NEURAL GEOLOCATION PREDICTION IN TWITTER

BY

PRAMOD SRINIVASAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor ChengXiang Zhai

ABSTRACT

Inferring the location of a user has been a valuable step for many applications that leverage social media, such as marketing, security monitoring and recommendation systems. Motivated by the recent success of Deep Learning techniques for many tasks such as computer vision, speech recognition, and natural language processing, we study the application of neural models to the problem of geolocation prediction and experiment with multiple techniques to analyze neural networks for geolocation inference based solely on text. Experimental results on the dataset suggest that choosing appropriate network architecture can all increase performance on this task and demonstrate a promising extension of neural network based models for geolocation prediction. Our systematic extensive study of four supervised and three unsupervised tweet representations reveal that Convolutional Neural Networks (CNNs) and `fastText` best encode the textual and geolocational properties of tweets. `fastText` emerges as the best model for low resource settings, providing very little degradation with reduction in embedding size.

To my parents, for their love and support.

ACKNOWLEDGMENTS

Foremost, I would like to thank Professor Chengxiang Zhai for and giving me the opportunity to work on the project. I am grateful for his support throughout my Masters', teaching me various concepts of Text Information Systems and guiding through the milestones. My special thanks to Ismini Lourentzou and Alex Morales for their contribution in this research, for developing my research skills, and for their continued support beyond the scope of this research. Finally, I would like to thank my parents, all of my peers and friends who have been encouraging and supportive throughout the duration of this project.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

With the prevalence of social media applications, an increasing number of internet users are publishing text information online. This influx provides a wealth of both text and social graph information and has thus become a rich source of insights into the people, opinions and events of the world. Popular social media services provide features to declare the location either by geo-tagging posts with GPS-based check-ins or by filling a text field in their profile description. Geolocation – the task of identifying the social media message's location can prove vital to various downstream applications, such as advertising, personalization, event discovery and trend analysis [1]. However such text-based descriptions are often unreliable and imprecise, leading to the issue of location sparsity [2]. Estimating user location is therefore essential for the necessary location annotation.

1.1 Motivation

Geographical information is a vital component for analytics where historical data is used to provide insights. For instance, responses to early signals of disease outbreaks or natural disasters have proven to be more effective with geolocation metadata [3]. Geographical information could be utilized at different granularities, such as the comparison rural and city mental health. However, most social media platforms do not provide geographical metadata. For instance, it is reported that less than 3% of Twitter data is accompanied by geo-coordinates [4], even though the platform supports geolocation metadata. This has restricted the usability of several tweets for location-based services and studies. Other platforms, such as discussion forums and blogs often do not have geolocation capabilities. There is thus a need to develop approaches to predict location information for social media content.

Motivated by the success of Deep Learning techniques such as computer vision, speech recognition and natural language processing, we study the application of neural networks to the problem of geolocation prediction. Recently, Deep Learning has shown good performance on various natural language processing (NLP) tasks, such as language modeling, sentiment analysis, POS tagging, named entity recognition and several others. An added advantage of these methods is that they perform well without the need of domain knowledge in the form of time-consuming feature engineering.

Although Deep Learning has been revolutionizing fields such as vision and language processing, the Information Retrieval communities are only beginning to leverage such approaches. Moreover, there is an increasing interest in both industry and academia on tapping the applications of neural network architectures to Social Media [5, 6].

The primary contribution of the work is the large scale comparison of state-of-the-art deep learning architectures. Our experiments have shown that carefully designed architectures can achieve better performance than a simple use of popular neural models such as auto-encoders. As most applications of deep learning advance by improving known models, this work serves as a good starting point that could benefit DNN practitioners to identify areas for further improvements in "neural geotagging".

1.2 Outline

The outline of the rest of this thesis comes in the following structure:

- We begin with a brief overview of the previous work done in geolocation prediction in Chapter 2.
- Then in Chapter 3, we introduce the notion of neural geolocation prediction, as well as describe the characteristics of the dataset;
- In Chapter 4, we presents the various architectures developed to detect location entities from tweets.
- We perform experiments to evaluate our text-based models and analyze the parameter tuning schemes in Chapter 5.

CHAPTER 2

RELATED WORK

Several researchers have studied how social media content such as tweets and posts can be used to predict user location. Almost all these geolocation prediction algorithms are primarily dependent on two common parameters: (1) a definition of what comprises a relationship in Twitter to create the social network, and (2) a source of ground truth location data to use in inference [4]. Following, we discuss the algorithms, how these different sources of information are used, and their complexities.

2.1 Geolocation inference based methods

One of the early works in location prediction was by Quercini et al [7]. The intuition is that if a place is frequently mentioned by users in their tweets, they are likely tweeting from that region. Gazetters were developed as geographical references to identify places that are known regardless of their geographic location in text to geo-tag news articles. Despite the premature success on longer and more homogeneous documents, there are several issues in applying this approach to social networks. For instance, its performance is impeded by the nature of tweets : they are short and informal, and moreover the chances of a user not mentioning the gazetted places in their tweets is very high.

2.2 Generative methods

Cheng et al [2] proposed a generative model for city-level geolocation of U.S Twitter users that identifies words in tweets with a strong local geo-scope (location-indicative words). In their approach, they considered tweets from a set of users belonging to a set of cities across the United States. They estimated the probability distribution of terms used in these tweets, across the cities considered in their data set. This probability distribution is then used to estimate the user location, given the set of terms used by a user, in his/her tweets. Experimenting with various smoothing techniques, their methods calculate the posterior probability of a user being from a city given his/her tweets. However, the tweet messages are considered as independent entities and relationship between different tweet messages, such as reply-tweet messages of the same user has been ignored. This failure to leverage such relationship information has an impact on the distribution of terms across the cities considered.

Eisenstein et al. created a geographic topic model by treating tweets as documents generated by two latent variables [8], i.e., topic and region. The problem is formulated as both a regression task that predicts geographical coordinates and a classification task, where labels are either the 48 contiguous U.S. states or Washington D.C. or a division between regions (West, Midwest, North-east and South). Their experiments included topical modelling, k-nearest neighbors, and several statistical methods such as LDA and regression. Although the model is limited to small datasets due to computational efficiency, an important contribution of this work is the creation of the first dataset available for the geolocation prediction task. However one has to deal with sparsity issues such as class imbalance due to the relatively small size of the dataset.

There have been several subsequent studies using generative models [9], discovering a fixed hierarchical structure over context, via a merging of global topics and regional languages. Extending this idea, Ahmed et al. [10] consider the relations between regions and propose a Chinese Restaurant Franchise based model to study users' geographical topics. Specifically, there is a hierarchical organization of the regions where regions in upper levels geographically encompass those in lower levels, and therefore are more diverse in terms of topics. A tweet defines a path from the root region to the leaf region, where its coordinates are sampled based on the Gaussian distribution of the leaf region [11], and its text content is generated based on the topics and language model of the leaf region.

2.3 Geodesic methods

Wing and Baldridge [12] divide the geographic surface of the Earth into uniform grids and then construct a pseudo-document for each grid. By controlling the granularity of a grid, document similarity is computed using language models and a nearest neighbors' approach is employed for location prediction.

Uniform grids do not take into account the skewness of the pseudo-document distribution; case in point being metropolitan areas typically cover most of the tweets, whereas rural areas face the issue of sparsity. Roller et al. [13] address this issue by constructing grids using a k-d adaptive tree, which enables the creation of more balanced pseudo-documents. They experiment on two datasets of geotagged tweets as well as dataset of geotagged English Wikipedia articles. A limitation with this work is that it is unable to discover shared structures in a location, without explicitly controlling the grid sizes. More recently, Wing and Baldridge [14] have showed the effectiveness of using logistic regression models on hierarchy of nodes in grids.

Han et al. [15] investigated several feature selection methods for identifying location-indicative words, such as Information gain ratio, geographic density and Ripleys K-statistic, as well as the impact of several additional features, such as non-geo-tagged tweets and metadata on predicting the city of a Twitter user or the actual coordinates. They discussed the effectiveness of these features on both regional and global datasets and analyze how user behavior can impact geolocation prediction.

Cha et al. [16] expanded upon Eisenstein's previous efforts through the use of unsupervised sparse vector training and supervised classification to predict user location based on k-nearest neighbor tweets with a cosine similarity measure. Using the Eisenstein corpus, Cha improved upon the accuracy results by 9% and 14% on the 4-way regional and 48-way 50 state level classification tasks, respectively. Their semi-supervised approach has shown state-of-the-art results for the GeoText dataset. However, the performance increase is due to incorporating word order information, i.e. word sequences, and therefore cannot be applied to already the already preprocessed datasets.

2.4 Neural network based methods

Liu and Inkpen [6] created a Neural Network architecture for the geolocation task. A three layer stacked denoising auto-encoder, paired with great-circle distance a loss function and early stopping is tested on Twitter datasets. Model training is based on backpropagation and stochastic gradient descent. The input to both models is a vector space representation for the text including the frequency counts for the 5,000 most frequent term unigrams, bi-grams and trigrams. The authors have analyzed two similar models for the task of estimating Twitter users' locations, namely one model that predicts the US state, and a second model that predicts the latitude and longitude coordinates of the user's location. Although this is the first approach leveraging deep neural network architectures to solve the geolocation task, little insight was given on how the choice of different components affects performance, for example the activation function, the number of layers, pre-training or parameter tuning. Besides, the datasets suffer from label imbalance which inturn had a negative effect on statistical classifiers, and adversely affected regression models because many target values were not sampled.

2.5 2016 Workshop on Noisy User Generated Text

As part of the 2016 Workshop on Noisy User-generated Text[1], a shared task to infer geolocation for Twitter posts and users was organized [17]. Participants were given a large training set collected from over one million users. The goal was two-fold: (1) infer the location for a given English Twitter post (tweet-level prediction); and (2) for a Twitter user (authoring primarily English Tweets), infer the users base city (user-level prediction).

[1]http://noisy-text.github.io/2016/geo-shared-task.html

CHAPTER 3

PROBLEM DEFINITION

There are four key components to a geolocation prediction system, which we discuss in this section and the following sections - (1) the models, (2) the dataset, (3) the evaluation methodology and (4) the evaluation metrics.

3.1 Models

The models developed for the geotagging task should capture the linguistic similarities among Twitter users by learning representations for users, also known as “user embeddings” alongside with the tweet embeddings. Geolocation task can be bolstered by unsupervised techniques on unlabeled social media. In machine learning terms, the task can be expressed as two distinct problems – classification task which puts each user into a geographical region [6] and regression predicts the most likely location of each user in terms of geographical coordinates.

The regression problem is defined by assigning a latitude/longitude tuple to a given input tweet and measuring the distance between the predicted co-ordinates and the true GPS labelled point. For the prediction task, the models produce latitudes and longitudes of a location. The objective function is defined as the great-circle distance between the estimated and actual coordinates. We can approximate the great-circle distance between any two locations on the surface of earth, using the Harvesine formula – the equation is given where Latitude is ϕ, Longitude is λ, and the Earth’s radius is r.

$$d_{gc} = 2r \arcsin\left(\sqrt{\sin^2\frac{\phi_2 - \phi_1}{2} + \cos\phi_1 + \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{3.1}$$

where,

- r is the Earth radius
- ϕ_1, ϕ_2 and λ_1, λ_2 are the latitude and longitude of the predicted and true coordinates
- d_{gc} is the final calculation of the distance, and consequently our error loss function.

The problem of geolocation prediction can also be cast as a text classification task, i.e. tweets from a particular geographical territory can be also be grouped to represent a class. As shown in Figure 3.1, models can be developed to predict the location from a set of pre-defined mutually exclusive classes such as the contiguous states of U.S. or a designated set of metropolitan city centres, thereby becoming a multi-class classification problem [17]. For this task, we used categorical cross entropy as the objective function. When the output layer activation is the softmax function, categorical cross entropy can be interpreted as the negative log likelihood or the KL-divergence between the output distribution and the target distribution, and is a typical loss function used in the deep learning literature.

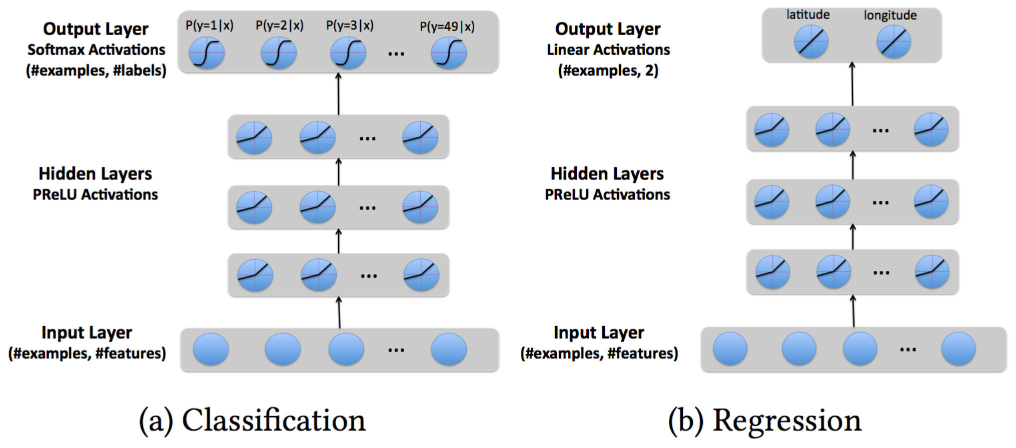

Figure 3.1: Neural models for Geolocation

The publicly-available Geoname dataset[1] was chosen as the the basis for the city classification. Geoname consists of city-level metadata, including the

[1]http://www.geonames.org/

full city name, population, latitude and longitude. The city name is associated with hierarchical regional information, such as its state and country it is located in, so as to distinguish London in Britain from London in Canada.

The loss functions are defined without regularizing the weights; to prevent overfitting, the early stopping technique is adopted [18] i.e., training stops when the models performance on the validation set no longer improves. Early stopping is an inexpensive way to avoid overfitting because even if the other hyper-parameters would lead to overfitting, early stopping has proven to considerably reduce the overfitting damage that would otherwise ensue. Specifically, we adopt the patience heuristic which defines the minimim number of training examples to be observed before deciding to stop training [19]; the pseudocode is shown below :

```
initialization
patience = 20, iteration = 1;
while iteration < patience do
    update parameters;
    if the performance improves then
        patience := max(patience, iteration * 2);
    end
    iteration += 1;
end
```

Algorithm 1: Early stopping

3.2 Dataset

We focus on social media posts from the website Twitter, which are an excellent testing ground for both word-level and character-level based models due to the noisy nature of text. Heavy use of slang and abundant misspellings mean that there are many orthographically and semantically similar tokens, and special characters such as emojis are also immensely popular and often convey useful semantic information.

The experiments are conducted on the WNUT dataset [14], which comprises the geotagged English language tweets from 2013 to 2016. In total, there were 1 million users for training, and 10K users for development and test, respectively, with every user assigned a class label to denote his or her primary location. These class labels have been extracted from metropolitan city centres in GeoNames [15] – a total of 2998 cities. We present the distribution of these classes in Fig 3.2. For reference, the top four labels with the majority of training data correspond to cities such as New York, Los Angeles and San Francisco, leading to a class imbalance problem [2]. The training data for the tweet-level task is based on a unique tweet from each of these 1 million users. The development and test data sizes are both 10K messages, different from the user-level development and test data.

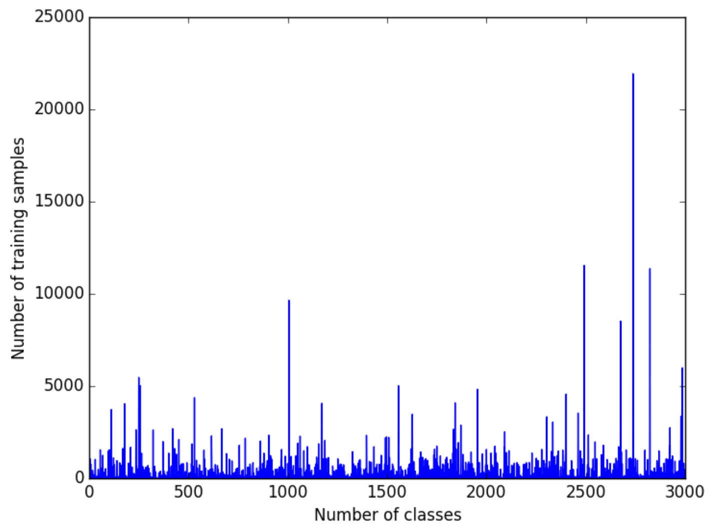

Figure 3.2: Distribution of training data across the cities

All tweets were filtered by Twitters language ID code, so as to include only text labelled as English. This was accomplished in two passes: in the first round, all users with geotagged tweets were selected; and in the second pass, users with atleast 10 geotagged tweets. The authors state that 10 geotagged tweets was an empirical requirement in order to obtain a reliable prediction of a users primary location and at the same time excluding the incorporation of bot-generated messages.

The training set consists of 12.8M tweets, the user-level development and test sets consist of 128k and 99k tweets, respectively, and the tweet-level development and test sets each contain 10k tweets. Figure 3.3 shows the spread of the WNUT training dataset in terms of the number of geo-tagged twitter posts by users in a given city. This analysis shows that most cities have a relatively small number of geo-tagged tweets.

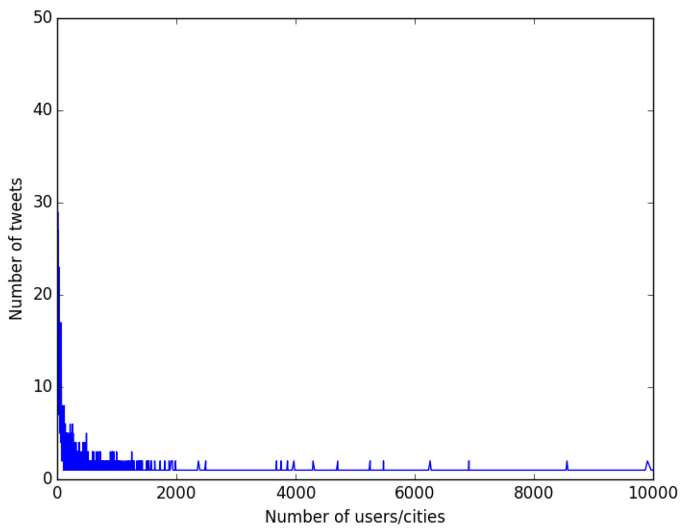

Figure 3.3: Statistical profile of the WNUT dataset

3.3 Evaluation methodology

The evaluation of the models can be done on two levels: tweet-level and user-level. The tweet-level is more practical in real world applications, as a tweet is a basic text unit created by a Twitter user and often is associated with a unique location. However, as a single tweet text can be brief and may not contain explicitly geolocation information, another popular setting has been user-level prediction, after aggregating a given users tweets. This is based on the assumption that every user has a primary location and that the primary location can be inferred from the aggregated tweet data. Both these approaches leverage the abundant high-quality geographical information offered by tweet metadata.

3.4 Evaluation metrics

To evaluate our models, we use the following evaluation metrics:

- **Accuracy:** The proportion of tweets (and users) that is correctly classified to their home location (city), out of all tweets (and users). This metric allows us to measure the correctness of our prediction algorithm in terms of percentage of true labelled cities.

- **Mean Error Distance:** The average error, in terms of distance, between the predicted cities and the ground truth cities of the tweets (and users). Even for mislabelled cities, a mislabelled city nearer to the ground truth city is deemed better, e.g., New York mislabelled as Chicago, is considered better than New York mislabelled as London. The mean error distance, measured in kilometers, aims to capture this aspect.

- **Median Error Distance:** The median error, in terms of distance, between the predicted cities and the ground truth cities of the tweets (and users). Similar to the Mean Error Distance, except that we are measuring the error distance in terms of median values.

The classification accuracy determines how well a given system performs in a *hard* classification task setting, in terms of whether they correctly predict the city or not. In addition to the accuracy, a number of evaluation metrics which capture spatial proximity are considered because they provide a *soft* evaluation as they reward near-miss predictions and penalize wildly long predictions.

CHAPTER 4

MODELS

In this chapter we describe the various models chosen for comparison. For each model, we give a brief summary of the architecture, while comparing and contrasting various character and word based approaches for both regression and classification tasks. Some models such as Convolutional neural networks capture the local contextual structure whereas recurrent neural networks are designed to model the global structure of the social media posts.

4.1 Unsupervised learning

Unsupervised learning tasks have no specific goals, which means the data are not labeled. A typical example of unsupervised learning is clustering, which aims to divide the whole dataset into multiple subsets such that one training example is more similar to training examples in the same subset than to those in other subsets. Below we list the set of unsupervised representation learning models which require an additional classifier in general to do the final classification.

4.1.1 Skip Thought Vectors

Kiros et al. created skip-though vectors [20], an encoder-decoder architecture which encodes whole sentences into a vector space, in a manner similar to the skip-gram method for learning word embeddings. It is based on the idea that a word's meaning is embedded by the surrounding vectors. Sentences that share semantic and syntactic properties are mapped to similar vector representations. The encoder maps the input sentence to a fixed-length vector representation and a decoder generates the sentences surrounding the original sentence – the encoded vectors are called skip-thought vectors.

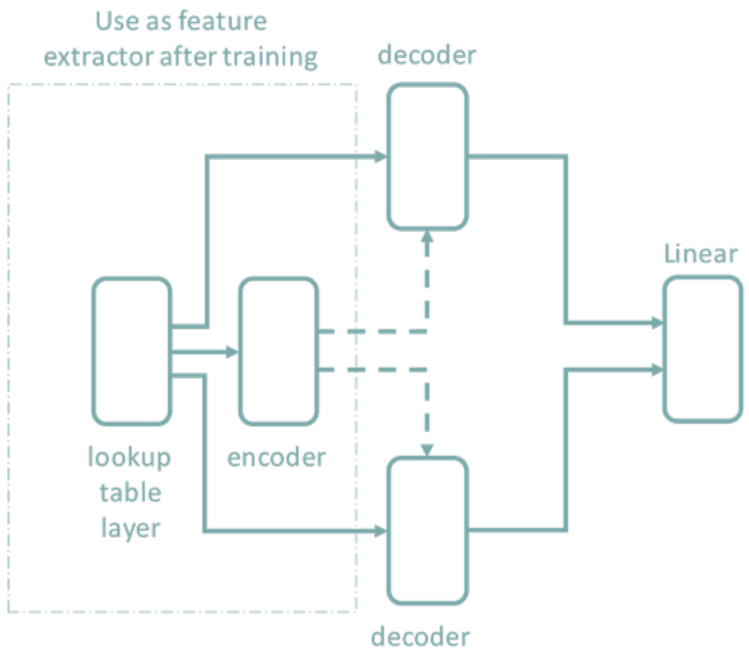

Figure 4.1: Skip-Thoughts Network architecture [21]

The encoder is built either using a recurrent neural network (RNN) layer or a bidirectional layer is able to capture the temporal patterns of sequential words vectors. The encoder uses a lookup table layer also known as an embedding layer to convert each word into a vector. The inputs represented as hidden states of the encoder are fed into two separate decoders - each to predict the previous and subsequent sentences. The decoders in turn use another set of recurrent layers, which share the same look-up table layer with the encoder as shown in Fig 4.1.

4.1.2 Stacked Denoising Auto-encoders

Autoencoders are trained to extract significant features by compressing input data to low-dimensional vectors. A denoising encoder is a stochastic version of the auto-encoder. As it reconstructs the input from its corrupted version, it aims to preserve the original information as well as undo the effect of the corruption process as shown in Fig 4.2.

Because each auto-encoder learns an abstract representation of the input, stacking a number of auto-encoders is likely to generate even more abstract representations, which in turn could improve the performance of the task.

Denoising autoencoders can be stacked in a greedy layerwise fashion for pre-training the weights of a neural network [22]. The outputs of each layer are wired to the inputs of the successive layer. The unsupervised pre-training, done one layer at a time captures a useful hierarchical grouping by minimizing the error in reconstructing the input. After the pre-training process, the network undergoes a second stage of training called fine-tuning in order to minimize the prediction error on a supervised task.

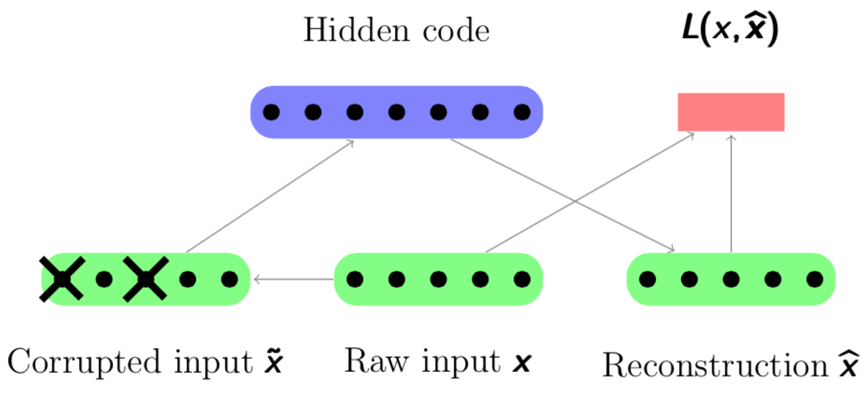

Figure 4.2: A graphical representation of the denoising autoencoder [23]. After the input x is corrupted to $\widetilde{x}$, the auto-enconder maps it to the hidden representation $\mathbf{h}$ and attempts to reconstruct x.

4.1.3 Variational Auto-encoders

Variational auto-encoders is a directed probabilistic graphical model based on a regularized version of the standard autoencoder that learns the parameters of a probability distribution modeling the input data. The VAE learns sentences not as single points, but as soft ellipsoidal regions in latent space, forcing the sentences to latent the space rather than memorizing the training data as isolated sentences. It is this constraint that differentiates a variational encoder from other standard encoders.

These parameters are trained via two loss functions : a generative loss which is the mean squared error E_q that measures how accurately the network reconstructs the input text and a latent loss, the KL-divergence between the learned latent distribution and the prior distribution which act as a regularization term.

Formally, the generative model $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$ is trained for data

$\mathbf{x}$ using latent variables $\mathbf{z}$ and an inference model $q(z|x,\phi)$ by optimizing a variational lower bound to the likelihood $p_\theta(x)$. The objective is a valid lower bound on the true log likelihood of the data, making the VAE a generative model, as shwon in Equation 4.1.

$$\mathcal{L}(x,\theta,\phi) = -\underbrace{[q(z|x,\phi)||p(z|\theta)]}_{\text{KL term}} + \underbrace{E_q\left[\log p(x|z,\theta)\right]}_{\text{autoencoding term}} \leq \log p_\theta(x) \qquad (4.1)$$

4.2 Supervised models

Below we list the set of supervised representation learning models which are capable of performing end-to-end classification.

4.2.1 Convolutional Neural Networks

The most basic deep learning model is a sequential convolutional neural network that can be trained on on top of word vectors obtained from an unsupervised neural language model. The inputs of the network classifier are preprocessed tweets that consist of a sequence of words. The model converts each word in the sequence into a continuous high-dimensional word vector. Therefore for each sentence, a sentence matrix is obtained as shown in the Fig 4.3. The first layer of the model is a convolutional layer, which applies a filter matrix of $h \times k$ to a window of h words to produce a feature. The i^{th} feature generated from a window of words $\mathbf{x}_{i:i+h-1}$ is represented by:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

The filter is applied to every window of words in the sentence matrix using the sliding window technique. Then, the next layer consists of the produced features which is called a feature map as shown in Equation 4.2.

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \qquad (4.2)$$

As the actual model contains several filters, each resulting in multiple feature maps. Finally, there is max-pooling layer which extracts the most important feature i.e. max value from each feature map before propagating it to the next layer. At the end, those extracted features are passed through a fully-connected layer and softmax layer to produce the probability distribution over class labels. Although the model used a very simple idea of convolutional layers, it produced notable results and achieved the state-of-the-art performance in various sentence classification tasks.

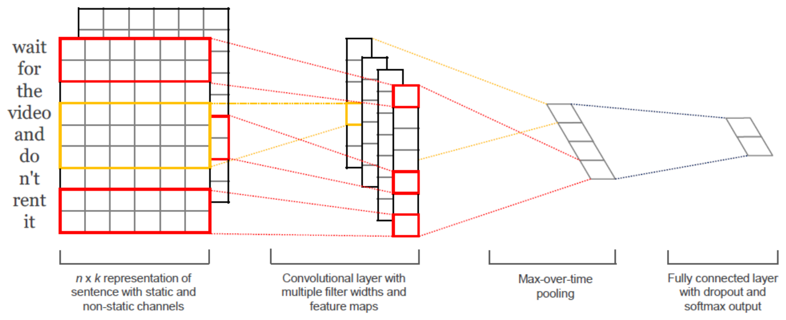
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 4.3: The Sequential CNN model architecture [24]

This model is considered sequential because the filters in the convolutional layer are applied to words sequentially in the order as they appear in the original input sentence. Thus, sequential CNNs present some limitations in their capability to capture structural features as well as long-distance dependencies of the sentence.

4.2.2 Long Short Term Memory Networks

LSTMs are recurrent neural networks that can be trained to construct low-dimensional vector representations of sentences from word embeddings [25]. LSTMs were introduced to address the problem of vanishing gradient problem – gradients must flow from later time steps of the sequence back to earlier parts of the input. This is difficult for long sequences, as the gradient tends to degrade, or vanish as it is passed backwards through time.

As shown in the Figure 4.4, the LSTM introduces a memory cell structure, governed by three gates in addition to a hidden state vector. An *input gate* is used to control how much the memory cell will be influenced by the new input; a *forget gate* dictates how much previous information in the memory cell will be forgotten; and an *output gate* controls how much the memory cell

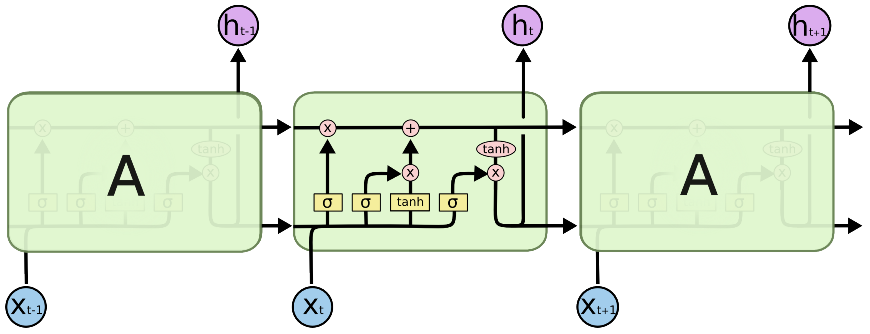

Figure 4.4: The various interacting layers of an LSTM module [26]

will influence the current hidden state. All three of these gates depend on the previous hidden state and the current input.

Here we formally describe the LSTM model [27] . Given an input sequence $X = (x_1, x_2, ..., x_N)$, the hidden vector sequence $h = (h_1, h_2, ..., h_N)$ and output vector sequence $Y = (y_1, y_2, ..., y_N)$ are computed by the LSTM. At each time step, the output of the module is controlled by a set of gates as a function of the previous hidden state h_{t1} and the input at the current time step x_t, the forget gate f_t, the input gate i_t, and the output gate o_t. These gates collectively decide the transitions of the current memory cell c_t and the current hidden state h_t. The LSTM transition functions are defined as follows:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ l_t &= tanh(W_l \cdot [h_{t-1}, x_t] + b_l) \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot l_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{4.3}$$

Here, σ is the *sigmoid* function that has an output in [0, 1], $tanh$ denotes the hyperbolic tangent function that has an output in $[-1, 1]$, and $\odot$ denotes the scalar product. The variable f_t controls the extent to which the information in the old memory cell is discarded, while the extent to which new information is stored in the current memory cell is controlled by i_t, and o_t is the output based on the memory cell c_t. In sequence-to-sequence generation tasks, an LSTM defines a distribution over outputs and sequentially predicts tokens using a softmax function.

$$P(Y|X) = \prod_{t\in[1,N]} \frac{exp(g(h_{t-1}, y_t))}{\sum_{y'} exp(g(h_{t1}, y'_t)} \tag{4.4}$$

where g is the activation function.

4.2.3 FastText

The `fastText` model is an extension of the word2vec [28] that can tackle sentence and document classification. Although it is at par with deep learning models in terms of accuracy, it can be trained an order of magnitude faster on a standard CPU. The input to the model is a sequence of words as shown in Fig 4.5, the word embeddings are averaged into text representations and fed to a linear classifier. It is trained with stochastic gradient descent and backpropagation with a linear decaying learning rate. The text representation can be considered as a hidden state that can be shared among features and classes. The softmax layer represents a probability distribution over a set of pre-defined classes [29].

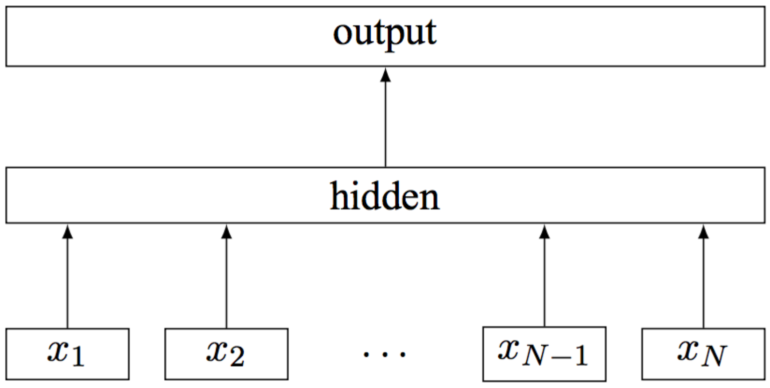

Figure 4.5: Model architecture of `fastText` for a sentence with N ngram features [29]

Although it is built on top of linear models with a rank constraint and a fast loss approximation, the authors have employed several computational tricks to boost its performance. Firstly, hierarchical softmax is used to decrease the training cost of the classifiers. Taking word order into account to track sequential data would add incredible amounts of complexity. However, in `fastText`, the sequential information is preserved without the use of RNNs by using a bag of n-gram features to capture partial information about local word order. Finally, the hashing trick [30] is employed to maintain fast and memory efficient mapping of the n-grams.

4.2.4 Tweet2Vec

This is a character composition model [31] operating directly on the character sequences to predict the user-annotated hashtags in a tweet. The Bi-directional Gated Recurrent Unit (Bi-GRU) does a forward and a backward pass on the entire sequence and the final states of the GRU are linearly combined to obtain the tweet embedding as shown in Figure 4.6. Subsequently, posterior probabilities over hashtags are computed by projecting this embedding to a softmax output layer.

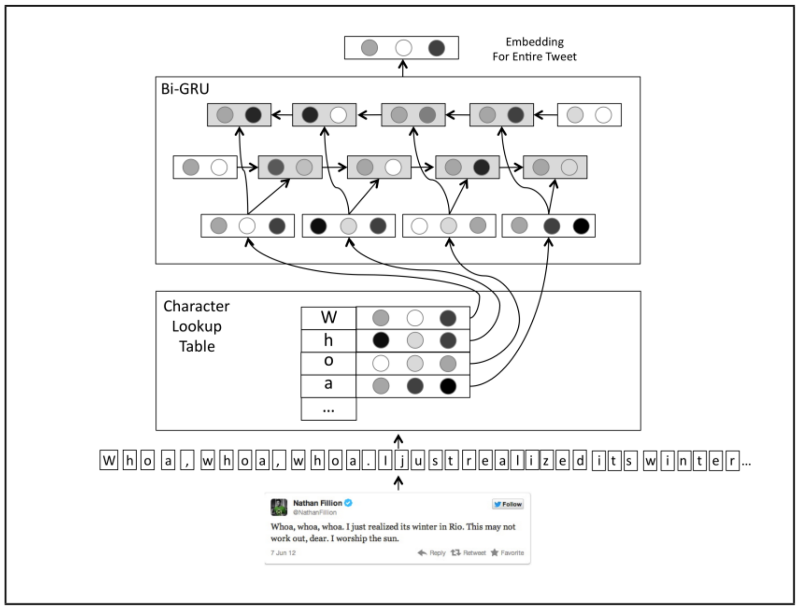

Figure 4.6: Tweet2Vec encoder architecture [31]

Formally, the input to the network is defined by a set of characters $\mathcal{C}$. The input tweet is decomposed into a stream of characters $c_1, c_2, \ldots, c_m$ each of which is represented as one-hot vectors, which are subsequently projected to a character space. Each of the GRU units process these vectors sequentially and start with the initial state h_0 to compute the sequence $h_1, h_2, \ldots, h_m$ as follows:

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ \widetilde{h_t} &= tanh(W_h x_t U_h (r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \end{aligned} \tag{4.5}$$

Here r_t, z_t are called the reset and update gates respectively, and $\widetilde{h_t}$ is the candidate output state which is converted to the actual output state h_t. The final state h_m^f from the forward-GRU and the initial state from the backward GRU are combined using a fully connected layer to give the tweet embedding e_t:

$$e_t = W^f h_m^f + W^b h_0^b$$

Finally the tweet embedding is passed through a linear layer whose output is the same size as the number of hashtags L in the dataset. The softmax layer is used to compute the posterior hashtag probabilities.

4.2.5 Hierarchical Attention Networks

The idea of attention in neural network architectures is loosely based on the visual attention mechanism found in humans. Human visual attention is able to focus on a certain portion of an image with high resolution while perceiving the neighboring image in low resolution, and adjusting the focal point over time. Incorporating attention to a document classification results in better performance because it captures which words and sentences contribute to the classification decision which can be of value in applications and analysis.

The intuition underlying the attention models is that not every portion of a document are equally relevant and that determining the useful sections of text involves modeling the interactions of the words, rather than modeling their presence in isolation.

Hierarchical attention networks are designed to mirror two important ideas in documents - the hierarchical structure where words combine to form sentences and sentences combine to form documents and the fact that different

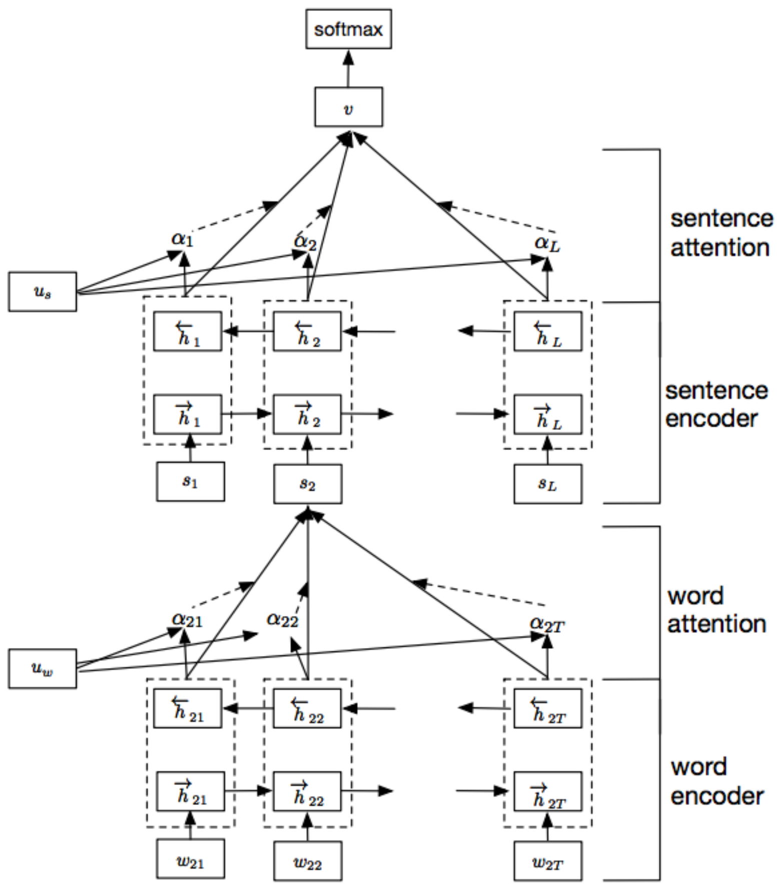

Figure 4.7: Hierarchical Attention Network [32]

words and sentences in a document are differentially informative. The model progressively builds a document vector by aggregating important words into sentence vectors and then aggregating important sentences vectors to document vectors [32].

As shown in Figure 4.7, it consists of a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. The word encoder embeds the words in a given input sentence into word vectors using a Bi-directional Gated Recurrent Unit [33]. The word-level attention layer extracts words that are important to the meaning of the sentence and collects the representation of such informative words to form a context vector. The entire process is repeated to get a document vector in a similar manner. The document vector, v as shown in Figure 4.7 is viewed as a high level representation of the document and can be utilized as features for the document classification task.

CHAPTER 5

EXPERIMENTAL EVALUATION

In this section, we present an evaluation of our models on the WNUT Task [17]. Our high-level goals are to evaluate the performances of various architectures in the geolocation prediction task; in addition, we attempt to understand the impact of the parameters affecting the evaluation metrics and understand the best configuration suitable for this task. All experiments were run on GeForce GTX 1080 graphics cards, with CUDA 8.0, CuDNNv3, and bleeding-edge installations of Keras and Theano. We present our results and compare across the various models that perform the same tasks, i.e. classification and regression at the tweet-level and the user-level.

5.1 Tweet-level tasks

The training data for the tweet-level task is based on a unique tweet from each of the same 1 million users. The development and test data sizes are both 10K messages in size, different from the user-level development and test data.

5.1.1 Classification

Tables 5.1 and 5.2 represent the comparison of the classification of tweets across various models on the development and test datasets respectively. We observe that the Convolutional Neural networks outperform the traditional methods such as Autoencoders and embedding approaches. This is because neural networks capture more contextual information of the features compared with traditional methods based and therefore may be more immune from the data sparsity problem.

It is also observed that the *tweet2vec* model outperforms the autoencoder

Table 5.1: Results on the Development Set for the Tweet Classification Task

Case	Loss	Accuracy (in %)
CNN	**6.00**	**36.8**
`fastText`	8.08	32.9
VAE	6.4	23.2
SDA	7.29	23.1
tweet2vec	7.04	33.08
LSTM	5.16	23.2
HAN	7.8	24.5

Table 5.2: Results on the Test Set for the Tweet Classification Task

Case	Loss	Accuracy (in %)
CNN	**6.19**	**48.29**
`fastText`	8.09	32.76
VAE	6.81	42.8
SDA	7.305	33.16
tweet2vec	7.40	33.04
LSTM	5.16	33.1
HAN	7.8	34.6

models by performing significantly well in the development set and comparably well in the test set. However, the improved performance comes at the cost of increased training time since moving from words to characters results in longer input sequences to the Gated Recurrent Unit (GRU) [31].

We now present the subsequent experimental findings which were employed to improve the performance of the CNN architecture for the WNUT Dataset. Specifically, we have covered the following aspects:

- Interlayer Batch Normalization
- Dropout
- Different Optimization Techniques

Interlayer Batch Normalization

Batch normalization is a simple and effective way to improve the performance of a neural network [34]. It is established that batch normalization enables the use of higher learning rates besides acting as a regularizer thus achieving

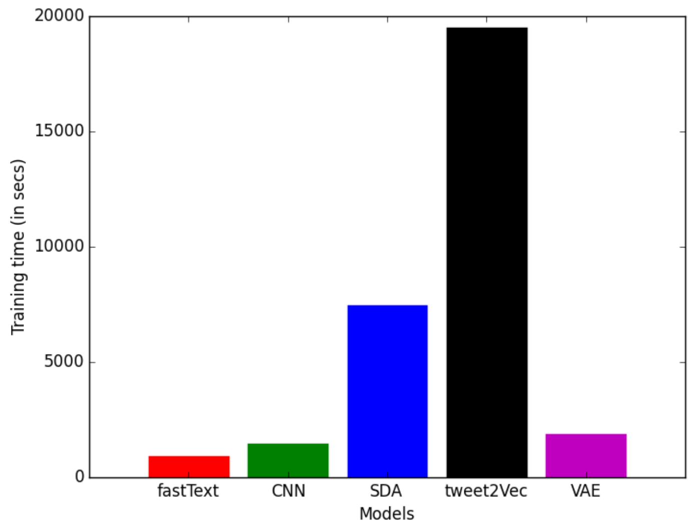

Figure 5.1: Parameter tuning for the tweet classification task

a speed-up in the training process. In the figure 5.2, we observe the positive effect of batch normalization as we increase the number of weight layers. An important observation is that during the test time, Batch Normalization layer functions differently. The mean and variance are not computed based on batch instead a single fixed empirical mean of activations during training is used.

Batch Normalization has clearly improved the gradient flow through the network :

- It also acts a great regularizer and can even serve as a worthy substitute of dropout if need arises.
- It has the potential to reduce dependence on initialization and can allow higher learning rates.

Dropout

As we have discussed earlier, a standard CNN consists of alternating convolutional and pooling layers, with fully-connected layers on top. Compared to regular feed-forward networks with similarly-sized layers, CNNs have much fewer parameters and connections and are therefore less prone to overfitting. We train various CNN models by separately and simultaneously introducing

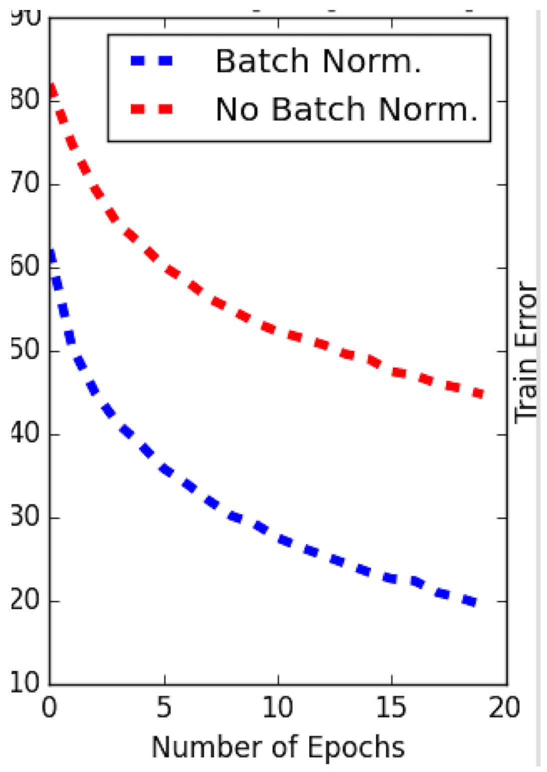

Figure 5.2: Train performance of various Architectures with and without Batch Normalization

dropout with max-pooling as shown in Figure 5.3. Here are some observations, we can infer from this experiment.

- The retention probability p is varied to observe the change in performance.
- There is visible increase in training error after having incorporated dropout.
- The main purpose of dropout is the regularization which can prevent overfitting for a value of p = 0.4, we observe the model performs better than the one with dropout.
- No dropout converges faster than dropout but not necessarily to the global optimum.

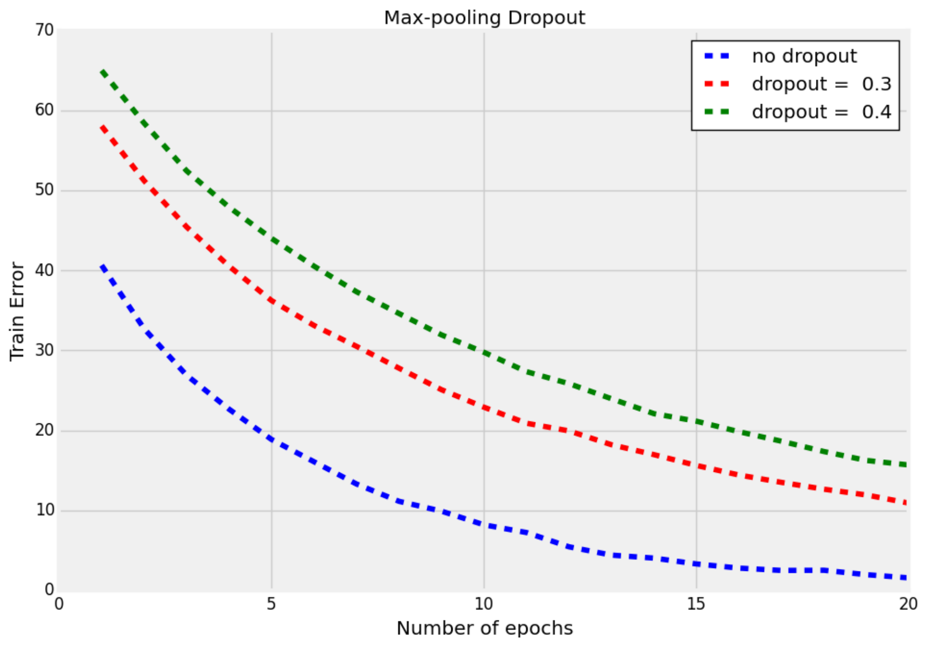

Figure 5.3: Train performance of various Architectures for various dropout probabilities

Different pooling and optimization techniques

By reducing the spatial size of the representation, the pooling layers not only help reduce the number of parameters but also provide a form of translational invariance and help improve generalization. This can help reduce computational time between the successive convolutional layers. The introduction of pooling layers have been also shown to counter the effects of overfitting as depicted in Figure 5.4.

5.1.2 Regression

Tables 5.1 and 5.2 represent the comparison of the classification of tweets across various models on the development and test datasets respectively. Comparing the various approaches, once again the Convolutional neural networks outperform the other approaches. This illustrates that the convolution-based framework is more suitable for constructing the semantic representation of texts compared with previous neural networks.The main reason is that CNN can select more features through the max-pooling layer and capture contextual information through convolutional layer [35].

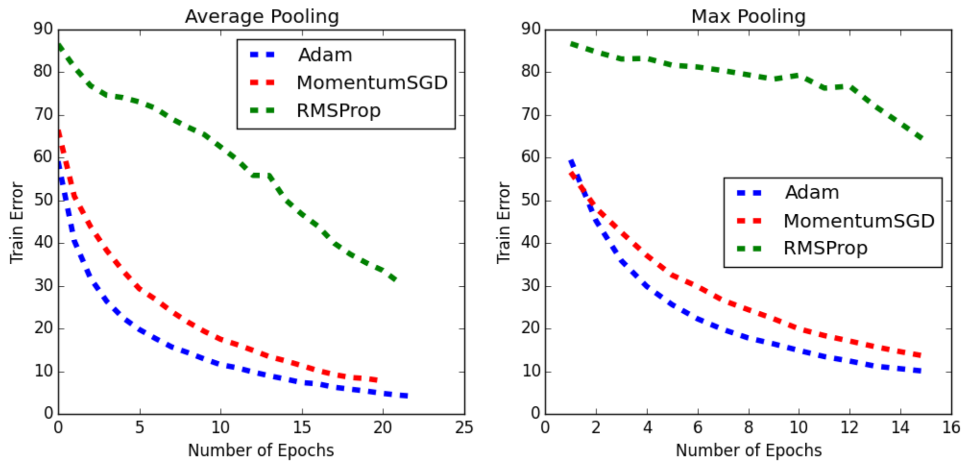

Figure 5.4: Train performance of various Architectures for various optimization strategies

Table 5.3: Results on the Test Set for the Tweet Regression Task

Case	Mean Distance (km)	Median Distance (km)	Acc@161 (in %)
CNN	**5220**	**4260**	**33.03**
`fastText`	8969	9121	32.6
SDA	7033	6123	30.4
tweet2vec	5988	4834	41.8
LSTM	6134	6541	32.4
HAN	3315	3451	23.1

Table 5.4: Results on the Development Set for the Tweet Regression Task

Case	Mean Distance (km)	Median Distance (km)	Acc@161 (in %)
CNN	4563	4413	43.81
`fastText`	6355.83	5187.50	40.71
SDA	5613	6233.83	40.92
tweet2Vec	5188	4134	31.1
LSTM	6243	6721	33.1
HAN	3515	3481	24.1

5.2 User-level tasks

As a single twitter post can be brief and may not contain explicitly geolocation information, another popular setting has been user-level prediction. This is performed after aggregating the set of tweets posted by users. The guiding assumption is that every user has a primary location and that the primary location can be inferred from the aggregated tweet data.

5.2.1 Classification

Tables 5.5 and 5.6 represent the comparison of the classification of tweets at the user level across various models on the development and test datasets respectively. These results indicate that Convolutional neural network based model has high scalability and is a competitive approach in geolocation prediction.

Table 5.5: Results on the Development Set for the User Classification Task

Case	Loss	Accuracy (in %)
CNN	**9.55**	**33.15**
SDA	8.12	23.63
`fastText`	7.778	23.45
tweet2Vec	10.40	23.45
LSTM	12.12	24.5
HAN	11.46	23.1

Table 5.6: Results on the Test Set for the User Classification Task

Case	Dev Loss	Accuracy (in %)
CNN	**9.25**	**35.13**
SDA	6.8531	33.2
`fastText`	7.778	33.45
tweet2Vec	10.65	32.3
LSTM	11.6	34.1
HAN	12.5	22.9

5.2.2 Regression

Tables 5.7 and 5.8 represent the comparison of the classification of tweets at the user level across various models on the development and test datasets respectively.

Table 5.7: Results on the Test Set for the User Regression Task

Case	Mean Distance (km)	Median Distance (km)	Acc@161 (in %)
CNN	5220	4260	23.03
SDA	5429	3906	21.12
`fastText`	**3849**	**5057**	**34.15**
tweet2Vec	4560	4667	22.56
LSTM	5013	3130	21.5
HAN	5312	4561	22.8

Table 5.8: Results on the Development Set for the User Regression Task

Case	Mean Distance (km)	Median Distance (km)	Acc@161 (in %)
CNN	5514	6134	34.66
SDA	5867	4711	30.34
`fastText`	**5786**	**5938**	**38.66**
tweet2Vec	8312	8453	31.45
LSTM	6031	6534	22.67
HAN	5812	5956	23.2

In table 5.9, we show the words with the smallest average distance errors for select cities. Our `fastText` model is able to distinguish several location indicative words, for instance it finds several restaurants local to a given city, for example, 'Chadwick' is a popular Brooklyn restaurant in New York. The model is also able to distinguish vernacular in twitter for those locations. In New York city, there are frequent mentions of clothing brands and their quality, which makes sense as it is often described as the fashion capital of the U.S.; in San Francisco technology terms take precedence owing to its proximity to Silicon Valley. These terms show that the tweets have some dynamic property to them and thus there is scope to incorporate methods that utilize temporal aspects, such as event discovery in order to better learn location indicative terms.

Table 5.9: Selected words with smallest average user errors in WNUT

New York	Dallas	Los Angeles	San Francisco
cashmere	sundance	fitzpatrick	engineers
authenticity	bachmann	2pac	bot
trousers	immigrants	guste	gadgets
chadwick	administration	morningside	workflow
afterparty	follow	good	birthday
wahlberg	brutality	afterhours	unfriend
pearls	socialist	alvaro	https

CHAPTER 6

CONCLUSION

We have investigated various neural network architectures for predicting the location of the Twitter social media users. This is the first systematic study of comparing various deep learning techniques for geolocation prediction, where each model has been trained on an order of magnitude more data than any prior work. Specifically, we explored how the parameters of each architecture affect the evaluation metrics. To compare the various models, we identified three key criteria and corresponding metrics, which fully capture the performance behavior of each method and allow for meaningful comparison, both in this and future work. The results indicate that the deep neural networks are capable of learning representations from raw input data that helps the inference of location of users without having to design any hand-engineered features. It also shows that deep learning models have the potential of being applied to solve real business problems that require location detection, in addition to their recent success in natural language processing tasks and to their well-established success in computer vision and speech recognition.

Future work will encompass the creation of datasets which can capture heterogeneous information in order to facilitate the hyperparameter tuning and for more advanced error analysis. There is also scope for further improvement by exploring unlabeled social media data with unsupervised techniques, such as, pre-training with autoencoders or adding social network and word order information with architectures, such as siamese [36] and recurrent networks [37] that would facilitate such user/word representation and might advance the neural geolocation task. We also plan to work on interpretation of distributed representations of nodes in a social network by leveraging various interesting graph properties.

REFERENCES

[1] M. Dredze, M. Osborne, and P. Kambadur, “Geolocation for twitter: Timing matters,” in *Proceedings of NAACL-HLT*, 2016, pp. 1064–1069.

[2] Z. Cheng, J. Caverlee, and K. Lee, “You are where you tweet: a content-based approach to geo-locating twitter users,” in *Proceedings of the 19th ACM international conference on Information and knowledge management.* ACM, 2010, pp. 759–768.

[3] K. Lee, A. Agrawal, and A. Choudhary, “Real-time disease surveillance using twitter data: demonstration on flu and cancer,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2013, pp. 1474–1477.

[4] D. Jurgens, J. McCorriston, Y. T. Xu, and D. Ruths, “Geolocation prediction in twitter using social networks: A critical analysis and review of current practice.” 2015.

[5] D. Y. Li Deng, “Deep learning: Methods and applications,” Tech. Rep., May 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/

[6] J. Liu and D. Inkpen, “Estimating user location in social media with stacked denoising auto-encoders,” in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, NAACL*, 2015, pp. 201–210.

[7] G. Quercini, H. Samet, J. Sankaranarayanan, and M. D. Lieberman, “Determining the spatial reader scopes of news sources using local lexicons,” in *proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems.* ACM, 2010, pp. 43–52.

[8] J. Eisenstein, B. O’Connor, N. A. Smith, and E. P. Xing, “A latent variable model for geographic lexical variation,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2010, pp. 1277–1287.

[9] J. Eisenstein, A. Ahmed, and E. P. Xing, "Sparse additive generative models of text." in *ICML*, L. Getoor and T. Scheffer, Eds. Omnipress, 2011, pp. 1041–1048.

[10] A. Ahmed, L. Hong, and A. J. Smola, "Hierarchical geographical modeling of user locations from social media posts," in *Proceedings of the 22nd international conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2013, pp. 25–36.

[11] Q. Yuan, G. Cong, K. Zhao, Z. Ma, and A. Sun, "Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users," *ACM Transactions on Information Systems (TOIS)*, vol. 33, no. 1, p. 2, 2015.

[12] B. P. Wing and J. Baldridge, "Simple supervised document geolocation with geodesic grids," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* Association for Computational Linguistics, 2011, pp. 955–964.

[13] S. Roller, M. Speriosu, S. Rallapalli, B. Wing, and J. Baldridge, "Supervised text-based geolocation using language models on an adaptive grid," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.* Association for Computational Linguistics, 2012, pp. 1500–1510.

[14] B. Wing and J. Baldridge, "Hierarchical discriminative classification for text-based geolocation." in *EMNLP*, 2014, pp. 336–348.

[15] B. Han, P. Cook, and T. Baldwin, "Text-based twitter user geolocation prediction," *Journal of Artificial Intelligence Research*, pp. 451–500, 2014.

[16] M. Cha, Y. Gwon, and H. Kung, "Twitter geolocation and regional classification via sparse coding," in *Proceedings of the 9th International Conference on Weblogs and Social Media (ICWSM 2015)*, 2015, pp. 582–585.

[17] B. Han, A. Hugo, A. Rahimi, L. Derczynski, and T. Baldwin, "Twitter geolocation prediction shared task of the 2016 workshop on noisy user-generated text," *WNUT 2016*, p. 213, 2016.

[18] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[19] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade.* Springer, 2012, pp. 437–478.

[20] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[21] NervanaBlog, "Building skip-thought vectors for document understanding," https://www.nervanasys.com/building-skip-thought-vectors-document-understanding, 2017, accessed: 2017-01-05.

[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[23] Y. Zhang, "Stacked denoising autoencoders," http://psyyz10.github.io/2015/11/SDA/, 2015, accessed: 2015-11-09.

[24] D. Britz, "Understanding convolutional neural networks for nlp," http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/, 2015, accessed: 2015-11-07.

[25] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[26] "Colah's Blog understanding lstms," http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[29] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[30] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning.* ACM, 2009, pp. 1113–1120.

[31] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2vec: Character-based distributed representations for social media," *arXiv preprint arXiv:1605.03481*, 2016.

[32] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of NAACL-HLT*, 2016, pp. 1480–1489.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[35] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." 2015.

[36] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[37] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.