion of positive predictions were actually correct [3.6] and recall represents what proportion of actual positives were identified correctly [3.7]. F1 score is the harmonic mean of precision and recall

With noise, the model sees more data samples as compared to no noise case as the embeddings keep changing for every epoch. Noise level of standard deviation 0.2 worsens the performance as opposed to 0.1 and 0.3 thresholds.

Example 10. *Source text represents the input text, Target represents the target text and Prediction1, Prediction2, Prediction3 represent the predictions of the model with 0.1, 0.2 and 0.3 noise levels respectively. Below is an article about British Fighter Plane from Wikipedia dataset[15]*

Source Text : *The Armstrong Whitworth FK10 was a British two-seat quadruplane (i.e., four* *wing) fighter aircraft built by Armstrong Whitworth during the First World War. While it was ordered in small numbers for the Royal Flying Corps and Royal Naval Air Service, it was not used operationally. It is one of the few quadruplane aircraft to reach production. The FK10 was designed in 1916 by Frederick Koolhoven, the chief designer of Armstrong Whitworth Aircraft as a single-engine two-seat fighter. Koolhoven chose the novel quadruplane layout, also used by Pemberton-Billing (later known as Supermarine) for the P.B.29E and Supermarine Nighthawk anti-Zeppelin aircraft, and the contemporary Wight Quadruplane scout. At roughly the same time, Sopwith were building the successful Sopwith Triplane fighter.*

Model predictions when trained with 0.1 standard deviation of noise : *The first prototype , the FK9 was built and first flown in the summer of 1916 , powered by a 110 hp (80 kW) Clerget 9Z engine . It had a shallow fuselage , with the wings joined by plank-like interplane struts , similar to those used by the Sopwith Triplane . In late 1916 , a production order for 50 was placed by the RFC for a modified version , the FK10 . The production FK10 had a new , deeper fuselage , and a new tail , but retained the wing planform of the FK9 . The FK10 showed inferior performance to the Sopwith 1Â¡ Strutter , which was already in service as a successful two-seat fighter , and only five were built of the RFC order , with a further three built for the RNAS . They were not used operationally and the design was not developed further . Some of the manufacturing designations are FK1 , FK2 , FK3 , FK4 , FK5 , FK6 , FK7 , FK8 , FK9 , FK10 . The Airships are 25r , R29 , and R33 . Transports are Albemarle , Argosy (AW.660), Awana . Some of the Airliners are Argosy (1920s), Atalanta , Ensign , and Apollo . The Reconnaissance are Tadpole and Wolf .*

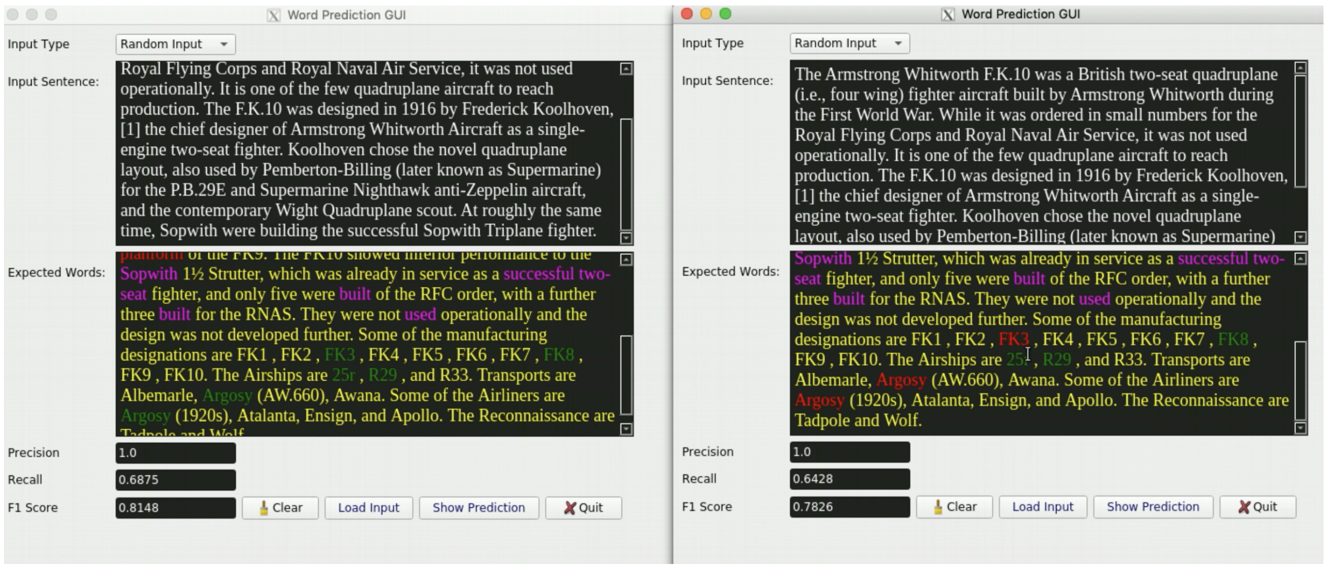
Here the model achieves an F1 score of 0.8148 with Precision of 1.0 and Recall of 0.6875.

Model predictions when trained with 0.2 standard deviation of noise : *The first prototype , the FK9 was built and first flown in the summer of 1916 , powered by a 110 hp (80 kW) Clerget 9Z engine . It had a shallow fuselage , with the wings joined by* *plank-like interplane struts, similar to those used by the Sopwith Triplane. In late 1916, a production order for 50 was placed by the RFC for a modified version, the FK10. The production FK10 had a new, deeper fuselage, and a new tail, but retained the wing planform of the FK9. The FK10 showed inferior performance to the Sopwith 1Â¡ Strutter, which was already in service as a successful two-seat fighter, and only five were built of the RFC order, with a further three built for the RNAS. They were not used operationally and the design was not developed further. Some of the manufacturing designations are FK1, FK2, FK3, FK4, FK5, FK6, FK7, FK8, FK9, FK10. The Airships are 25r, R29, and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.*

Here the model achieves an F1 score of 0.7826 with Precision of 1.0 and Recall of 0.6428.

Model predictions when trained with 0.3 standard deviation of noise : *The first prototype, the FK9 was built and first flown in the summer of 1916, powered by a 110 hp (80 kW) Clerget 9Z engine. It had a shallow fuselage, with the wings joined by plank-like interplane struts, similar to those used by the Sopwith Triplane. In late 1916, a production order for 50 was placed by the RFC for a modified version, the FK10. The production FK10 had a new, deeper fuselage, and a new tail, but retained the wing planform of the FK9. The FK10 showed inferior performance to the Sopwith 1Â¡ Strutter, which was already in service as a successful two-seat fighter, and only five were built of the RFC order, with a further three built for the RNAS. They were not used operationally and the design was not developed further. Some of the manufacturing designations are FK1, FK2, FK3, FK4, FK5, FK6, FK7, FK8, FK9, FK10. The Airships are 25r, R29, and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.*

Here the model achieves an F1 score of 0.8148 with Precision of 1.0 and Recall of 0.6875.



Input Sentence: Royal Flying Corps and Royal Naval Air Service, it was not used operationally. It is one of the few quadruplane aircraft to reach production. The F.K.10 was designed in 1916 by Frederick Koolhoven, [1] the chief designer of Armstrong Whitworth Aircraft as a single-engine two-seat fighter. Koolhoven chose the novel quadruplane layout, also used by Pemberton-Billing (later known as Supermarine) for the P.B.29E and Supermarine Nighthawk anti-Zeppelin aircraft, and the contemporary Wight Quadruplane scout. At roughly the same time, Sopwith were building the successful Sopwith Triplane fighter.

Expected Words: platform of the FK9. The FK10 showed inferior performance to the Sopwith 1½ Strutter, which was already in service as a successful two-seat fighter, and only five were built of the RFC order, with a further three built for the RNAS. They were not used operationally and the design was not developed further. Some of the manufacturing designations are FK1 , FK2 , FK3 , FK4 , FK5 , FK6 , FK7 , FK8 , FK9 , FK10. The Airships are 25r , R29 , and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.

Precision: 1.0
Recall: 0.6875
F1 Score: 0.8148

Input Sentence: The Armstrong Whitworth F.K.10 was a British two-seat quadruplane (i.e., four wing) fighter aircraft built by Armstrong Whitworth during the First World War. While it was ordered in small numbers for the Royal Flying Corps and Royal Naval Air Service, it was not used operationally. It is one of the few quadruplane aircraft to reach production. The F.K.10 was designed in 1916 by Frederick Koolhoven, [1] the chief designer of Armstrong Whitworth Aircraft as a single-engine two-seat fighter. Koolhoven chose the novel quadruplane layout, also used by Pemberton-Billing (later known as Supermarine)

Expected Words: Sopwith 1½ Strutter, which was already in service as a successful two-seat fighter, and only five were built of the RFC order, with a further three built for the RNAS. They were not used operationally and the design was not developed further. Some of the manufacturing designations are FK1 , FK2 , FK3 , FK4 , FK5 , FK6 , FK7 , FK8 , FK9 , FK10. The Airships are 25r , R29 , and R33. Transports are Albemarle, Argosy (AW.660), Awana. Some of the Airliners are Argosy (1920s), Atalanta, Ensign, and Apollo. The Reconnaissance are Tadpole and Wolf.

Precision: 1.0
Recall: 0.6428
F1 Score: 0.7826

Figure 3.22: GUI comparison showing the word predictions for different models on Wikipedia dataset.The GUI on the left shows the model predictions when induced noise level was 0.1 and the right shows the model predictions when induced noise level was 0.2

The words in the target text are highlighted in various colors to understand the predictions. Yellow implies stop words, words with less than 3 characters, high frequency words, low frequency words. Magenta implies words which are common between source and target texts, red implies the words which are not predicted by the model, and green shows model's prediction.

As opposed to BBC News dataset, the effect of noise on Wikipedia dataset is not that significant. The performance metrics even though they are varying the difference is very low. The precision of the model increases slightly due to the addition of noise, however the recall was effected slightly causing an over all decrease in the F1 score. When the noise of standard deviation of 0.2 is added, the model performance is worse as opposed to standard deviation thresholds of 0.1 and 0.3. The effect of 0.1 and 0.3 noise thresholds are almost same.

]

4. SUMMARY AND CONCLUSIONS

We presented an neural approach for word predictions which can not only predict immediate words but also farthest words in the document. We also presented a modified object detection model trained specially for detecting faces with greater accuracy and speed. Our word prediction model is first of its kind to generalize the language model. For the evaluation criteria, the proposed technique can achieve F1 score of 0.48 with a precision of 0.80 and recall of 0.34. We also presented an analysis for effect of Gaussian noise on these models as well as the memory capabilities and observed that training models with addition of some noise helps them produce robust word predictions. Presented techniques are useful for many down stream applications like machine translation, recommender systems etc. With this technique, we modelled a typical language modeling problem as multi-label classification problem. We also developed an easy to use GUI which can extract context information from the user text and show the possible word predictions.

4.1 Challenges

The main challenge in this work to get a decent recall. As the model tries to predict as many words as possible from the rest half of the document. The problem becomes intractable when the document size is too large. Hence we employed techniques lemmatization and TF-IDF [25] to filter words in the expected target.

4.2 Further Study

There is a scope to increase the model's F1 score by further improving the recall. This could be done by making the model focus more on important words and provide suggestion based on the context of these important words. Using multi-head attention mechanism like Transformers or Variational Auto Encoders [26] might help improve the recall.

REFERENCES

[1] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *CoRR*, vol. abs/1703.03130, 2017.

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2000.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.

[4] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, (Santa Cruz, California, USA), pp. 310–318, Association for Computational Linguistics, June 1996.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013.

[6] S. Vajapeyam, "Understanding shannon's entropy metric for information," *CoRR*, vol. abs/1405.2061, 2014.

[7] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *CoRR*, vol. abs/1609.07843, 2016.

[8] C. Gong, D. He, X. Tan, T. Qin, L. Wang, and T. Liu, "FRAGE: frequency-agnostic word representation," *CoRR*, vol. abs/1809.06858, 2018.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[10] B. Krause, E. Kahembwe, I. Murray, and S. Renals, "Dynamic evaluation of transformer language models," *CoRR*, vol. abs/1904.08378, 2019.

[11] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *CoRR*, vol. abs/1508.06615, 2015.

[12] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *CoRR*, vol. abs/1602.02410, 2016.

[13] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, "A neural knowledge language model," *CoRR*, vol. abs/1608.00318, 2016.

[14] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*, pp. 377–384, ACM Press, 2006.

[15] Wikimedia, "Wikimedia downloads," 2018. [Online; open source data dump].

[16] Wikipedia contributors, "Robot — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Robot&oldid=902106168`, 2019. [Online; accessed 20-June-2019].

[17] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[21] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Comput.*, vol. 8, pp. 643–674, Apr. 1996.

[22] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24–38, 1992. Exported from https://app.dimensions.ai on 2019/05/25.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.

[25] J. Ramos, "Using tf-idf to determine word relevance in document queries," 1999.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

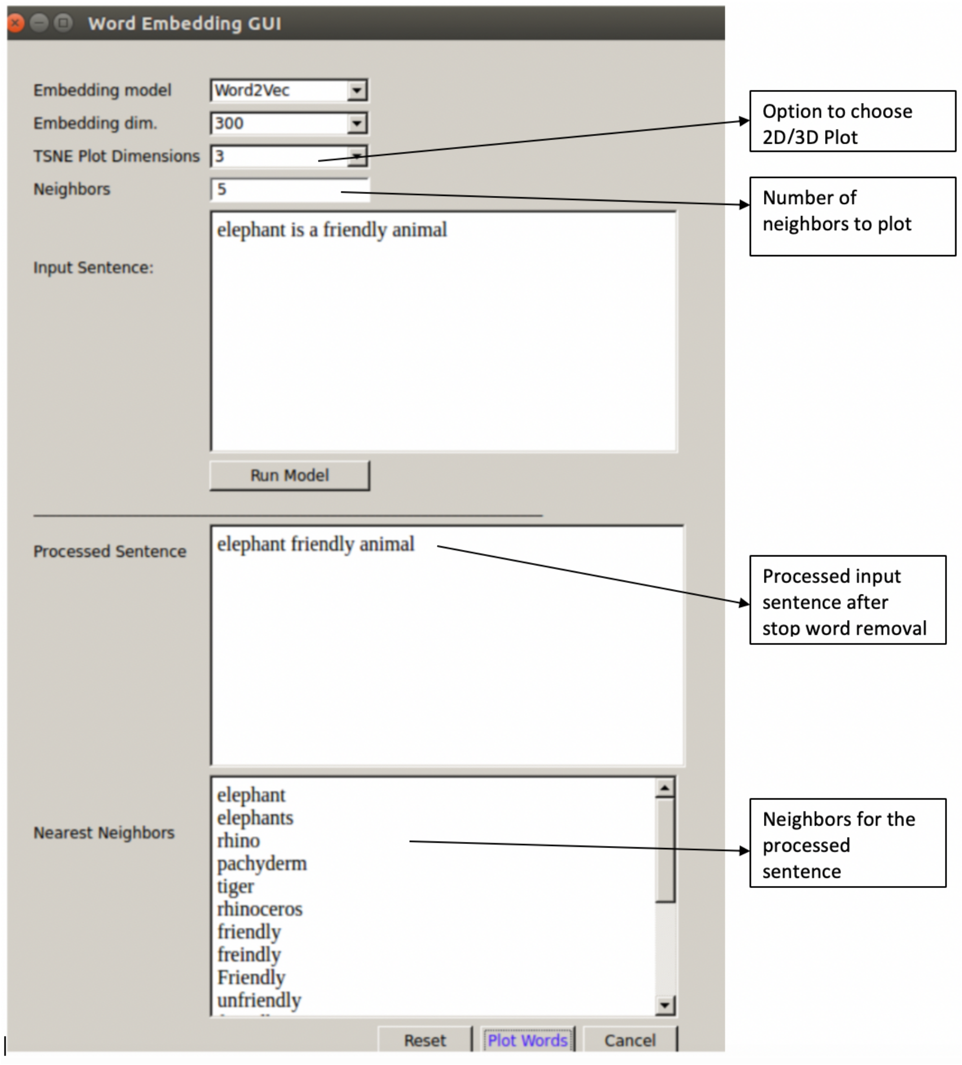
APPENDIX A

VISUALIZATION GUI

This section presents the details for Graphical User Interface developed for visualizing text as a path. Figure A.1 shows the GUI interface and explains the basic structure and functions. The GUI has multiple user options to visualize a given text as a path. User has an option to select either “Glove” or “word2vec” pretrained embeddings to convert the text to vectors. The “Embedding dim” option allows users to select from a list of available dimensions for word embedding. Higher the dimension, the better is the vector representation. We project the high dimensional embedding to 2-D/3-D using t-sne[17] algorithm. We also provided an option to choose the number of neighbors to be plotted along with the actual text.

In the figure we can see that the GUI processed the input sentence by removing the stop words and finding the nearest neighbors in the embedding space. word2vec is provided with a function “most_similar” which returns the top n similar words with their distances as well. However, Glove does not provide such functionality. We had used cosine similarity as a distance measure to calculate the neighbors and get the nearest ones by sorting the distances. As we can see from the below figure A.1, the input sentence is processed and only the non-stop words are remained. The 5 nearest neighbors for each word in the final processed sentence are achieved using the *most_similar* function of Word2vec. The words elephant, elephants, rhino, pachyderm, tiger, rhinoceros are close by in the 300 dimensional embedding space.

Example 8 Let us consider the sentence “elephant is a friendly animal” and see the visualization of this sentence as a path. The GUI output and the 2-D, 3-D representations of the sentence are shown in detail below:


Word Embedding GUI
Run Model
Reset
Plot Words
Cancel
Option to choose 2D/3D Plot
Number of neighbors to plot
Processed input sentence after stop word removal
Neighbors for the processed sentence

Embedding model: Word2Vec
Embedding dim.: 300
TSNE Plot Dimensions: 3
Neighbors: 5
Input Sentence: elephant is a friendly animal

Processed Sentence: elephant friendly animal

Nearest Neighbors:
elephant
elephants
rhino
pachyderm
tiger
rhinoceros
friendly
freindly
Friendly
unfriendly

Figure A.1: Visualization GUI with annotation of various options

The actual words are repre- sented using red markers where as the neighbors with blue. If the processed sentence has 3 words and the user opted for 5 neigbors, then the total number of data points for plotting is 3+3*5 = 18 words. The first 3 are the actual words and the rest 15 words are neighbors. So for a processed sentences with w words and n neighbors will have w*(n+1) words for plotting.

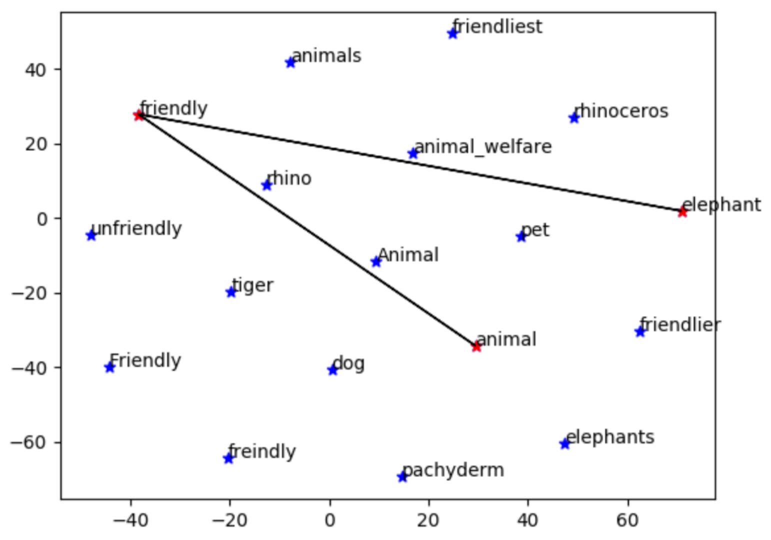


Figure A.2: Visualization of sentence in 2-D using Word2vec, 300 - dimensional vector and 5 neighbors

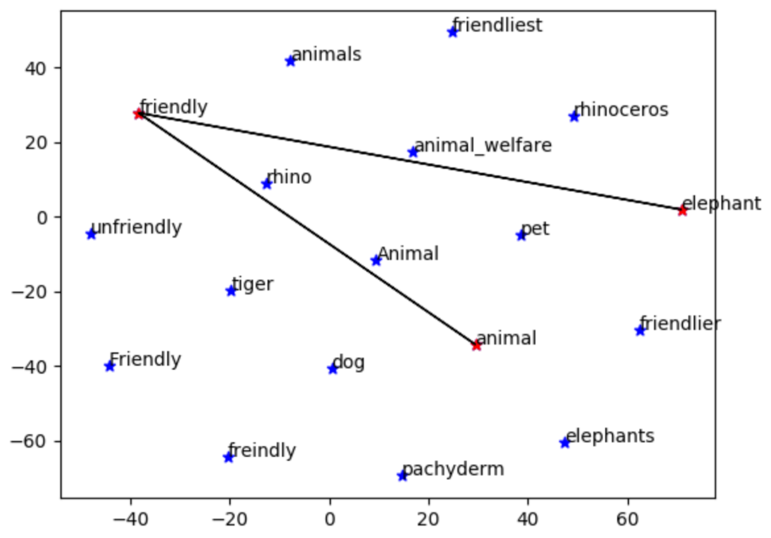


Figure A.3: Visualization of sentence in 3-D using Word2vec, 300 - dimensional vector and 5 neighbors

APPENDIX B

WORD PREDICTION GUI

This section presents the details for Graphical User Interface developed for user incorporating the proposed technique. Figure A.1 shows the GUI interface and explains the basic structure and functions. GUI has the option to load text from predefined directory or accept the user input. After entering the text in the "Input sentence" field and clicking on "Show Prediction" option produces the prediction in "Expected Words" section. At the back-end, the model's output (numbers) are converted back to English vocabulary.

After the prediction, each example is validated using the three performance metrics i.e., F1 , precision and recall. The values for these are displayed in the respective fields.

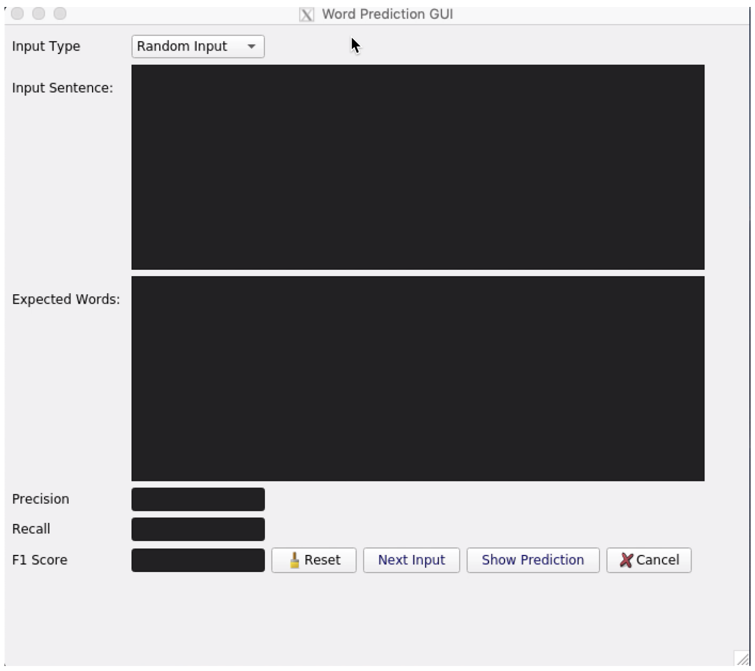
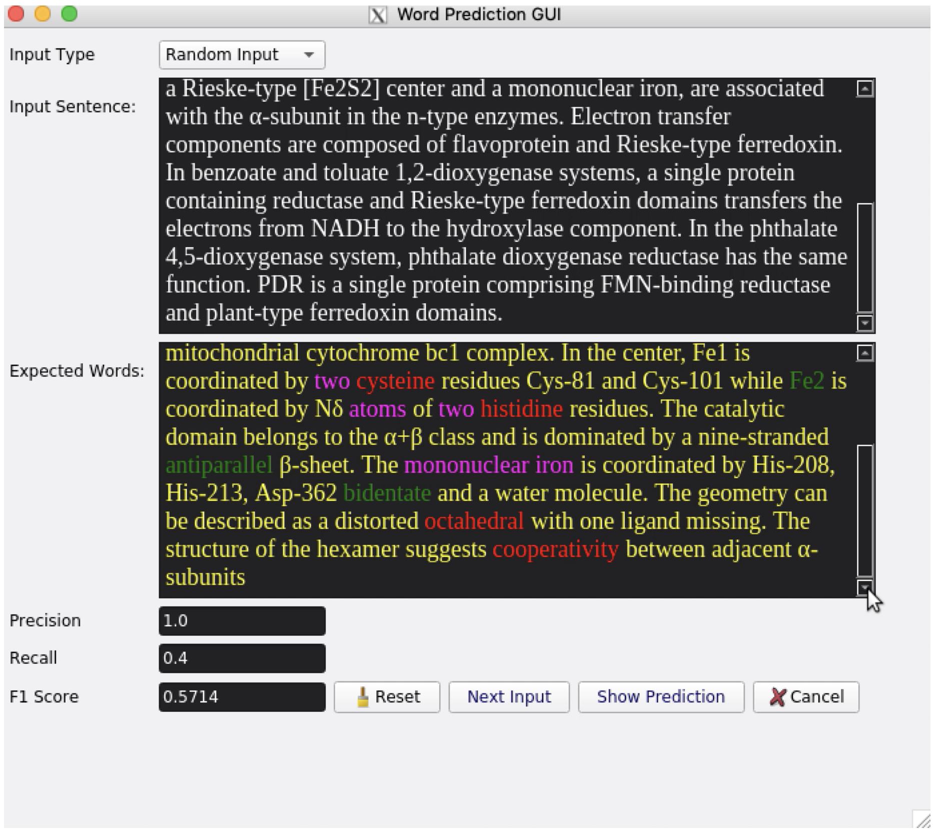


Figure B.1: GUI application to use the model

Once the user enters the input and expected words, the model predicts the words in the section "Expected Words". The corrected predictions are highlighted in green, missed predictions in red, common words between input and expected words in magenta, and stopwords in yellow to help understand better. Here in the example about structure of a chemical compound, the model understood it has to deal with war and it made predictions like “antiparallel”, “bidentate”, “Fe2”.



a Rieske-type [Fe2S2] center and a mononuclear iron, are associated with the α-subunit in the n-type enzymes. Electron transfer components are composed of flavoprotein and Rieske-type ferredoxin. In benzoate and toluate 1,2-dioxygenase systems, a single protein containing reductase and Rieske-type ferredoxin domains transfers the electrons from NADH to the hydroxylase component. In the phthalate 4,5-dioxygenase system, phthalate dioxygenase reductase has the same function. PDR is a single protein comprising FMN-binding reductase and plant-type ferredoxin domains.

mitochondrial cytochrome bc1 complex. In the center, Fe1 is coordinated by two cysteine residues Cys-81 and Cys-101 while Fe2 is coordinated by Nδ atoms of two histidine residues. The catalytic domain belongs to the α+β class and is dominated by a nine-stranded antiparallel β-sheet. The mononuclear iron is coordinated by His-208, His-213, Asp-362 bidentate and a water molecule. The geometry can be described as a distorted octahedral with one ligand missing. The structure of the hexamer suggests cooperativity between adjacent α-subunits

Precision: 1.0
Recall: 0.4
F1 Score: 0.5714

Figure B.2: GUI showing the prediction