Opinion Mining on English and Urdu News Articles

A thesis submitted in fulfillment of the requirements for the degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

RUTURAJ MOHANTY

Entry Number: 2018MCS2012

Under the guidance of

PROF. PARAG SINGLA

and

PROF. MAYA RAMANATH

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi.
June 2020.

Certificate

This is to certify that the thesis titled **OPINION MINING ON ENGLISH AND URDU NEWS ARTICLES** being submitted by **RUTURAJ MOHANTY** for the award of **Master of Technology** in **Computer Science & Engineering** is a record of bona fide work carried out by them under our guidance and supervision at the **Department of Computer Science & Engineering**. The work presented in this thesis has not been submitted elsewhere either in part or full, for the award of any other degree or diploma.

Prof. PARAG SINGLA **Prof. MAYA RAMANATH**

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Acknowledgments

I am grateful to my project guide Prof. PARAG SINGLA and Prof. MAYA RAMANATH, for the opportunity to work under them and be part of the Data Analytics team. Their guidance was instrumental in giving direction to this work. We thank our project associates, Sailesh, Keshav Kolluru for taking time to mentor us. They were always approachable and helped us make progress. We also thank our family and friends for keeping us motivated.

RUTURAJ MOHANTY

Abstract

Opinion Mining is an important yet challenging task. It is a discipline of text classification which continues to be an active area of interest amongst Natural Language Processing researchers. It is also often termed as Sentiment Analysis. It analyses and classifies the user generated data like reviews, blogs, comments, articles, chunks, phrases etc. The main objective of Opinion mining is Sentiment Classification i.e. to classify the opinion into positive, neutral and negative classes. In this thesis work the main focus is to analyze and classify Kashmir or Indian Media specific news articles chunk-wise rather than classifying the entire article. Some chunks are manually labelled and the rest are artificially labelled using a semi-supervised based approach. A multiplicative-Long-Short-Term-Memory(mLSTM) based language model is trained on half a million news articles to extract the required features and then a classifier is trained on top of the extracted features to obtain the desired opinion. Individual neurons of the language model are studied and analyzed to find out if any particular neuron is highly responsible to capture the opinion. A novel unsupervised learning based approach using the technique of zero shot learning is also proposed to break a news article into chunks at test time. A gradient based coloring scheme technique is used to color the chunks to illustrate a more noticeable and meaningful opinion. Final focus concludes with the analysis of opinion on Urdu news articles using a language translator and the trained mLSTM language model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

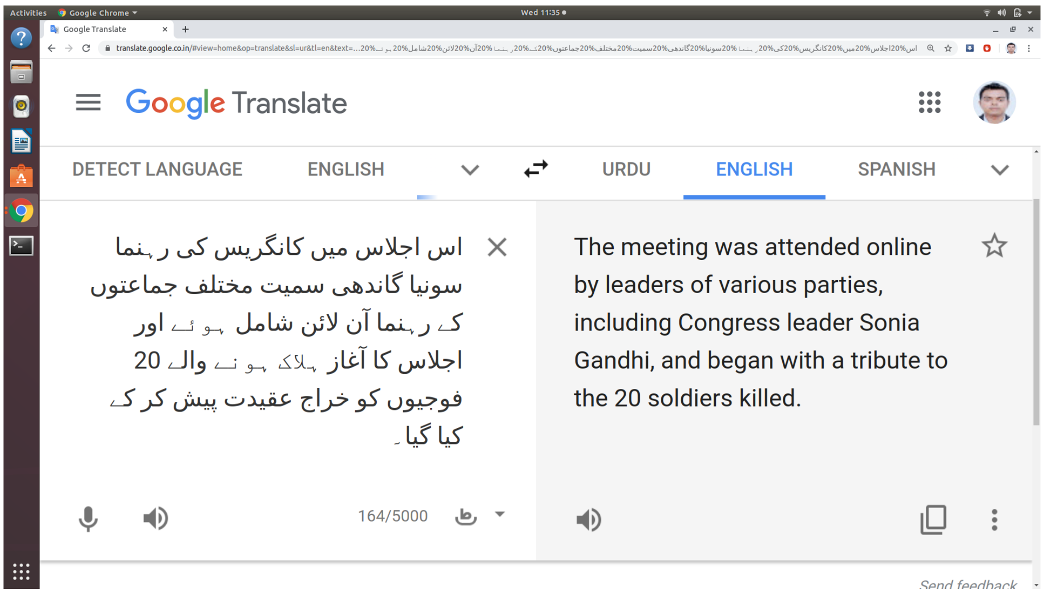
1.1 Motivation

The most challenging task in this thesis work is dealing with Kashmir specific news articles. Most of the articles are centered towards war, army, militants, terrorism, border issues etc. This is one of the reasons why most of the text classifiers like Naive-Bayes, SVM, RNN etc. fail to capture the underlying opinion in such articles. Most of the predictions are simply negative because of presence of such words. The second challenge is to break the article into meaningful chunks. Normal libraries like nltk, spacy are capable of just splitting an article into sentences or word tokens instead of meaningful chunks. The third challenge is to integrate the model with a translator to help in prediction of opinions on Urdu articles. Proper encoding needs to be used while dealing with Urdu articles as normal utf-8, latin-1 is not able to handle all the Urdu characters. This is still an open challenge for most of the Urdu translators like Google translator[3], Yandex[12], IBM watson[5] etc. Currently they just ignore any such unknown characters whose encoding are not found and translate based on an overall meaning of the sentence. Figure 1.1, 1.2 and 1.3 gives a comparison overview of the three translators and how they translate differently for the same given Urdu text.

The final challenge is to deploy the model on the web which would enable the end user to read articles online directly dealing with the user interface without actually bothering about the underlying code and command execution.

1.2 Problem statement

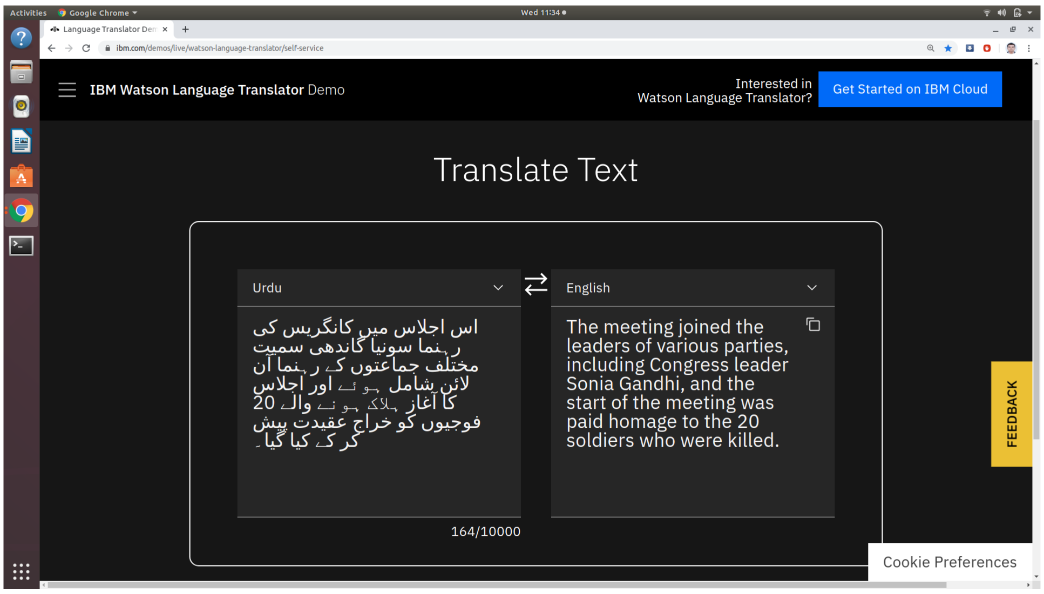
The Opinion Mining project endeavours to develop a web based software to help analyze opinion on news articles centered around Indian media in general



اس اجلاس میں کانگریس کی رہنما سونیا گاندھی سمیت مختلف جماعتوں کے رہنما آن لائن شامل ہوئے اور اجلاس کا آغاز ہلاک ہونے والے 20 فوجیوں کو خراج عقیدت پیش کر کے کیا گیا۔

The meeting was attended online by leaders of various parties, including Congress leader Sonia Gandhi, and began with a tribute to the 20 soldiers killed.

Figure 1.1: Google Translate

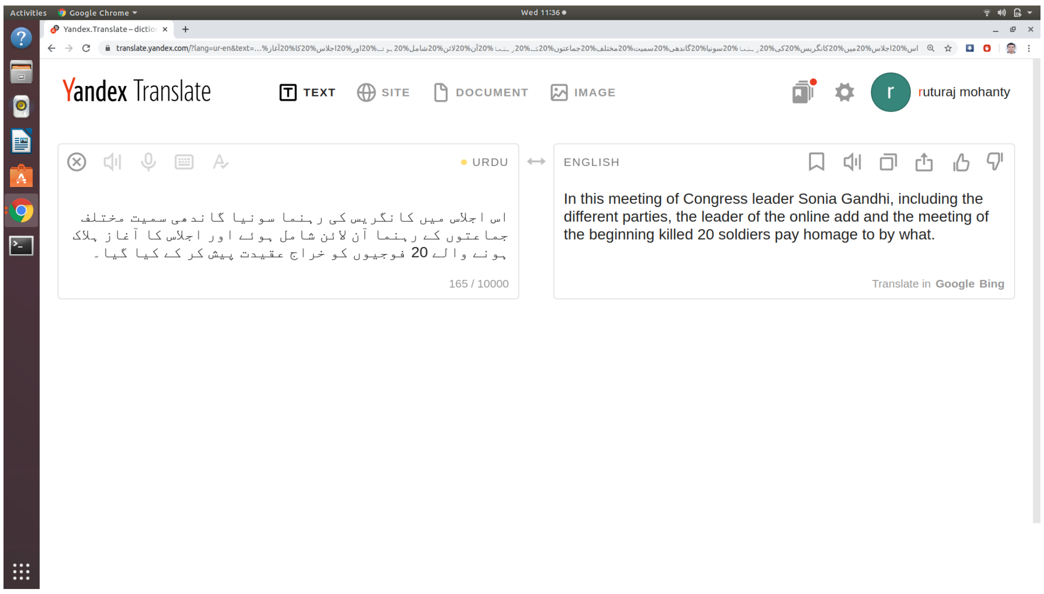


اس اجلاس میں کانگریس کی رہنما سونیا گاندھی سمیت مختلف جماعتوں کے رہنما آن لائن شامل ہوئے اور اجلاس کا آغاز ہلاک ہونے والے 20 فوجیوں کو خراج عقیدت پیش کر کے کیا گیا۔

The meeting joined the leaders of various parties, including Congress leader Sonia Gandhi, and the start of the meeting was paid homage to the 20 soldiers who were killed.

Figure 1.2: IBM Translate

and Kashmir related articles to be specific. One of its focus is to translate Urdu articles and then predict opinion on individual chunks. Through this



اس اجلاس میں کانگریس کی رہنما سونیا گاندھی سمیت مختلف جماعتوں کے رہنما آن لائن شامل ہوئے اور اجلاس کا آغاز ہلاک ہونے والے 20 فوجیوں کو خراج عقیدت پیش کر کے کیا گیا۔

In this meeting of Congress leader Sonia Gandhi, including the different parties, the leader of the online add and the meeting of the beginning killed 20 soldiers pay homage to by what.

Figure 1.3: Yandex Translate

work, different model architectures are examined with a trade-off between accuracy and run time performance. Some accuracy improvement techniques are also carried out. Finally the trade-off between domain specific accuracy over the model's ability to generalise is tested and analyzed.

1.3 Thesis Organisation

The organization of the rest of the thesis is as follows, chapter 2 gives a background of concepts related to sentiment analysis with emphasis on the classifiers used and features extracted. Chapter 3 gives the overview of our approach and solution. Chapter 4 provides the details about the experimental setup and the obtained results. Chapter 5 presents the prototype, software design of IITD News Reader and finally in chapter 6 we conclude by highlighting the specific contribution done in the project and possible future directions.

Chapter 2

Background

2.1 Opinion Mining: A brief introduction

Opinion mining (also sentiment mining, sentiment classification, subjectivity analysis, review mining or appraisal extraction, and in some cases polarity classification) deals with the computational treatment of opinion, sentiment, and subjectivity in text. It intends to ascertain the attitude or opinion of a speaker or writer with respect to a certain topic or target. The attitude could reflect his/her judgment, opinion or evaluation, his/her mind state (how the writer feels at the time of writing) or the intended emotional communication (how the writer wants to affect the reader). Subjective language, or the opinions uttered by a writer, can be categorized into a number of classes or categories: e.g. positive, negative and neutral (i.e. determining the valence, which is also out focus of study); or into an n-point scale - e.g. very good, good, satisfactory, bad, very bad; or into a number of emotions: e.g. joy, sadness, anger, surprise, disgust and fear. In this respect, a sentiment analysis task is seen as a classification task where each category represents a sentiment. Also, based on the nature of task, we often speak of word-level, sentence-level and document-level sentiment classification. In our study we , will dealt with chunk-level sentiment classification. A chunk is simply a meaningful slice from the text. So each article is broken into many consecutive meaningful slices or chunks. In Figure 5.5, 5.6, 5.7 one can clearly observe the difference of splitting the text sentence, phrase and chunk wise.

2.2 Challenges

Dealing with news articles is itself a challenging task. This is because it is not bounded to any specific domain, i.e. news can be related to politics, sports, war, financial etc. Moreover most of the news articles these days are bit biased towards conveying a negative sentiment, i.e. The number of news articles with positive and neutral sentiment are quite less compared to negative sentiment carrying news articles. Kashmir specific articles have a lot of words and phrases related to army, warfare, militants, killings, terrorists, border issues etc. So, mining positive or neutral sentiments from such articles is a tedious task. Now breaking such articles into meaningful chunks is more difficult since we also need to understand the underlying sentiment of the article before concluding an opinion about a chunk. The word ”opinion” here is itself ambiguous. It might be the opinion of the author, or the reader or with respective to someone, or simply generalised view. Capturing individuals sentiment or opinion is difficult as individuals judgement is based on preconceived knowledge, experiences, societal thoughts, personal views, perception etc.

2.3 Learning Algorithms

Researchers have experimented with a number of supervised and unsupervised machine learning techniques for sentiment classification. The current focus would be on two of the well-known supervised learning classifiers for our study Naive-Bayes(NB) and Support Vector Machines(SVM). A variant model(SVM-NB) created using the best of both these models would be used as a baseline model for our purpose. We would be then discussing about some of the unsupervised machine learning techniques using language models, their significance and usage.

2.3.1 Supervised Learning - Naive Bayes

Naive Bayes (NB) [10] is a well-known supervised machine learning technique that is often used for classification tasks. This classifier is called naive because it assumes that the probabilities being combined are independent of each other: the probability of one word in the document being in a specific category is unrelated to the probability of the other words being in that category. Calculating the entire document probability is a matter of multiplying all the probabilities of the individual words in that document. Bayesian classifiers are often used for document classification because they require far less computing power than other methods. According to Jurafsky and Martin, the NB classifier is often used as a good baseline method, with results that are sufficiently good for practical use.

2.3.2 Supervised Learning - Support Vector Machines

Support Vector Machines(SVMs)[6] are often regarded as the classifier that yields the highest accuracy results in text classification problems. They operate by constructing a hyperplane with maximal Euclidean distance to the closest training examples. Simply put, SVMs represent examples as points in space which are mapped to a high-dimensional space where the mapped examples of separate classes are divided by an as wide as possible tangential distance to the hyperplane. New examples are mapped into that same space, and depending on which side of the hyperplane they are positioned, they are predicted to belong to a certain class. SVM hyperplanes are fully determined by a relatively small subset of the training instances, which are called the support vectors. The rest of the training data have no influence on the trained classifier. SVMs have been employed successfully in text classification.

2.3.3 Supervised Learning - Naive Bayes - Support Vector Machines(SVM-NB)

Naive Bayes classifiers are widely used for text classification, however, the strong independence assumptions between features limit their performance in accurately identifying spams. To address this issue, a support machine vector based naive Bayes - SVM-NB - filtering system. The SVM-NB [7] first constructs an optimal separating hyperplane that divides samples in the training set into two categories. For samples located nearby the hyperplane, if they are in different categories, one of them will be eliminated from the training set. In this way, the dependence between samples is reduced and the entire training sample space is simplified. With the trimmed training set, the naive Bayes algorithm is applied to classify texts in the test set. We would be using this model as a baseline for our purpose.

2.3.4 Unsupervised Learning - Language Models

A language model is a probability distribution over all the strings in the language. In other words, the language model assigns a probability to every word/term in the language (Gudivada et al., 2015). Therefore, the primary task in language modeling is to construct a probability distribution function. Once the probabilities for individual words are known, we can compute probabilities for any phrase or sentence in the language. The higher the probability of a sentence, the more likely the sentence is a valid construction in the language. For example, assume that the probabilities associated with the words—information, retrieval, models—are 0.1, 0.15, and 0.05, respectively. Then the probability of the phrase information retrieval models is (0.1) (0.15) (0.05) = 0.00075. Language models have many uses including Part of Speech (PoS) tagging, parsing, machine translation, handwriting recognition, speech recognition, and information retrieval.

There are primarily two types of Language Models:

- Statistical Language Models: These models use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words
- Neural Language Models: These are new players in the NLP town and have surpassed the statistical language models in their effectiveness. They use different kinds of Neural Networks to model language

We would be focusing on Neural Language model for our purpose since they capture deeper understanding of the context in texts and also are scalable in adaptation to multiple domains. Next we would be discussing some of the Neural Language Models which are quite popular and are used in this thesis work.

Language Models - Google BERT

Bidirectional Encoder Representations from Transformers (BERT) [9] is a technique for NLP (Natural Language Processing) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. Google is leveraging BERT to better understand user searches. BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. BERT takes into account the context for each occurrence of a given word.

The paper[9] presents two model sizes for BERT:

> BERT BASE – Less in size, faster, compute efficient and performance comparable to BERT LARGEBERT LARGE – A ridiculously huge model which achieved the state of the art results reported in the paper, slower, uses enormous amount of space and resources for getting trained

BERT is basically a trained Transformer Encoder stack [8].

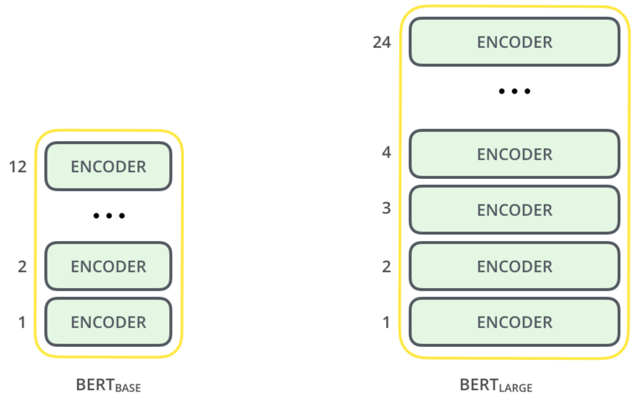
- 

Figure 2.1: Bert Encoder Representation

The encoders are all identical in structure (yet they do not share weights). Each one is broken down into two sub-layers, i.e. Feed forward neural network and Self Attention layer.

The first input token is supplied with a special [CLS] token for reasons that will become apparent later on. CLS here stands for Classification.

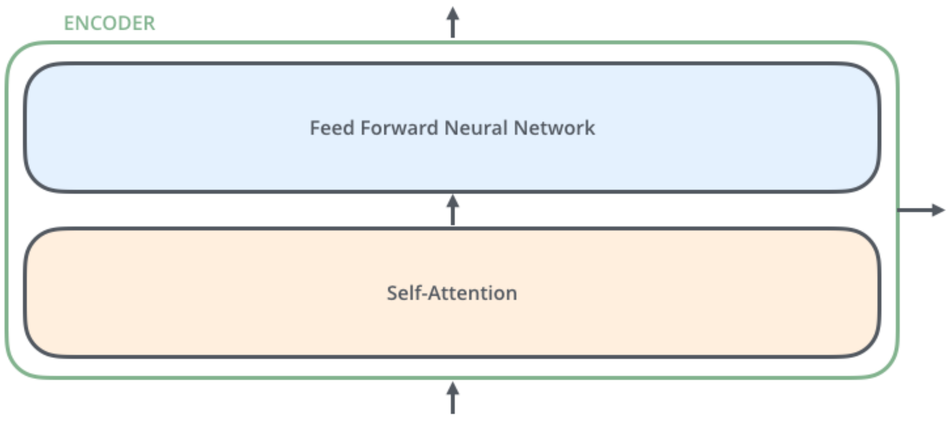


Figure 2.2: Encoder

Just like the vanilla encoder of the transformer, BERT takes a sequence of words as input which keep flowing up the stack. Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder.

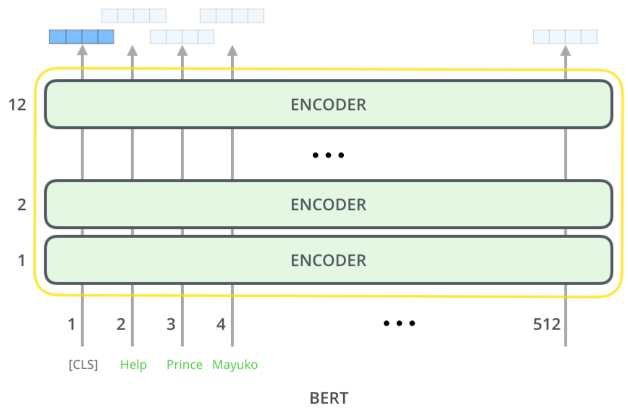


Figure 2.3: Encoder Input Output

The final output of BERT model is the encoded vector of the input string. A decoder model is used to decode this encoded vector and translate it to some other language. But for out purpose we would be treating this encoded vector as the extracted features of the inputted string and then train a classifier using these extracted features.

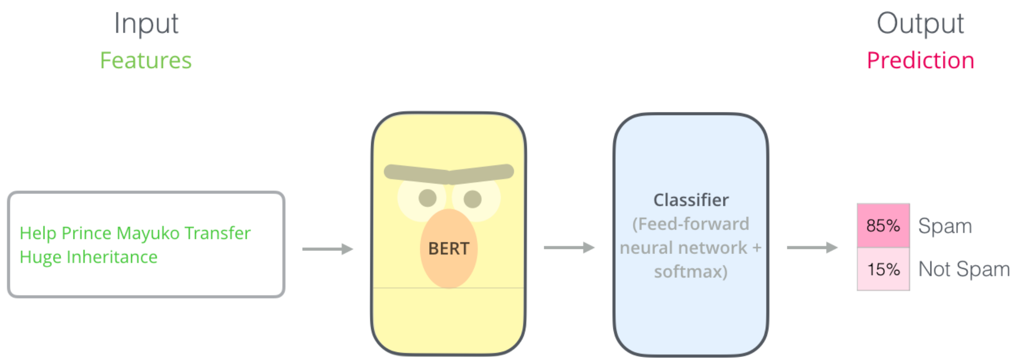

Input
Features
Output
Prediction
Help Prince Mayuko Transfer Huge Inheritance
BERT
Classifier
(Feed-forward neural network + softmax)
85% Spam
15% Not Spam

Figure 2.4: Bert Text Classification

To train such a model, we mainly have to train the classifier, with minimal changes happening to the BERT model during the training phase. This training process is called Fine-Tuning, and has roots in Semi-supervised Sequence Learning. We would be experimenting with it and analyzing the results.

Training BERT like language model is difficult and resource expensive. So we would be training a multiplicative long short term memory (mLSTM) model from scratch for our purpose. Before we dive into mLSTM we need to explore two other models which would help us in understanding mLSTM in a better way.

Language Models - Multiplicative RNN

The multiplicative RNN (mRNN) [8] is an architecture designed specifically to allow flexible input-dependent transitions. Its formulation was inspired by the tensor RNN, an RNN architecture that allows for a different transition matrix for each possible input. The tensor RNN features a 3-way tensor $W_{hh}^{1:N}$, which contains a separately learned transition matrix W_{hh} for each input dimension. The 3-way tensor can be stored as an array of matrices

$$W_{hh}^{1:N} = W_{hh}^{1} ...... W_{hh}^{N} \tag{2.1}$$

where superscript is used to denote the index in the array, and N is the dimensionality of x_t. The specific hidden-to-hidden weight matrix $W_{hh}^{x_t}$ used for a given input x_t is then

$$W_{hh}^{x_t} = \sum_{n=1}^{N} W_{hh}^{n} x_t^{n} \tag{2.2}$$

For language modelling problems, only one unit of x_t will be on, and $W_{hh}^{x_t}$ will be the matrix in $W_{hh}^{1:N}$ corresponding to that unit. Hidden-to-hidden propagation in the tensor RNN is then given by

$$\hat{h} = W_{hh}^{x_t} \, h_{t-1} + W_{hx} \, x_t \tag{2.3}$$

The large number of parameters in the tensor RNN make it impractical for most problems. mRNNs can be thought of as a shared-parameter approximation to the tensor RNN that use a factorized hidden-to-hidden transition matrix in place of the normal RNN hidden-to-hidden matrix Whh, with an input-dependent intermediate diagonal matrix $diag(W_{mx} x_t)$. The input-dependent hidden-to-hidden weight matrix, $W_{hh}^{x_t}$ is then

$$W_{hh}^{x_t} = W_{hm} \, diag(W_{mx} x_t) \, W_{mh} \tag{2.4}$$

An mRNN is thus equivalent to a tensor RNN using the above form for $W_{hh}^{x_t}$. For readability, an mRNN can also be described using intermediate state mt as follows:

$$m_t = (W_{mx}\, x_t) \odot (W_{mh}\, h_{t-1}) \tag{2.5}$$

$$\hat{h} = W_{hh}\, m_t + W_{hx}\, x_t \tag{2.6}$$

mRNNs have improved on vanilla RNNs at character level language modelling tasks (Sutskever et al., 2011; Mikolov et al., 2012), but have fallen short of the more popular LSTM architecture, for instance as shown with LSTM baselines from (Cooijmans et al., 2017). The standard RNN units in an mRNN do not provide an easy way for information to bypass its complex transitions, resulting in the potential for difficulty in retaining long term information.

Language Models - Long Short Term Memory(LSTM)

LSTM [4] is a commonly used RNN architecture that uses a series of multiplicative gates to control how information flows in and out of internal states of the network (Hochreiter Schmidhuber, 1997). There are several slightly different variants of LSTM, and we present the variant used in our experiments. The LSTM hidden state receives inputs from the input layer x_t and the previous hidden state h_{t1}:

$$\hat{h}_t = W_{hx}\, x_t + W_{hh}\, h_{t-1} \tag{2.7}$$

The LSTM network also has 3 gating units – input gate i, output gate o, and forget gate f – that have both recurrent and feed-forward connections:

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1}) \tag{2.8}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1}) \tag{2.9}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1}) \tag{2.10}$$

where σ is the logistic sigmoid function. The input gate controls how much of the input to each hidden unit is written to the internal state vector ct, and the forget gate determines how much of the previous internal state ct1 is preserved. This combination of write and forget gates allows the network to control what information should be stored and overwritten across each time-step. The internal state is updated by

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(\hat{h}_t) \tag{2.11}$$

The output gate controls how much of each unit's activation is preserved. It allows the LSTM cell to keep information that is not relevant to the current output, but may be relevant later. The final output of the hidden state is given by

$$h_t = tanh(c_t) \odot o_t \tag{2.12}$$

LSTM's ability to control how information is stored in each unit has proven generally useful.

Language Models - Multiplicative LSTM(mLSTM)

Since the LSTM and mRNN architectures are complimentary, multiplicative LSTM (mLSTM)[2], a hybrid architecture that combines the factorized hidden-to-hidden transition of mRNNs with the gating framework from LSTMs. The mRNN and LSTM architectures can be combined by adding connections from the mRNN's intermediate state mt (which is redefined below for convenience) to each gating units in the LSTM, resulting in the following system:

$$m_t = (W_{mx}\, x_t) \odot (W_{mh}\, h_{t-1}) \tag{2.13}$$

$$\hat{h}_t = W_{hx}\, x_t + W_{hm}\, m_t \tag{2.14}$$

$$i_t = \sigma(W_{ix} x_t + W_{im}\, m_t) \tag{2.15}$$

$$o_t = \sigma(W_{ox} x_t + W_{om}\, m_t) \tag{2.16}$$

$$f_t = \sigma(W_{fx} x_t + W_{fm}\, m_t) \tag{2.17}$$

We set the dimensionality of m_t and h_t equal for all our experiments. We also chose to share mt across all LSTM unit types, resulting in a model with 1.25 times the number of recurrent weights as LSTM for the same number of hidden units. The goal of this architecture is to combine the flexible input-dependent transitions of mRNNs with the long time lag and information control of LSTMs. The gated units of LSTMs could make it easier to control (or bypass) the complex transitions in that result from the factorized hidden weight matrix. The additional sigmoid input and forget gates featured in LSTM units allow even more flexible input-dependent transition functions than in regular mRNNs.

Chapter 3

Methodology

In this section, we discuss the choice of model architecture and trade-offs. We also review typical strategies to improve accuracy exercised during training and explored strategies for robust predictions.

3.1 Dataset Information

Around 0.5 million raw news articles comprising of Kashmir specific articles and Indian media news articles were crawled and collected. This set of articles is used to train our mLSTM unsupervised language model from scratch. Around 500 articles were labelled chunk-wise as positive, negative or neutral to generate around 7000 labelled chunks. This set of chunks is used to train the final opinion predicting classifier using the features generated after processing the set through our language model.

3.2 Coloring Scheme Used

Instead of coloring the chunks directly as positive, negative or neutral, a gradient based coloring scheme was used for more understand-ability. To achieve this, the scores of each of the labels(Negative ,Positive, Neutral) was mapped to corresponding range of RGB scale respectively with the help of a mapping function, i.e. Negative score was mapped to R(Red) scale 120-255, Positive was mapped to G(Green) scale 120-255, Neutral scale was mapped to B(Blue) scale 120-255. The lower bound 120 was used instead of 0 to make the color look significant and the upper bound was kept at 255. Mapping function used was as follows:

$$slope = \frac{outputEnd - outputStart}{inputEnd - inputStart} \tag{3.1}$$

$$output = outputStart + slope * (input - inputStart) \tag{3.2}$$

outputStart was kept at 255(the upper bound of RGB value), *outputEnd* was kept at 120(the lower bound of RGV value). We mapped (255,120) to (*outputStart*, *outputEnd*) instead of (120,255) to (*outputStart*, *outputEnd*) because we wanted the darker shade of each color to be mapped to the highest input scores. *inputStart* and *inputEnd* values were the corresponding extremes of normalised score values of each sentiment with their thresholds, i.e. (-1.0, -0.05) corresponds to negative sentiment, (-0.05, 0.05) corresponds to neutral sentiment and (0.05 to 1.0) corresponds to positive sentiment. The threshold -0.05 and 0.05 were chosen based on results on validation dataset. *input* was the actual score obtained after passing the chunk through the classifier. Figure 3.1 depicts the coloring scheme used.

3.3 Solution Approach

We would be now discussing the tasks undertaken to find the solution to our problem statement.

Choice of Language Model

We chose mLSTM language model as our training model in reference to the OpenAI paper [1]. Our initial assumption was to see if in our case also the mLSTM model behaves the same way as illustrated in the OpenAI paper [1], i.e. The sentiment depends only on a particular sentiment neuron. This is one of the reasons why mLSTM was chosen over other language models. Also training mLSTM is simpler, resource economical and the model itself is very robust in handling new unseen inputs.

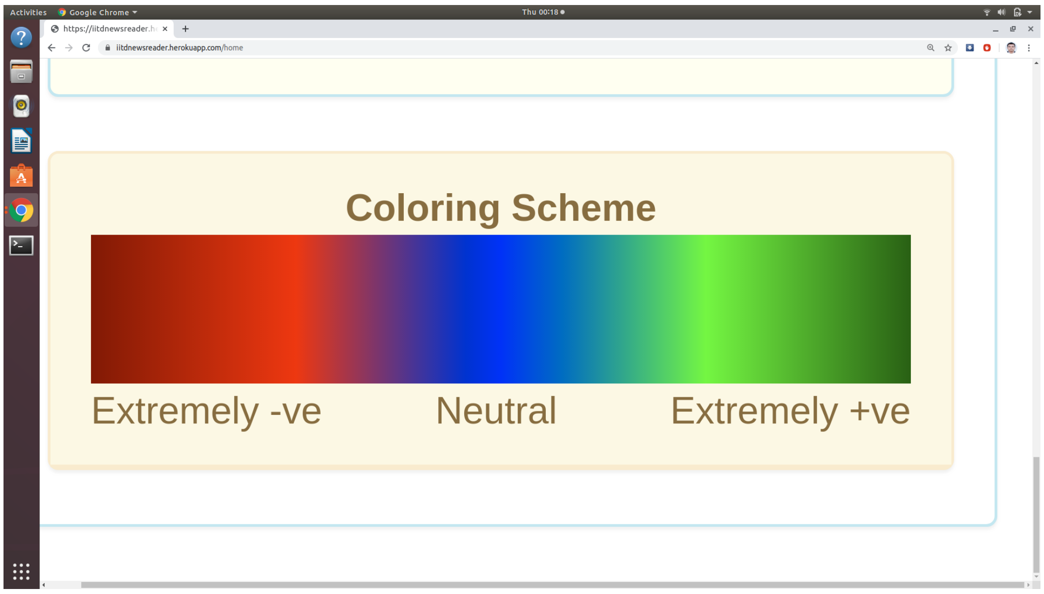


Figure 3.1: Coloring Scheme Used

Language Model Training

Our mLSTM language model was trained on 0.5 million news articles comprising of both Kashmir specific and Indian Media news articles. The entire dataset was pre-processed and then divided into 10 different shards to successfully fit in the memory. Each shard was then processed one after the other to train the model which took around 15 days of time. The model was trained for a total of 100 epochs with an average bpc(bits per character) reaching 1.15 . A total of 3 models were trained with different hyper parameter settings. The sequence length was kept at 256, vocab size at 256, word embedding size at 64 and batch size at 128. Two models were trained using 2048 RNN hidden units and one models with 4096 RNN hidden units. The models were trained on HPC on one V100 gpu, training took around 15days for a single model to get trained. During training very often the gpu memory showed memory overflow errors as the entire tensor graph was being loaded into the 32GB V100 memory. To avoid this the batch size was sometimes lowered to 64 and check-pointing was done to ensure uninterrupted training of the model from the instance where it stopped previously.

Classifier Training

After language model training, several classifiers like xgboost, softmax, neural network, svm etc. were trained on top of our mLSTM language model. The training data used here were the 7000 labelled chunks. These chunks were then fed into our mLSTM language model, the output of the language model were the semantic features. These features are then used to train the classifier and the labelled sentiment were used as the target values. As the training data was quite less, a form of semi-supervised technique was used to generate more training data. We would be discussing about it in the next sub section.

Semisupervised Approach to Generate More Labelled Chunks

Dr. Parth Talukdar's work [11] was used for this purpose. Out of our pre-labelled 7000 chunks, 700 chunks were used as seed labels. Seed labels are just the initial labels. Another set of 100 new chunks were used as the new labels. A graph was constructed between every pair of nodes with the edge weight being cosine similarity of the two chunks(extracted features from our mLSTM language model was used for representing a chunk). Then this graph was passed through the developed software to obtain the new labels. In the end of this process we got 7100 labelled chunks in total. It didn't improve a lot on the overall accuracy, and also the software being old and outdated consumed a lot of memory and was hard to run using all the 7000 chunks as seed labels. Nevertheless this approach seems quite promising and in future with sufficient resources can be further extended.

Chunking Algorithm

Till now we haven't discussed how the chunking of entire news article was done during test time. In this sub section we would be going through the chunking process that we used during test time. We have used a zero-shot classifier kind of approach for this purpose. This algorithm simulates exactly the way we human do chunking. We keep on reading words one after the other till the tone of sentence doesn't change. Tone can be anything but in our case tone is one among the sentiment (Positive, Negative, Neutral). In other words, we read a word, mark its tone, then move to next word and see if we add the new word to our existing word does that changes the tone ? if not then add the current word to the previous word else break the chunk here. We keep on doing this till we reach the end of the article. To find the tone, we are comparing the cosine similarity of current sentence with three words, i.e. Positive, Negative, Neutral. In short, we are passing our current formed sentence into our mLSTM language model, the output gives the feature vector say FV1, then we are passing the three words (Positive, Negative, Neutral) into out mLSTM language model one by one, again the output gives three different feature vectors say FV2, FV3, FV4. Now we take the cosine similarity between (FV1, FV2), (FV1, FV3), (FV1, FV4), the argument maximum of these three values decides our tone. The simplest form of the algorithm(pseudo-code) in details is as follows:-

```
1) Get news article
2) Break the news article into phrases using nltk library
3) Iterate on all the phrases
4) Let the previous tone is T1
5) If arg_max{cosine_similarity(curr_chunk + phrase, positive),
   cosine_similarity(curr_chunk + phrase, negative),
   cosine_similarity(curr_chunk + phrase, neutral)} == T1:
    curr_chunk = curr_chunk + phrase
    continue
   Else:
    save curr_chunk in chunks list
    set curr_chunk as empty
```

But in practice we are using a bit complicated version of it. Here's what we have used in practice:-

```
1) Get news article
2) Break the news article into phrases using nltk library
```

```
3) Iterate on all the phrases
4) Let the previous tone is T1
5) If arg_max{cosine_similarity(curr_chunk + phrase, positive),
   cosine_similarity(curr_chunk + phrase, negative),
   cosine_similarity(curr_chunk + phrase, neutral)} == T1
   and cosine_similarity(curr_chunk + phrase, T1) >=
   cosine_similarity(curr_chunk, T1) + Threshold:
    curr_chunk = curr_chunk + phrase
    continue
   Else:
    save curr_chunk in chunks list
    set curr_chunk as empty
```

The additional condition

```
cosine_similarity(curr_chunk + phrase, T1) >=
cosine_similarity(curr_chunk, T1) + Threshold
```

ensures that the tone is always monotonically increasing. This is significant in most of the chunks that we deal with in day to day life. For Eg. Consider the chunk, "He is a good boy", when our algorithm encounters "He" it is just somewhat on the positive side, then it encounters "is" and our chunk becomes "He is", now our model is bit more positive, then it encounters "good" and our chunk becomes 'He is good", now our model is more confident and is on positive side, finally it encounters "boy" and our chunk becomes "He is good boy", now our model is on positive side but adding the word "boy" didn't improve any further on the positive score. So we would break the chunks as "He is good" and "boy". Although it looks a bit odd now, practically when we would be coloring it using gradient coloring technique it would make a lot more sense. The chunk "He is good" would be colored in dark green shade and "boy" would be shaded in a bit light green shade. So overally it looks more meaningful and dynamic as it can capture complex chunks and thereby help improve on the accuracy of our work. This in fact worked tremendously well, it helped in generalisation and adaptation to lot of other domain news articles. A small threshold is used to ignore insignificant or small change is score. Figure 3.2 demonstrates a conceptual view of how the chunking algorithm works.

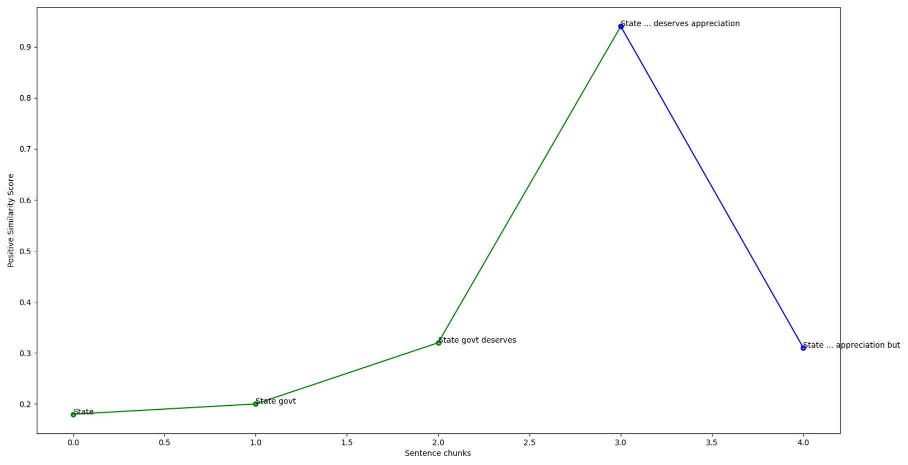


Figure 3.2: An example of Chunking Algorithm

On Urdu Articles

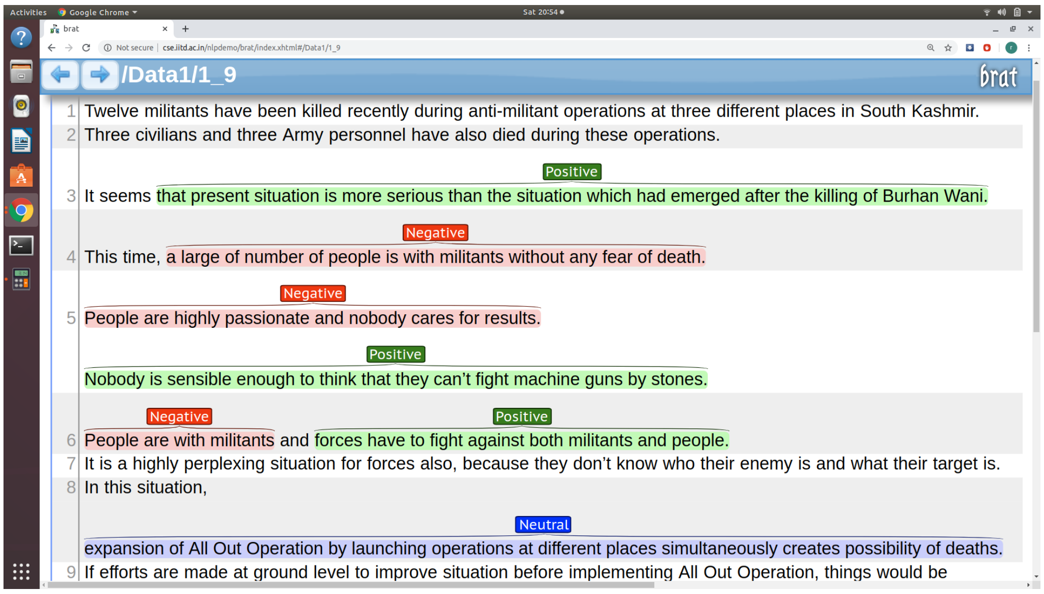
To handle our model on Urdu articles we first translated the Urdu articles into English. This was done using IBM's watson language translator tool which gave really good results. After translation we followed our normal steps to predict sentiment on the translated english article and display the results accordingly.

Chapter 4

Experimentation, Results and Observations

4.1 Labelling of Chunks

We used a rapid annotation tool, namely brat, to do labelling of chunks. Brat is a web-based tool for text annotation; that is, for adding notes to existing text documents. It is designed in particular for structured annotation, where the notes are not freeform text but have a fixed form that can be automatically processed and interpreted by a computer. It also supports the annotation of n-ary associations that can link together any number of other annotations participating in specific roles. We configured the environment settings as per our requirement.



1 Twelve militants have been killed recently during anti-militant operations at three different places in South Kashmir.
2 Three civilians and three Army personnel have also died during these operations.

Positive
3 It seems that present situation is more serious than the situation which had emerged after the killing of Burhan Wani.

Negative
4 This time, a large of number of people is with militants without any fear of death.

Negative
5 People are highly passionate and nobody cares for results.

Positive
Nobody is sensible enough to think that they can't fight machine guns by stones.

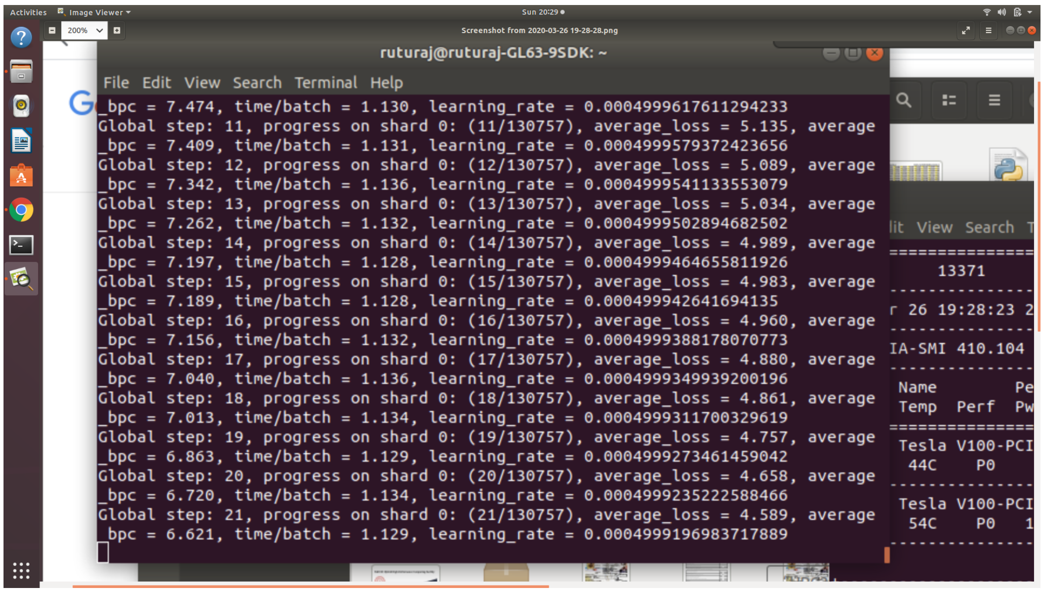
Negative | Positive
6 People are with militants and forces have to fight against both militants and people.
7 It is a highly perplexing situation for forces also, because they don't know who their enemy is and what their target is.
8 In this situation,

Neutral
expansion of All Out Operation by launching operations at different places simultaneously creates possibility of deaths.
9 If efforts are made at ground level to improve situation before implementing All Out Operation, things would be

Figure 4.1: Manual labelling of chunks

4.2 Unsupervised learning

Our unsupervised mLSTM language model was trained for 100 epochs. Each epoch consisted of going through the 10 shards. Each shard consisted of $\frac{1}{10}th$ of the entire dataset. The average bpc (bits per character) obtained was 1.15. Time per batch was around 1.3 seconds when the batch size was kept at 128. The learning rate was decayed by a constant factor and after reaching a bpc of 1.2 it was kept constant at a very lower value. The total time to train one model took around 15 days of time. Initially the model was trained on Google cloud with its free credits. But later as shifted to HPC(High Performance Computing) IIT Delhi on priority usage settings. The model was trained on HPC IIT Delhi on a V100 gpu with 32GB gpu memory capacity. It can also be trained on 4 P100 gpus with 32GB memory capacity each in a distributed fashion. The time taken is almost the same, in fact V100 took less time compared to 4 P100 gpus. One significant note is if you are training in distributed fashion you need to take utmost care while updating the weights. I recommend training on 1 single V100 gpu rather than 4 P100 gpu's to get better performance and accuracy.


Activities
Image Viewer
Sun 20:29
200%
Screenshot from 2020-03-26 19-28-28.png
ruturaj@ruturaj-GL63-9SDK: ~
File Edit View Search Terminal Help

```
_bpc = 7.474, time/batch = 1.130, learning_rate = 0.0004999617611294233
Global step: 11, progress on shard 0: (11/130757), average_loss = 5.135, average
_bpc = 7.409, time/batch = 1.131, learning_rate = 0.0004999579372423656
Global step: 12, progress on shard 0: (12/130757), average_loss = 5.089, average
_bpc = 7.342, time/batch = 1.136, learning_rate = 0.0004999541133553079
Global step: 13, progress on shard 0: (13/130757), average_loss = 5.034, average
_bpc = 7.262, time/batch = 1.132, learning_rate = 0.0004999502894682502
Global step: 14, progress on shard 0: (14/130757), average_loss = 4.989, average
_bpc = 7.197, time/batch = 1.128, learning_rate = 0.0004999464655811926
Global step: 15, progress on shard 0: (15/130757), average_loss = 4.983, average
_bpc = 7.189, time/batch = 1.128, learning_rate = 0.000499942641694135
Global step: 16, progress on shard 0: (16/130757), average_loss = 4.960, average
_bpc = 7.156, time/batch = 1.132, learning_rate = 0.0004999388178070773
Global step: 17, progress on shard 0: (17/130757), average_loss = 4.880, average
_bpc = 7.040, time/batch = 1.136, learning_rate = 0.0004999349939200196
Global step: 18, progress on shard 0: (18/130757), average_loss = 4.861, average
_bpc = 7.013, time/batch = 1.134, learning_rate = 0.0004999311700329619
Global step: 19, progress on shard 0: (19/130757), average_loss = 4.757, average
_bpc = 6.863, time/batch = 1.129, learning_rate = 0.0004999273461459042
Global step: 20, progress on shard 0: (20/130757), average_loss = 4.658, average
_bpc = 6.720, time/batch = 1.134, learning_rate = 0.0004999235222588466
Global step: 21, progress on shard 0: (21/130757), average_loss = 4.589, average
_bpc = 6.621, time/batch = 1.129, learning_rate = 0.0004999196983717889
```

```
lit  View  Search  T
===============
      13371
---------------
r 26 19:28:23 2
---------------
IA-SMI 410.104
---------------
 Name         Pe
 Temp   Perf  Pw
===============
 Tesla  V100-PCI
  44C     P0
---------------
 Tesla  V100-PCI
  54C     P0    1
---------------
```

Figure 4.2: Unsupervised Language Model Training

Model	Hidden Units	Batch Size	Vocab Size	Embedding Size	Avg BPC
1	2048	128	256	64	1.21
2	2048	128	256	128	1.20
3	4096	128	256	64	1.15

Table 4.1: Models and their avg bits per character

Looking at table 4.1 it is clear that increasing the embedding size doesn't improve the average bpc by a significant amount. Increasing the embedding size also increases the time taken to train the model by significant amount. So in model 3 we have used embedding size to be 64. The sequence length was kept at 256, it was found out to be optimal as increasing it to 512 gave memory error and reducing it to 128 resulted in poor avg bpc.

4.3 Supervised Classifier Training

Here we trained different classifier like Softmax, SVM, xgboost, neural network on top our unsupervised language model. We also experimented with some pre-existing language models like BERT. We have used SVM-NB as our baseline comparison model. We have used a train-test split ratio of 9:1 on our total number of labelled chunks. 5-fold test accuracy is reported to capture more generalised accuracy.

4.3.1 Finding out the best learning rate

To find out the best learning rate, we simulated the training for a few epochs and plotted the graph. On basis of the graph, we chose the learning rate that corresponded to minimum loss. Refer Figure 4.3 and 4.4 for an illustrative view.

4.3.2 Observations

From Table 4.2 it can be clearly seen that the unsupervised language model + classifier models performed really well in comparison to our baseline SVM-

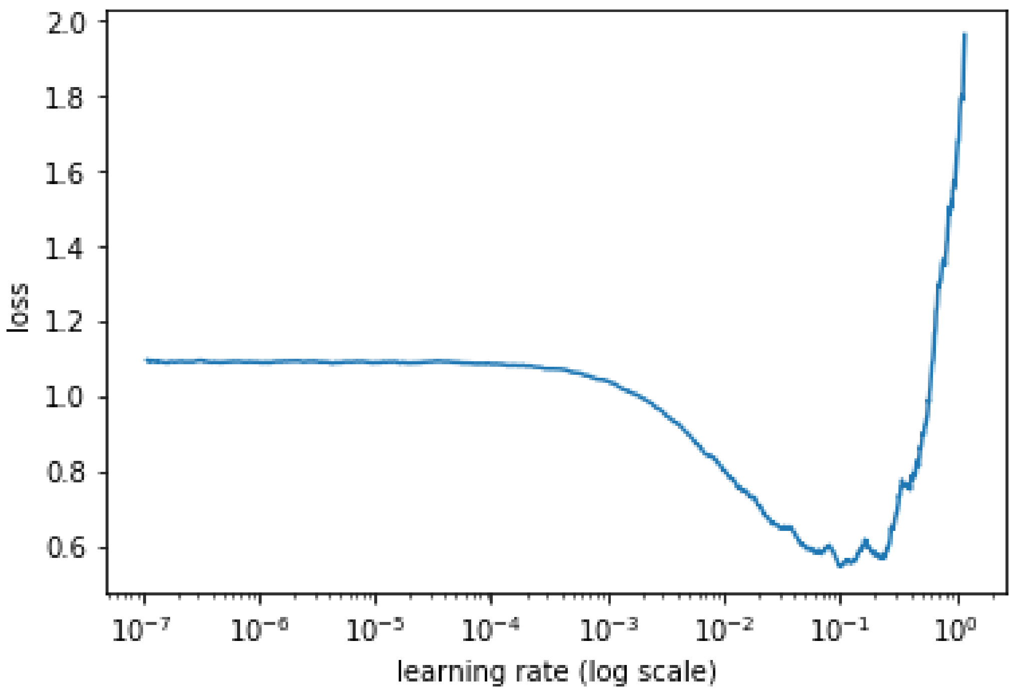


Figure 4.3: Loss vs learning rate - SVM-NB

NB model. In our domain (Kashmir and Indian media news articles) our model outperformed BERT's accuracy. Although BERT is a generalised language model and can be used in a variety of other domains like sports, politics, finance etc which our model fails to do as it is trained only on India and Kashmir specific news articles. Out of the many classifiers Softmax seems to perform really well and faster to train also. During experimentation XGBoost and SVM many a times deviated by significant margin in their test time accuracy but Softmax almost remained constant within a small range. Neural nets on the other hand can undoubtedly perform really well on top of our unsupervised model, but it requires more number of training examples or chunks. One should definitely go for neural networks if there is sufficient labelled chunks.

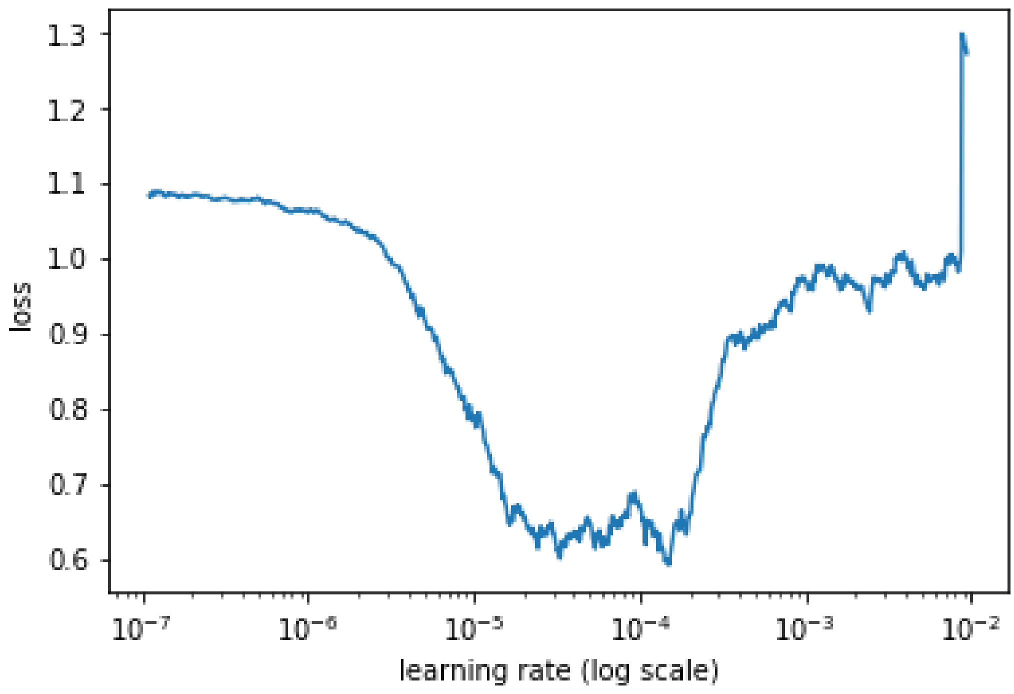


Figure 4.4: Loss vs learning rate - Language Model + Classifier

Language Model	Classifier	Test Accuracy
-	SVM-NB	68.8
BERT	Softmax	84.6
mLSTM	Softmax	90.52
mLSTM	XGBoost	92.06
mLSTM	Neural-net	85.20
mLSTM	SVM	91.44

Table 4.2: Models and their 5-fold test set accuracy

Chapter 5

User Interface and Practical Application View

In this chapter we would be viewing the developed software, it's architecture and practical usage and application.

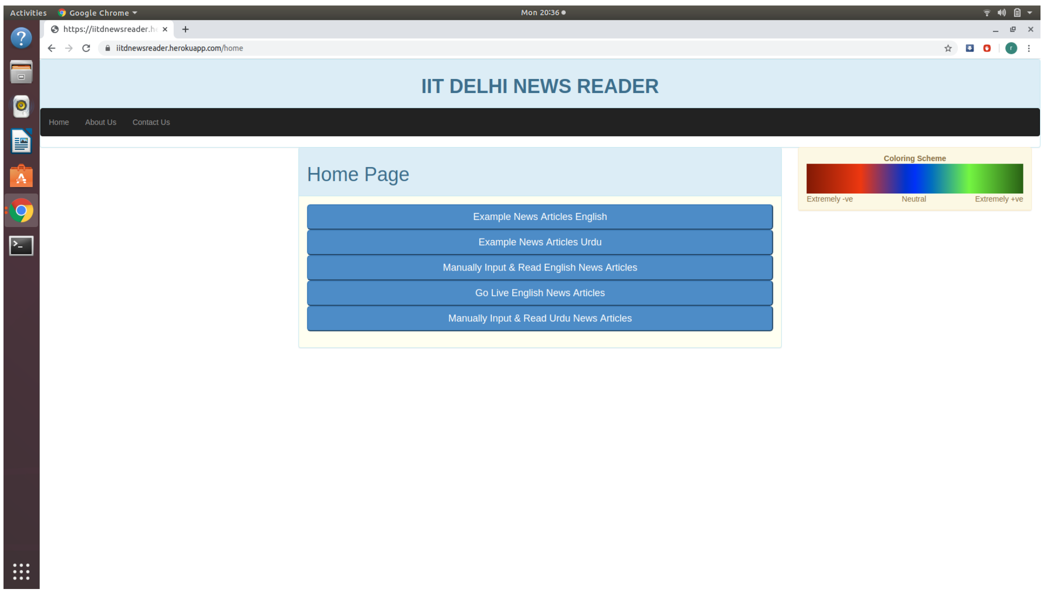
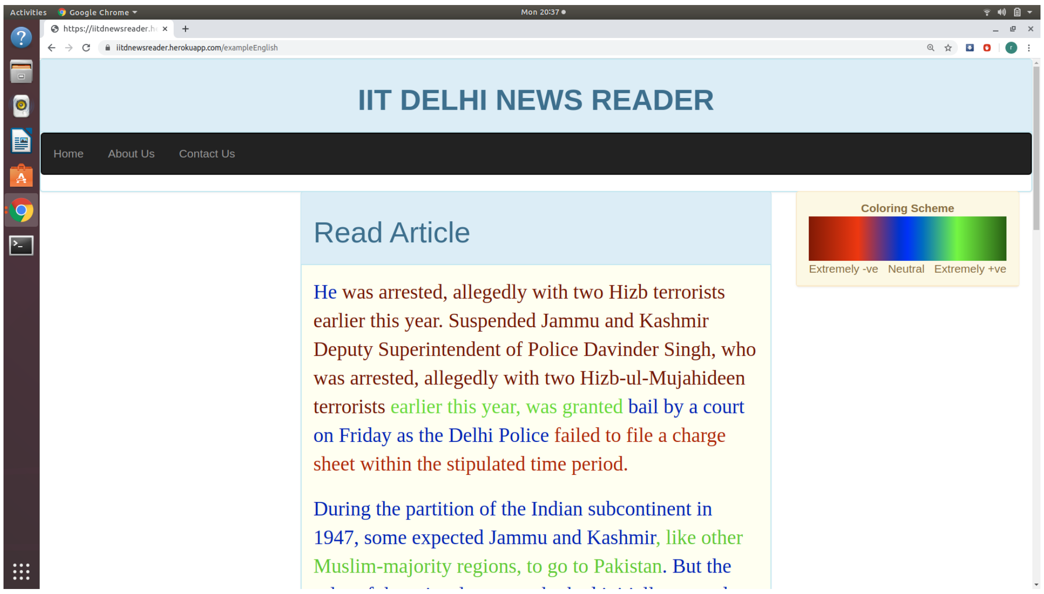

IIT DELHI NEWS READER
Home
About Us
Contact Us
Home Page
Example News Articles English
Example News Articles Urdu
Manually Input & Read English News Articles
Go Live English News Articles
Manually Input & Read Urdu News Articles
Coloring Scheme
Extremely -ve
Neutral
Extremely +ve

Figure 5.1: Home Page With Different Options

Although we have used our own chunking algorithm to break or splits into chunks, a separate option is also provided to do chunking based on sentence-wise split and phrase-wise split. The default one is the chunk-wise split, i.e. using our own chunking algorithm.



IIT DELHI NEWS READER

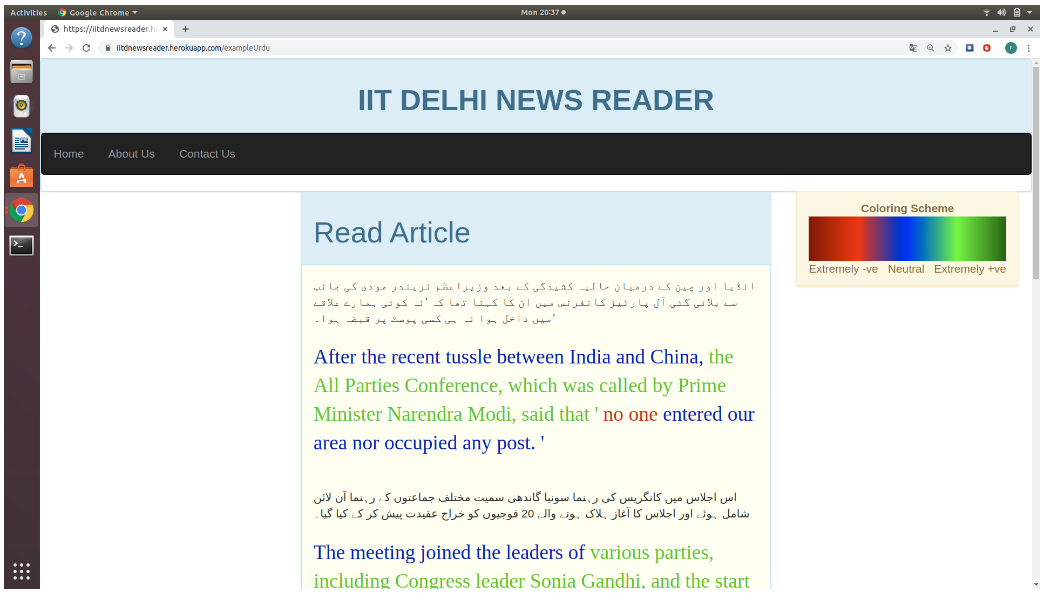
Home About Us Contact Us

Read Article

He was arrested, allegedly with two Hizb terrorists earlier this year. Suspended Jammu and Kashmir Deputy Superintendent of Police Davinder Singh, who was arrested, allegedly with two Hizb-ul-Mujahideen terrorists earlier this year, was granted bail by a court on Friday as the Delhi Police failed to file a charge sheet within the stipulated time period.

During the partition of the Indian subcontinent in 1947, some expected Jammu and Kashmir, like other Muslim-majority regions, to go to Pakistan. But the

Figure 5.2: Example English News Article



IIT DELHI NEWS READER

Home About Us Contact Us

Read Article

انڈیا اور چین کے درمیان حالیہ کشیدگی کے بعد وزیراعظم نریندر مودی کی جانب سے بلائی گئی آل پارٹیز کانفرنس میں ان کا کہنا تھا کہ ’نہ کوئی ہمارے علاقے میں داخل ہوا نہ ہی کسی پوسٹ پر قبضہ ہوا۔‘

After the recent tussle between India and China, the All Parties Conference, which was called by Prime Minister Narendra Modi, said that ' no one entered our area nor occupied any post. '

اس اجلاس میں کانگریس کی رہنما سونیا گاندھی سمیت مختلف جماعتوں کے رہنما آن لائن شامل ہوئے اور اجلاس کا آغاز ہلاک ہونے والے 20 فوجیوں کو خراج عقیدت پیش کر کے کیا گیا۔

The meeting joined the leaders of various parties, including Congress leader Sonia Gandhi, and the start

Figure 5.3: Example Urdu News Article

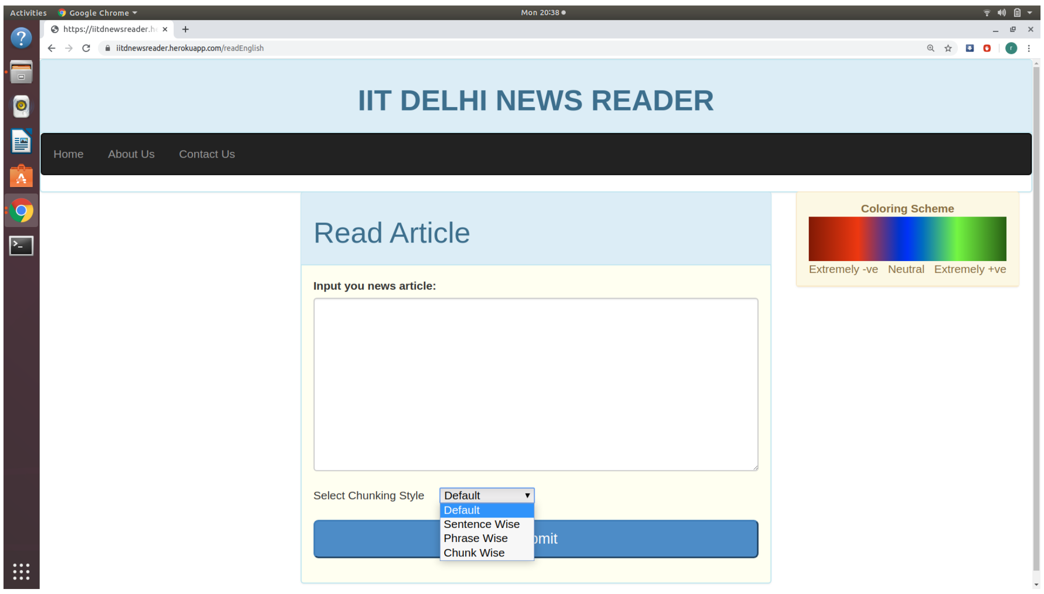
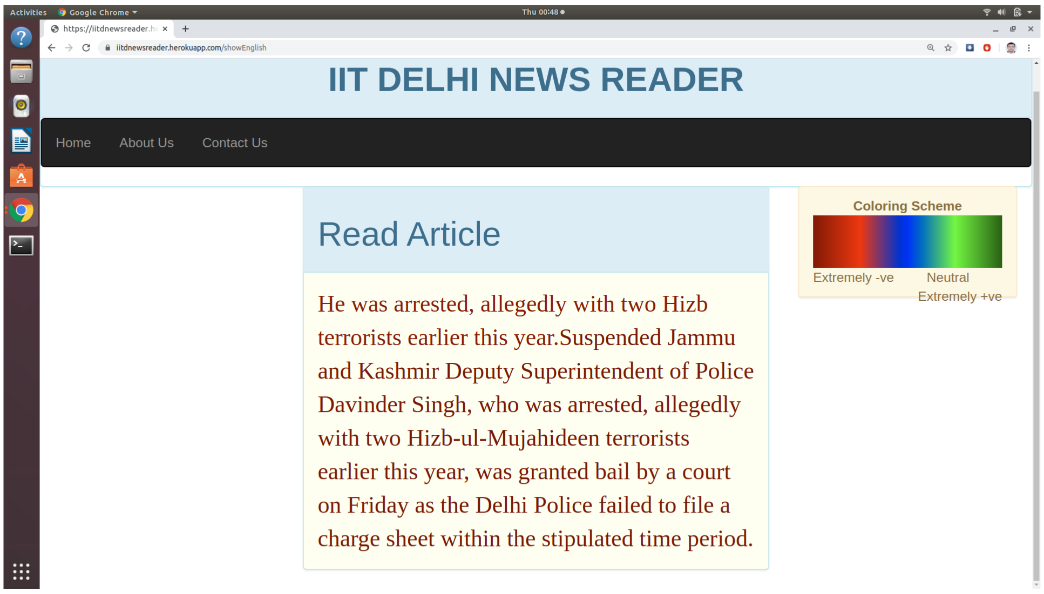
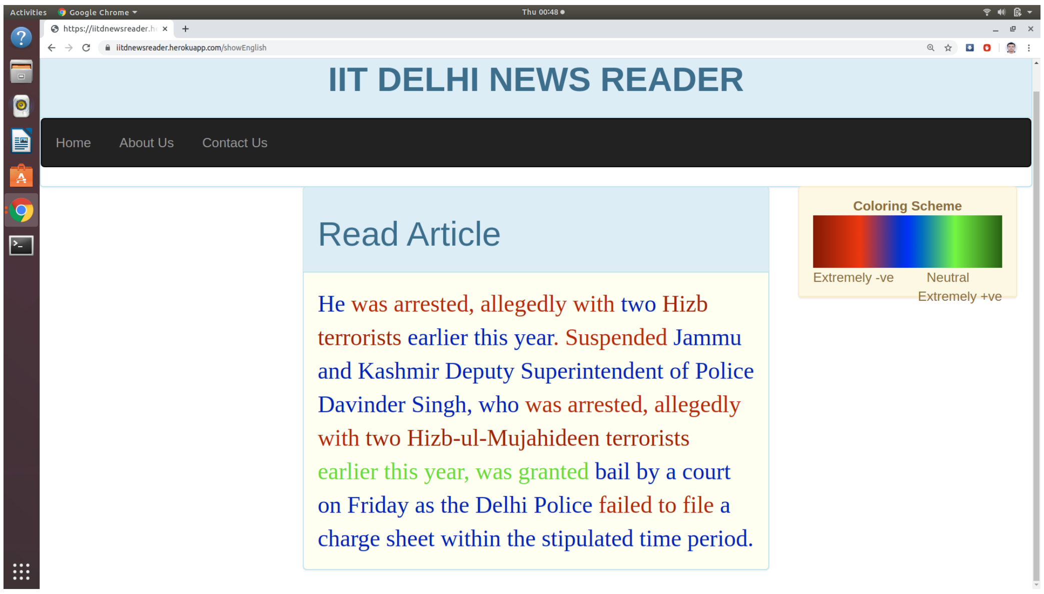


Figure 5.4: Option to manually input news article and chunking style



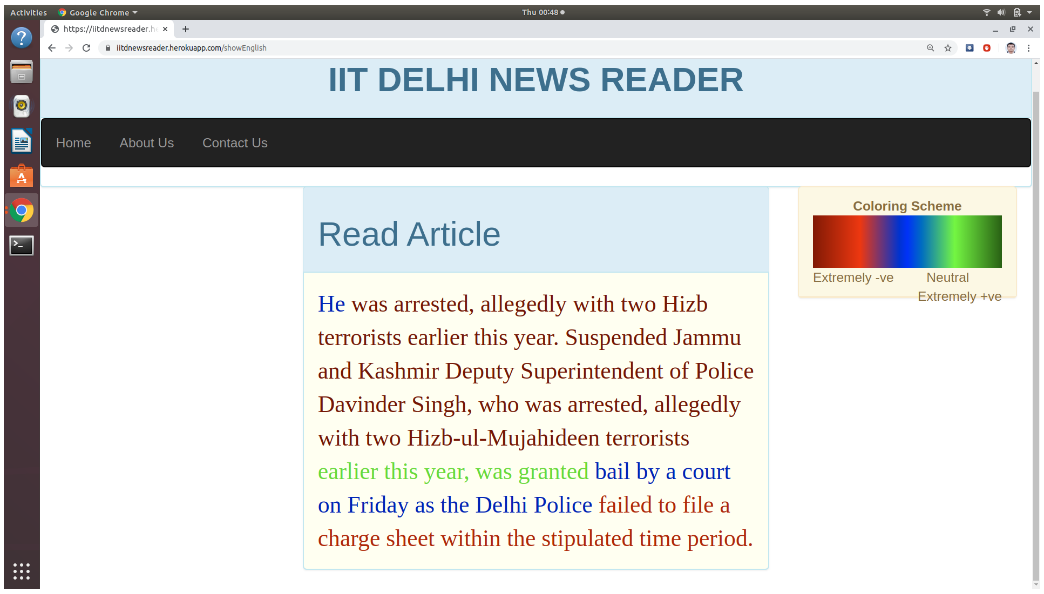
He was arrested, allegedly with two Hizb terrorists earlier this year.Suspended Jammu and Kashmir Deputy Superintendent of Police Davinder Singh, who was arrested, allegedly with two Hizb-ul-Mujahideen terrorists earlier this year, was granted bail by a court on Friday as the Delhi Police failed to file a charge sheet within the stipulated time period.

Figure 5.5: Split sentence wise



He was arrested, allegedly with two Hizb terrorists earlier this year. Suspended Jammu and Kashmir Deputy Superintendent of Police Davinder Singh, who was arrested, allegedly with two Hizb-ul-Mujahideen terrorists earlier this year, was granted bail by a court on Friday as the Delhi Police failed to file a charge sheet within the stipulated time period.

Figure 5.6: Split phrase wise



He was arrested, allegedly with two Hizb terrorists earlier this year. Suspended Jammu and Kashmir Deputy Superintendent of Police Davinder Singh, who was arrested, allegedly with two Hizb-ul-Mujahideen terrorists earlier this year, was granted bail by a court on Friday as the Delhi Police failed to file a charge sheet within the stipulated time period.

Figure 5.7: Split chunk wise

Chapter 6

Conclusion

We have explored to a greater depth in our work. Still there are many scopes left to be further explored. The chunking algorithm can be further optimized and chunking can be done solely based upon the clients requirement. We have used character level embedding for our purpose while training the mLSTM language model, word level or sentence level embedding can be definitely a way which should be tried on. Not just restricted to Urdu or English, The work can be extended to many other languages. The classifier can be further trained on more number of labelled chunks to improve further on accuracy. The language model can also be trained with more number of raw news articles to diversify the scope. Lastly, as discussed, semi-supervised learning approach to generate more labelled chunks is one of the fields which one should definitely try for in future work.

Bibliography

[1] O. AI. Unsupervised sentiment neuron. https://openai.com/blog/unsupervised-sentiment-neuron/.

[2] I. M. S. R. Ben Krause, Liang Lu. mlstm - multiplicative long shot term memory. https://arxiv.org/abs/1609.07959.

[3] Google. Google translator for translating from one language to the other. https://translate.google.co.in/.

[4] F. B. Hasim Sak, Andrew Senior. Lstm - long shot term memory. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf.

[5] IBM. Watson translator for translating from one language to the other. https://www.ibm.com/demos/live/watson-language-translator/self-service.

[6] IEEE. Support vector machine. https://ieeexplore.ieee.org/document/708428.

[7] IEEE. Support vector machine - naive bayes. https://ieeexplore.ieee.org/document/7820655g.

[8] G. H. Ilya Sutskever, James Martens. mrnn. https://icml.cc/Conferences/2011/papers/524_icmlpaper.pdf.

[9] K. L. K. T. Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805.

[10] Stanford. Naive bayes stanford article. https://web.stanford.edu/class/archive/cs/cs109/cs109.1178/lectureHandouts/210-naive-bayes.pdf.

[11] D. P. P. Talukdar. Junto label propagation toolkit. https://talukdar.net/.

[12] Yandex. Yandex translator for translating from one language to the other. `https://translate.yandex.com/`.