A Deep Learning Approach towards Cross-Lingual Tweet Tagging

Nikhil Bharadwaj Gosala
BITS Pilani, Hyderabad
nikhil.gosala@gmail.com

Shalini Chaudhuri
BITS Pilani, Hyderabad
shalini_chaudhuri@yahoo.co.in

Monica Adusumilli
BITS Pilani, Hyderabad
adusumillimonica@gmail.com

Kartik Sethi
BITS Pilani, Hyderabad
kartik1295@gmail.com

ABSTRACT

Named Entity Recognition (NER) is important in analysing the context of a statement and also the sentiments associated with it. Although Twitter Data is noisy, it is valuable due to the amount of information it can provide. Therefore, NER for Twitter Data is necessary. Our model aims to extract the named entities from tweets using a Recurrent Neural Network Core. Long Short Term Memory (LSTM) was used to learn long term dependencies in our supervised learning model. The sequence-to-sequence architecture was used in the implementation of our supervised learning model.

CCS Concepts

•**Information systems** → **Information extraction;**

Keywords

Recurrent Neural Network, Tweet Tagging

1. INTRODUCTION

Sequence Tagging, especially Named Entity Recognition (NER) has been, for a very long time, a classic NLP task [1]. A lot of research has been directed towards it for the past couple of decades. The output of the NER module, the tagged entities, play a significant role in determining the working of many other applications. For instance, these tags are widely used in measuring the sentiment in a sequence of posts, finding the context of a message, and identifying key elements referred to in a set of documents. These tags could be very generic (such as 'noun') or specific (such as 'name of person') depending on the task at hand. Generic tags are usually helpful in learning the structure and automatically generating new sentences in an unknown language. On the other hand, specific tags are widely used by search engines to generate user and product specific advertisements.

Twitter data stores a lot of information that, when extracted and processed properly, can offer a great deal of knowledge. They are the most up-to-date and inclusive sources of information that is currently available on the internet largely due to its low-barrier of entry, and the wide use of mobile devices [4]. Although tweets follow basic grammar rules, they are both extremely noisy, and difficult to comprehend. Due to this very basic nature of tweets, many traditional NER tools fail miserably in tagging them. On the contrary, the human brain does a great job in making sense of these kinds of tweets. This is the motivation behind the choice of exploring the field of Artificial Neural Networks.

Thus, to make an effort in efficiently tagging tweets by modelling a very basic version of the human brain, Recurrent Neural Networks, especially the Long Short Term Memory (LSTM) model was used.

2. RELATED WORK

Most of the existing NER taggers are based on linear statistical models like the Hidden Markov Model [3] and Conditional Random Fields [2]. More recently, owing to its promising results in sequence tagging tasks, Convolutional Neural Networks has gained a lot of attention for the task of Named Entity Recognition. The use of RNN and especially LSTMs have been extensively discussed by [1] wherein they demonstrate the amazing performance of RNNs.

3. SYSTEM DESCRIPTION

The aim of the task was to tag twitter data that contained a mixture of both Hindi and English tokens. The system can be subdivided into the following three modules:

1. Pre-Processing
2. RNN Core
3. Post-Processing

3.1 Pre-Processing

Pre-processing is an inevitable step that should be adopted before processing any kind of data to remove the unwanted values and reduce the noise in the dataset. Adhering to its definition, the pre-processing phase was used to clean and structure the data into a form that could be read by our Tagging Model. The pre-processing phase comprises of the following stages:

1. **Removal of HTML Escape Characters:** It was observed that many of the HTML characters were not replaced by their system equivalent characters. For example, *&* was present instead of the normal *&* token. Such HTML escape characters were taken care of by using the 'html' package in Python.
2. **Tweet Tokenization:** The tweets were tokenized using Regular Expressions. As is with Twitter Data, some words/tokens can easily be tokenized by looking

at their regular expression. For instance, Twitter Handles always start with an @ and any token that starts with @ is a twitter handle. This observation helped us in tokenizing the twitter data with great effect. The following are a list of all the regular expressions used to tokenize the tweet:

(a) **Emoticons:** r"(?: [:=;] [oO\-]? [D\)\]\(\]/)"

(b) **HTML Tags:** r'<[^>]+>'

(c) **Twitter Handles:** r'(?:@[\w_]+)'

(d) **Hash Tags:** r'(?:\#+[\w_]+[\w\'_\-]*[\w_]+)'

(e) **Whitespaces:** r'[\n\t\r]+'

(f) **URLs:** r'http[s]?://(?:[a-z]|[0-9]|[$-_@.& +]

(g) **Numbers:** r'(?:(?:\d+,?)+(?:\.?\d+)?)'

(h) **Words with ' and -:** r"(?:[a-z][a-z'\-_]+[a-z])"

(i) **Other Words:** r'(?:[\w_]+)'

(j) **Everything Else:** r'(?:\S)'

3. **Stop Word Removal:** Stop Words are those words that occur far too frequently to have any effect on the classification or tagging task. Stop words were tackled by using the Stop Words corpus from NLTK and appending it with words and punctuations that occur far too frequently.

4. **Unicode Emoji Removal:** Some of the emojis could not be captured using the regular expressions. For instance, ■Ÿ‘a and ■Ÿ‘a, could not be captured using regular expressions. For such cases, Unicode ranges were used to strip the tweet of emojis.

5. **Rule Tagging:** Owing to the structure of twitter data, some of the tokens can be directly tagged based on Regular Expressions. For instance, any token that begins with a # can be categorized as a Hash Tag and any token that begins with @ is a Twitter Handle. Such tokens were tagged using regular expressions and custom tags. The custom tokens added to the corpus were - *HTML_TAG, TWITTER_HANDLE, HASH_TAG, WHITESPACE, URL, EMOTICON, RT and OTHER.*

6. **Common Misspelling Mapping:** Owing to the 140-character limit of Twitter and the widespread use of SMS lingo, many tweets consist of common SMS lingo. For example, the word for is commonly written as 4 and the word because is written as *bcoz, coz, bcz* and so on. All such misspellings were mapped to the correct spelling of the word. This was done to reduce the number of unique words in the corpus.

7. **Token List Generation:** The pre-tagged text was added to different lists based on the tag. These lists were used later in the tagging process to tag tokens that could not be tagged by the model.

The output of the Pre-Processing module was a *vert* file that is commonly used in many commercially available Part-of-Speech taggers.

3.2 RNN Core

Upon studying various models for NER tagging, Deep Learning and especially Recurrent Neural Networks (RNNs) was chosen for the task of Tweet Tagging. In RNN, there were multiple models available and of all the models, we decided to go with the Sequence-to-Sequence (seq2seq) model. In seq2seq model, each input token has a corresponding tag associated with it. This feature of seq2seq model was consistent with that of the twitter data provided and thus was used for tweet tagging (Each token in the twitter data had a corresponding tag. If it did not, we assigned a custom tag to it).

LSTMs are special kind of RNNs that are capable of learning long-term dependencies. An LSTM cell has multiple gates that define which data to be retained and which data to be forgotten. By training the weights of these gates, one can control the amount of data to be retained and the amount of data to be forgotten. In our implementation, each node in the RNN was a GRU cell. A GRU cell is very similar in function to an LSTM cell, but is computationally much more efficient. Keeping efficiency in mind, GRU cell was chosen over an LSTM cell.

A RNN consists of multiple hidden layers and each layer contains multiple nodes. Each node is like a neuron in the human brain that can retain some information and can make decisions based on this information. The complexity of the model can be varied by changing the number of nodes per layer or the number of hidden layers in a model.

The supervised approach was chosen to train the RNN. In the supervised model, the desired output (target data) is provided along with the training data. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are propagated back through the system, causing the system to adjust the weights that control the RNN. This process occurs multiple times and the weights are continuously tweaked. For reducing the error, Adam Optimizer was used instead of the more common Gradient Descent or the Stochastic Gradient Descent Algorithm. This decision was made because the TensorFlow implementation of Adam Optimizer uses the *moving averages of the parameters* to tweak the weights. The main advantage of this approach is that it has a large step size and thus converges much faster than the Gradient Descent Algorithm.

3.3 Post-Processing

After the model was trained, it was used to predict the data. But because the model was not 100% accurate, some of the tokens were left untagged (i.e. they were tagged with a custom token *OTHER*). These tokens were then checked with the Token Lists and any untagged token that was found in the token list was tagged with the corresponding token.

This step also included removal of tags that were not a part of the annotated tags file. For instance, custom tags like *URL, HASH_TAG, TWITTER_HANDLE* and so on were removed from the final output file to keep the output file consistent with that of the given annotated file.

Apart from the earlier steps, this step also included a function to merge any two consecutive tokens having the same tag into a single word. As an example, *Nikhil Bharadwaj* would have been composed of two tokens *Nikhil* and *Bharadwaj* with the same tag. Because both these tokens are consecutive and have the same tag (*NAME*), they are merged and the phrase *Nikhil Bharadwaj* is tagged as *NAME*.

4. EVALUATION AND RESULTS

Two runs were performed to tag the twitter data. Both the runs used the RNN model but the parameters i.e. the number of hidden layers and number of nodes per layer were modified. The learning rate in both the runs was set to 0.003 and the decay rate was set at 0.97. In Run 1, 3 layers were used with each layer having 192 nodes and in Run 2, 4 layers were used with each layer having 256 nodes. In the case of the Run 1, the final error, after all the iterations was around 0.6 whereas in Run 2, the final error was around 0.45.

The results obtained by using this model were very encouraging. An accuracy of 59.28% and a recall of 19.64% was achieved with an F1 score of 29.50 in the case of Run 1. For Run 2, an accuracy of 61.80% and a recall of 26.39% was achieved with an F1 score of nearly 37. These numbers show that the more complex model (Run 2) was better in capturing a lot more information than the less complex model.

The future direction of research focuses on improving the accuracy by making the annotated data much more comprehensive. While analysing the tagged output, it was observed that a lot of tokens were tagged as *OTHER*. This was due to the fact that most of the tokens in the corpus did not have any tag. This meant that the model was extremely biased towards assigning *OTHER* to any unknown token. This problem was made less severe by adding custom tags to the data based on Regular Expressions. For instance, any token beginning with a # is destined to be a hash tag. So the token *HASH_TAG* was assigned to it. It was also observed that the words in the corpus were not frequent enough. The model, thus, could not learn a lot of information about such words. This issue could be resolved by either using external tagged data, or by making the corpus much more comprehensive.

5. REFERENCES

[1] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[2] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

[3] A. McCallum, D. Freitag, and F. C. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

[4] A. Ritter, S. Clark, O. Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.