Content-based genre classification of large texts

Amr Shahin

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Computer Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

May 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Amr Shahin**

Entitled: **Content-based genre classification of large texts**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Examiner
Dr. Ching Yee Suen

______________________ Examiner
Dr. Charalambos Poullis

______________________ Supervisor
Dr. Adam Krzyzak

Approved by ______________________
Narayanan Lata, Chair
Department of Computer Science and Software Engineering

__________ 2019

____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Content-based genre classification of large texts

Amr Shahin

The advent of Natural Language Processing (NLP) and deep learning allows us to achieve tasks that sounded impossible about 10 years ago, one of those tasks is genre classification for large text bodies. Movies, books, novels, and various other texts more often than not, belong to one or more genres, the purpose of this research is to classify those texts into their genres while also calculating the weighed presence of this genre in the aforementioned texts. Movies in particular are classified into genres mostly for marketing purposes, and with no indication on which genre is the most autocratic.

In this thesis, we explore the possibility of using deep neural networks and NLP to classify movies using the contents of the movie script. We follow the philosophy that scenes makes movies and generate the final result based on the classification of each individual scene. the results were obtained by training Convolutional Neural Networks (ConvNet or CNN) and Hierarchical Attention Networks (HAN) and compare their performance to the de-facto architectures for NLP, namely Recurrent Neural Networks (RNN) and Attention Models.

The results we got on the validation data-set are comparable to those obtained by similar research done mostly on sentiment analysis or rating predictions, the accuracy is about 85% which is an acceptable measure in the literature. We dedicated a part of our conclusion discussing how our models would perform on a larger dataset and what steps could be taken to increase the accuracy.

Acknowledgments

I would first like to thank my thesis advisor Professor Adam Krzyzak of the ENCS department at Concordia. Professor Krzyzak's guidance and support were consistent and unconditional, his vast and impressive knowledge in machine learning and deep learning steered me in the right direction whenever needed, as well as providing me with very valuable suggestions.

Furthermore, I would like to acknowledge my employer thought my studies, Verdant Environmental Technologies, for their support and understanding. I would also like to thank the experts who were involved in the validation survey for this research project: Dr. Ali Frejat and Mrs Najlaa Al Qawasmi, Without their passionate participation and input, the validation survey could not have been successfully conducted. Also, I would also like to thank Compute Canada for providing me with the resources needed to run my research.

Finally, I must express my very profound gratitude to my mother and sisters: Jihad, Dania, Rand And Tala, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Movie watchers, as well as movie recommendation engines, do not have any empirical means of determining a movie's genre(s). Moreover, the genre provided by sources like imdb.com do not indicate the dominating genre and generally select from a predetermined limited set of genres. In this thesis, we show that better results can be obtained via deep learning, our work shows that our work both generates an accurate dominating genre for the movies along with a probability for each genre. This research covers the entire process: Collecting and annotating data, evaluating the performance of different algorithms, and lastly future work and the possibility of integrating the work into recommendation engines.

1.2 Related Work

In this section, we will be covering generic text classification research along with genre classification as they both are closely related to our work.

Brezeale and Cook [5]. Used a combination of closed captions[1] and visual features to detect the genre of a video using a support vector machine (SVM). The authors used 81 movies from the MovieLens project [2] and were able to achieve 89.71% accuracy when using closed captions as the feature vector inputted to the SVM, it is not clear, however how the authors calculated the accuracy when the movie belongs to multiple genres. Moreover, the usage of classical machine learning algorithms like SVM and bag of words does not scale well when working with a larger dataset. The authors also pointed out that the closed captions "typically won't include references to non-dialog sounds", we found that using the original script of the movie circumvents this problem as not only it includes the actual speech, but the "general feeling" of the scene.

Aside from the aforementioned paper to our best knowledge, there was no other research that directly works with text genres using the movie script, we will be presenting research done on text classification in general in the rest of this chapter.

Chen and Soo. [6] implemented a Convolutional Neural Network (CNN) and Highway Networks to detect humor in both English and Chinese. The author's dataset consisted of 16000 one-liners constructed by Mihalcea and Strapparava. [7], Pun of the Day constructed from https://www.punoftheday.com/, Short Jokes Dataset from Kaggle project, the Chinese data was constructed from PTT Jokes, the largest terminal-based bulletin board system (BBS) in Taiwan. The authors used a CNN network with varying filter size and a highway layers that allows the data to travel through the network skipping some layers for faster training. The model was able to achieve an accuracy of 0.897, 0.894, 0.906 and 0.957 on 16000 One-Liners, Pun of the Day, Short Jokes and PTT Jokes respectively. The table below, taken directly from the paper, shows samples of true and false positives and negatives:

[1] https://en.wikipedia.org/wiki/Closed_captioning
[2] https://grouplens.org/datasets/movielens/

Sentence	
TP	when he gave his wife a necklace he got a chain reaction
TN	the barking of a dog does not disturb the man on a camel
FP	rats know the way of rats
FN	it's a fact taller people sleep longer in bed

Li and Qian [4]. compared using Long-Short Term Memory (LSTM) Recurrent Neural Network (RNN) for a three-way sentiment analysis (positive, negative and neutral). The paper suggests that the structure of LSTMs allows discovering both long and short patterns in the data as opposed to RNNs which suffer from exploding and vanishing gradients. The authors used four datasets: comments from the website http://jd.com (two types of comments, in Chinese), travel comments from http://www.ctrip.com/ (Chinese) and English movie reviews. The travel comments movie reviews are classified into positive and negative manually, while the comments from jd.com are classified into positive, negative and neutral. The experiment was done by training three individual LSTM networks to detect positive, negative and neutral comments respectively. Table 1.1 shows a summary of the results of the authors' works.

Table 1.1: Summary of the results in [4]

Data Source	Sentiment	Accuracy
jd.com	Positive	95.62%
jd.com	Negative	88.7%
jd.com	Neutral	91%
ctrip	Positive	87.6%
ctrip	Negative	88.95%
English movie reviews	Positive	84.54%
English movie reviews	Negative	89.99%

Unlike previous work Zhang et al. [8]. represented the words as a raw signal to mitigate the language-dependency and overcome spelling mistakes, arguing that representing the whole word as a vector in deep learning methods suffers major shortcomings, as opposed to the classical method of word-embedding. The authors show that Convolutional Neural Networks (ConvNets) are able to extract features automatically. Their model accepts a one-hot encoded vector of letters that acts as an input to two ConvNets that are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers, the first ConvNet takes a feature vector of size 256 and the second ConvNet takes a vector of size 1024 as inputs. Later in 2018 [9]. Xiao and Cho criticized the usage of a small receptive field in [8] arguing that it leads to a deeper ConvNet and thus, a larger number of parameters.

Wang et al. [10] used an attention-based LSTM NN structure with GloVE word embedding to obtain sentiment analysis on SemEval 2014 Task 4 dataset [3], their model achieved an accuracy of 84.0 on three-way (positive, negative and neutral) and 89.9 on positive/negative classification. Although different from multi-class genre classification, the results in this paper show great potential for LSTM NN in general and attention-based LSTMs in particular, more details on this will be provided in later chapters as this is the structure we use in our experiments in similar.

Yogatama et al. [11]. argue that using discriminative models yields higher error rates when the data distribution changes and found that using generative models leads to lower error rates, with slower training nevertheless. The authors used the dataset available from [8] (`http://goo.gl/JyCnZq`) which consist of news classification, sentiment analysis, wiki article classification, and Q&A categorization. The authors used Naive Bayes classifier, Kneser–Ney Bayes classifier and Naive Bayes neural network as their baseline generative models. In their experiment, the authors used: Discriminative LSTM which learns how to classify a document based on the training data, and a Generative LSTM which they show can learn to classify new classes independently with the drawback being the need to train a new model from scratch for each new class. The authors conclude that generative models perform better when it comes to small data albeit is more resource intensive.

Yang et al. [3] applied a two-level hierarchical attention network for sentiment analysis on the following datasets: Yelp reviews, IMDB reviews, Yahoo answers, and

[3]http://alt.qcri.org/semeval2014/

Amazon reviews. The authors' model embeds the word into vectors and uses a word encoder followed by an attention vector that detects the words that contribute the most to the classification, the model also applies a similar technique to the sentences that represent each class where each sentence is encoded using a Gated Recurrent Unit (GRU) and the sentences that contribute the most to the correct class are rewarded with high attention values.

1.3 Objectives, Contributions and Challenges

In this chapter we list both our objectives and our contributions to the literature, one of which is the dataset we used in our research, we also list the challenges we faced while building the dataset.

1.3.1 Objectives and Contributions

- The main objective of this work is building a large text classifier that, not only can detect the genre of the text, but also is capable of calculating the percentage of the domination of this genre in the text. The methodology for achieving this is described in details in Chapter 3

- The outcome of our research will help NLP researchers working with a genre-specific classification to have a solid ground to start from, we clearly detail the models we experimented and their results, provide intuition behind the results of each model and the thinking process we went through while experimenting with various hyper parameters and we show the final results of each experiment.

- Finally, the results of our working have the potential of being utilized in recommendation engines, the existing content-based engine either utilize a single genre of the movie or include all genres without any notion of their dominance in the movie.

1.3.2 Dataset

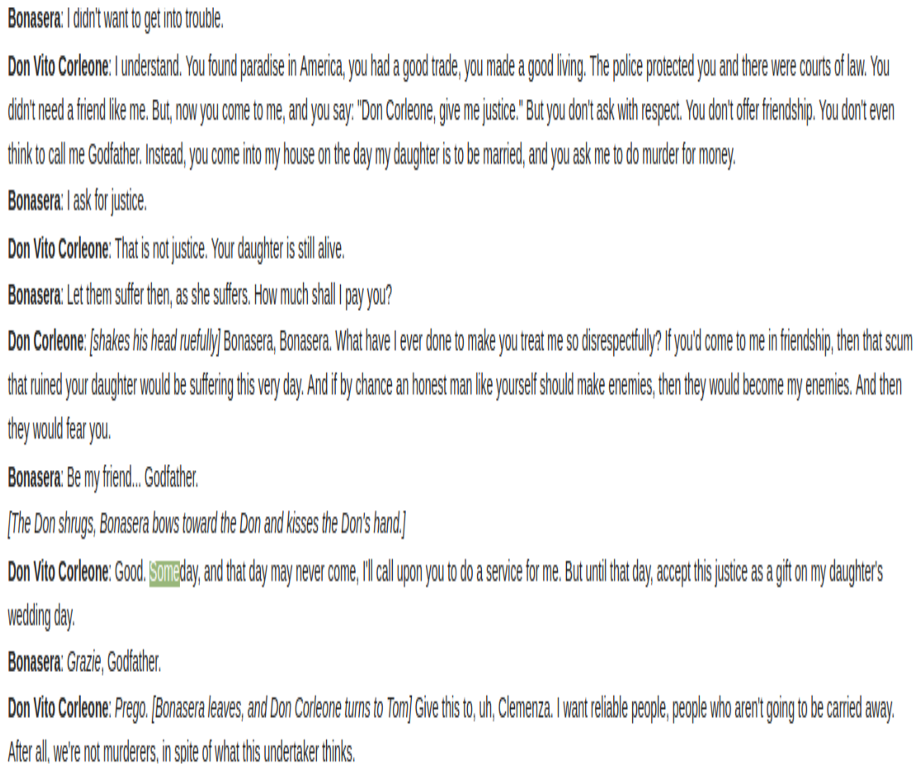
One of the biggest challenges we faced while writing this thesis is finding a suitable dataset that contains proper training data. The form of data we were looking for is a sentence mapped to a single genre (i.e: [But you step aside for the good of the party; people won't forget. The President and I won't let them] belongs to the genre 'politics', [He's in love with you. I've only ever seen him look at one other girl the way he looks at you] is 'romance', etc ...). Unfortunately, such dataset does not exist.

1.3.2.1 Available Datasets and their issues

We considered using The Internet Movie Script Database [4] which contains the full script of most movies (see figure 1.2) and parsing the contents, however, this leaves us with a large text corpus that belongs to multiple genres, which presents two issues: 1. the resources required to run an RNN to handle a corpus of this size are enormous (see Chapter 2 for details), and 2. Training a NN on a text that belongs to multiple genres will lead to the network learning the joint probability of the genres which is not desired in our experiment.

Another option was using a news dataset which contains texts mapped to a certain news category such as https://www.kaggle.com/crawford/20-newsgroups or https://www.kaggle.com/therohk/india-headlines-news-dataset, we found that there is no clear one-to-one mapping between a news class and a movie genre aside from politics and sports which will cause our model not to scale well, should we decide to add more genres to our work.

[4] https://www.imsdb.com/


Bonasera: I didn't want to get into trouble.

Don Vito Corleone: I understand. You found paradise in America, you had a good trade, you made a good living. The police protected you and there were courts of law. You didn't need a friend like me. But, now you come to me, and you say: "Don Corleone, give me justice." But you don't ask with respect. You don't offer friendship. You don't even think to call me Godfather. Instead, you come into my house on the day my daughter is to be married, and you ask me to do murder for money.

Bonasera: I ask for justice.

Don Vito Corleone: That is not justice. Your daughter is still alive.

Bonasera: Let them suffer then, as she suffers. How much shall I pay you?

Don Corleone: *[shakes his head ruefully]* Bonasera, Bonasera. What have I ever done to make you treat me so disrespectfully? If you'd come to me in friendship, then that scum that ruined your daughter would be suffering this very day. And if by chance an honest man like yourself should make enemies, then they would become my enemies. And then they would fear you.

Bonasera: Be my friend... Godfather.

[The Don shrugs, Bonasera bows toward the Don and kisses the Don's hand.]

Don Vito Corleone: Good. Someday, and that day may never come, I'll call upon you to do a service for me. But until that day, accept this justice as a gift on my daughter's wedding day.

Bonasera: *Grazie*, Godfather.

Don Vito Corleone: *Prego. [Bonasera leaves, and Don Corleone turns to Tom]* Give this to, uh, Clemenza. I want reliable people, people who aren't going to be carried away. After all, we're not murderers, in spite of what this undertaker thinks.

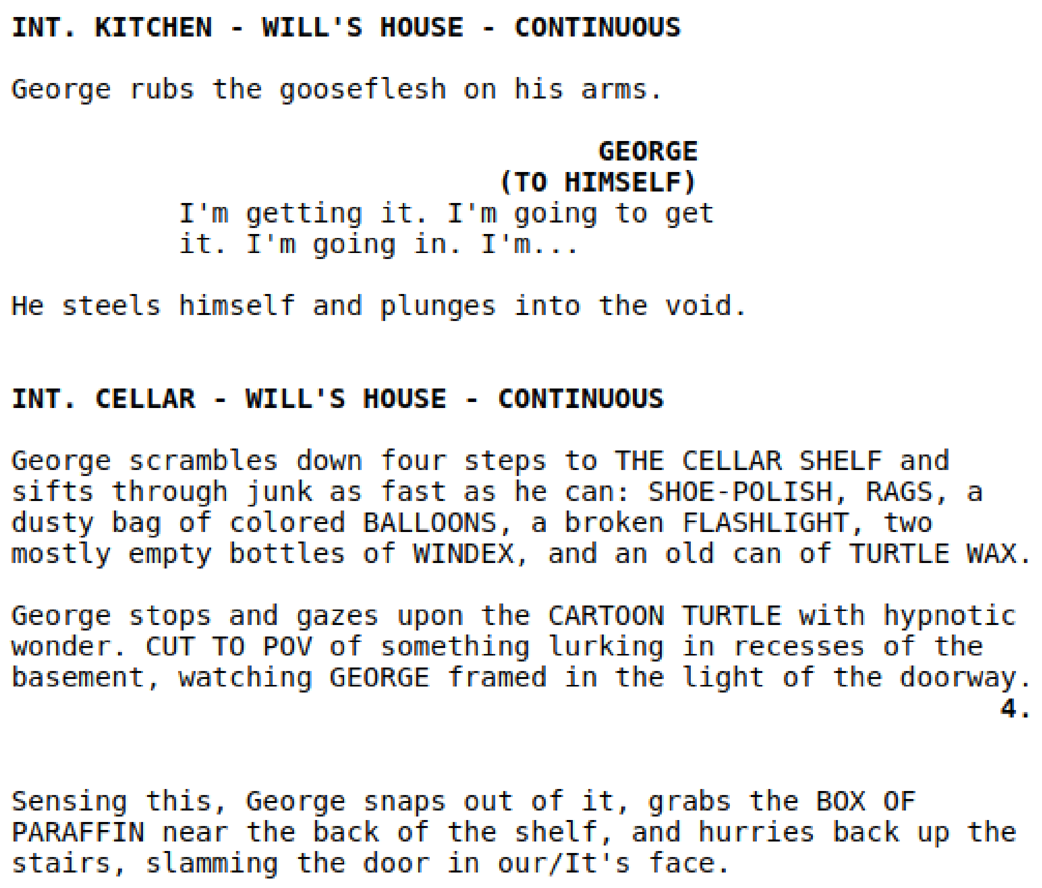
Figure 1.1: Quotes from the movie "Godfather" taken from wikiquote.com

1.3.2.2 Building a Custom Dataset

Due to the above-mentioned reasons, we decided to build our own dataset and we chose five genres: action, comedy, drama, politics, and romance to collect data for. Our process starts with scraping Google search using the queries: "Top <genre> movies" and "Top <genre> series" [5]. Out of the search result, we manually selected a collection of movies and series that we felt represent the selected genre best (see Appendix A for the list of movies and series). Using this set of movies and series, we built a tool that scrapes the API of https://en.wikiquote.org/ collecting quotes belonging to this set. Figure 1.1 shows a sample taken from the movie Godfather.

This method provided us with the data and format we needed, however, the number of samples we collected varied greatly between genres (Having this variety in the samples can hurt the calculation of the accuracy, see Chapter 2 for details), for

[5]Thus, looking up the genre "Romance" will result in queries: "Top romance movies" and "Top romance series"


```
INT. KITCHEN - WILL'S HOUSE - CONTINUOUS

George rubs the gooseflesh on his arms.

                                        GEORGE
                                   (TO HIMSELF)
          I'm getting it. I'm going to get
          it. I'm going in. I'm...

He steels himself and plunges into the void.


INT. CELLAR - WILL'S HOUSE - CONTINUOUS

George scrambles down four steps to THE CELLAR SHELF and
sifts through junk as fast as he can: SHOE-POLISH, RAGS, a
dusty bag of colored BALLOONS, a broken FLASHLIGHT, two
mostly empty bottles of WINDEX, and an old can of TURTLE WAX.

George stops and gazes upon the CARTOON TURTLE with hypnotic
wonder. CUT TO POV of something lurking in recesses of the
basement, watching GEORGE framed in the light of the doorway.
                                                           4.


Sensing this, George snaps out of it, grabs the BOX OF
PARAFFIN near the back of the shelf, and hurries back up the
stairs, slamming the door in our/It's face.
```

Figure 1.2: A sample movie script

this reason, we decided to append data we scraped from https://www.goodreads.com/quotes in order to have matching numbers in our five genres. We managed to collect 3000 samples for each genre which we have used both for training and testing.

1.4 Algorithms and Models

In this thesis, we start off by presenting our two most successful models that were able to achieve test accuracy of 75% achieved by a Convolutional Neural Network (CNN) model and 89% accuracy achieved by a Hierarchical Attention Network (HAN) classifying text corpuses into the aforementioned genres, we analyze the models and how they relate to our dataset, and mention the flow of experiments that we did that lead us to the final model structure. We also give some examples of models that were expected to perform well due to their wide usage for NLP and related tasks but bucked our expectations. Moreover, we used two most well known word-embedding algorithms, GloVe: Global Vectors for Word Representation, and word2vec negative sampling and found little differences in performance when using either.

Furthermore, our work showed some interesting trends when it comes to classifying genres, some genres are very likely to have con flits with other genres due to similar sentence structure. We document these findings along with our recommendations to circumvent or avoid such cases.

We also provide our technical findings as well as the limitations that researchers studying the same topic could run into, we suggest ways to work around these limitations when possible and we document the cases where we found that resources could be a show-stopper to any future work that can be done.

1.4.1 Usage of the trained model

As mentioned in the previous section, the model will be trained to classify shorter texts samples taken from movies as opposed to full movie/book scripts, and thus; the model cannot be used as-is to classify a large corpus. To circumvent this limitation, we divide any corpus into logical smaller sections [6] which can be classified individually, the overall genre(s) assigned to each logical section will determine the overall genre(s) of the corpus, this model proved to work as we were able to correctly classify the genre of three movies correctly as we show in Chapter 3

[6] For instance: a movie scene, a book paragraph, etc ...

1.5 Overview of the Thesis

The rest of this document is laid out as follows: In chapter 2 we give an overview of the theory and concepts that are essential to understanding the work done in the rest of the project. In chapter 3, we introduce the model and methods used in this project. Then we present our experiments, results, and evaluation in chapter 4. Finally, we conclude the report in chapter 5 and also give some ideas for future work and how our work can be used to predict the genres of a large texts corpus.

Chapter 2

Background

In this chapter, we provide some of the background knowledge necessary that will help the reader understand our work, we describe the Recurrent Neural Networks, Convolutional Neural Networks, attention models and various other related algorithms that we felt help the reader understand our work.

2.1 Convolutional Neural Networks

Convolutional Neural Networks or ConvNets were first introduced by Lecun et al. [12] as an alternative to multilayer perceptron (MCP) for computer vision.

A ConvNet consists of multiple convolutions and pooling layers. At the end follows normally a fully connected layer [1]. A pooling layer (max or average) is applied after one or multiple convolution layers. The convolution layers have the task to extract useful features from the input, which results in multiple feature maps. The pooling layer reduces the spatial size of these feature maps.

[1]The purpose of the final fully connected layer is typically used as an output layer but it is possible to stack fully connected layers as well

2.1.1 Convolution Layer

A convolution layer is similar to a fully connected layer in the sense that it consists of a weight, but with a different arrangement and different connections of the weight. The main other differences to the neural networks are:

- The weights are multi-dimensional as opposed to single-dimension in the fully connected layer.
- Weight sharing
- Local connectivity

The multiple dimensional arrangements come from the input data. For instance, a sentence in the case of NLP gets mapped into X dimensional layer using word embedding, which in turns becomes the input of the ConvNet. The output of a convolution layer is again a multidimensional matrix representing the feature maps of the input X the number of filters in this layer. Every filter produces a feature map. Finally, Local connectivity means, that not all units of the input are connected with the output unit. The size of the local connectivity is described by the kernel size.

Weight sharing means, that the same weights are used for multiple output units. Through this, the ConvNet gets the property, that the features are invariant against translation. This means, that a feature can be found on the complete input. Figure 2.1 provides an example of a convolutional neural network.

2.1.2 Convolution Layer Equations and Memory Requirements

The computations in each convolutional layer require performing a convolution of each filter across the entire input which in turn is passed to the activation function.

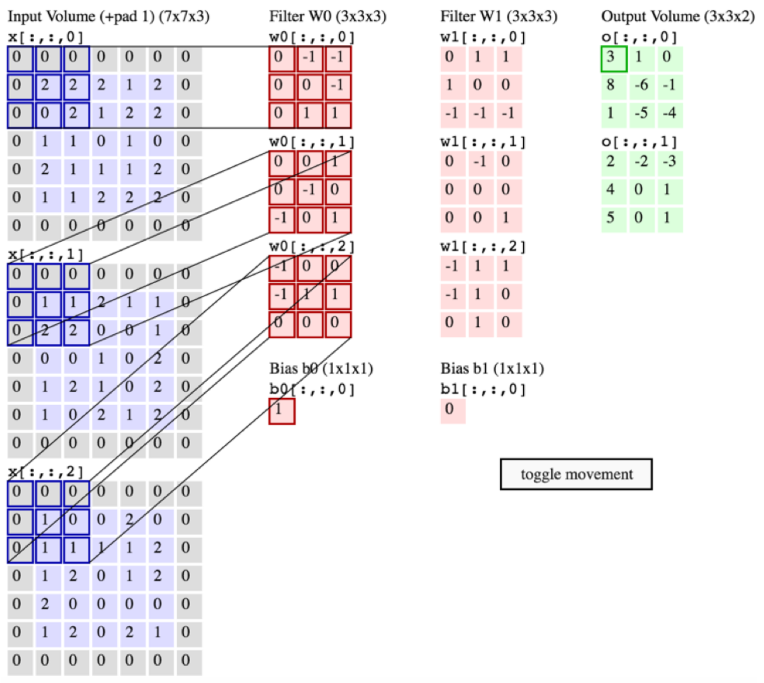


Figure 2.1: A convolutional neural network (source: https://medium.com/@phidaouss/convolutional-neural-networks-cnn-or-convnets-d7c688b0a207)

The filter is multiplied element-wise with the upper-leftmost values of the input. The result is added together to produce the output. In the next step, the filter is shifted by stride S across the input, to produce the next output activation. This process is repeated for the entire output to compute the output activation. The aforementioned process can be described by the following equations:

$$\begin{aligned} O[p,q,k] &= \sum_{c=1}^{C}\sum_{s=1}^{S}\sum_{r=1}^{R} = I[x,y,c] \times F_k[r,s,c] \\ &\forall p = 1..P, q = 1..Q, k = 1..K \\ x = p \times m &+ r - 1, y = q \times m + s - 1 \end{aligned} \tag{1}$$

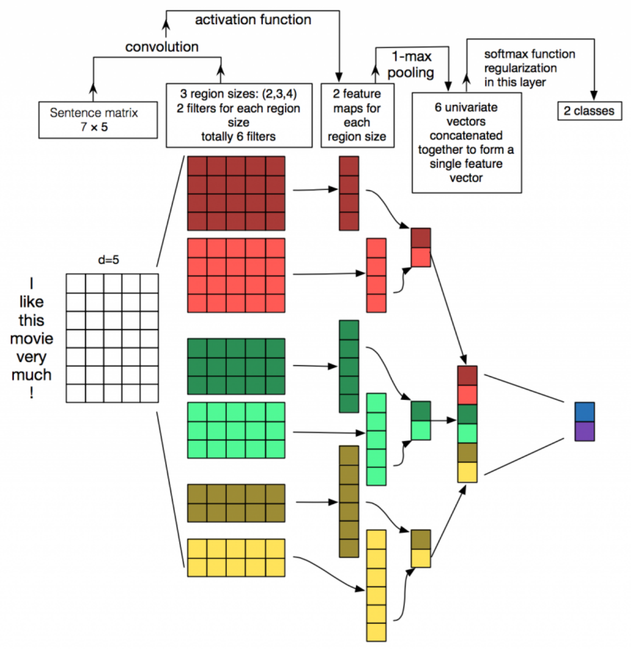

activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Figure 2.2: one-dimensional convolutional layer (image source: Figure 1 Zhang, Y. & Wallace. [1])

2.1.3 1D Convolutions

To understand the 1D convolutional layer, we must take a closer look at the input compared the traditional 2d convolutions applied to images. In the case of images, the convolutional layer slides over patches of the image in order to extract features, in this case, the input is a matrix of size $A \times B$ representing the image. In the case of colored images, the input expands to contain 3 channels representing the RGB of the image. In the case of text, the input is the word embedding representation of a sentence, assuming we use an embedding of size 100, the input is a single vector, with each entry being of size 100. The convolution, in this case, slides over X words at a time as opposed to image patches in the case of 2d convolutions. Figure 2.2 shows a 1D convolutional layer with a textual input.

2.1.4 Pooling Layer

The pooling layers are typically applied after convolutional layers. they reduce the size of the feature maps and thus the number of parameters leading to faster learning and fewer memory requirements. In other words, the pooling layer down samples the feature maps. With one pooling with the stride size of 2×2, the spatial dimension of the feature maps is reduced by 75%. It is worth noting that the pooling layer has no activation function or weights to learn which makes it very fast. Furthermore, the pooling layer helps the network to be invariant to small changes of the input as they mostly have no effect on the values of the outputs of the pooling layer.

2.2 Recurrent Neural Networks

Regular (feed forward) networks do not have the ability to work with a sequence of inputs (a simple example being stock prices for the past X days). Time-Series data is critical to many applications such as NLP where the final output might be affected not only by the input but also by the order in which the input appears. For example, one might use each word of a sentence one by one as input. Since feedforward networks have no concept of state, it is not possible to detect dependencies between cohesive words being input to the network at different times (i.e: "very good" is a positive review, while "not very good" is a negative review). Recurrent neural networks solve this issue by introducing a recurrent connection from a neuron's previous state to its next one. See Figures 2.3, 2.4 for a visual representation of an RNN.

The basic equations of the RNN are:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{2}$$

$$h^{(t)} = \tanh(a^{(t)}) \tag{3}$$

$$o^{(t)} = c + Vh^{(t)} \tag{4}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \tag{5}$$

Where: $x^{(t)}$ is the input at time step t, $s^{(t)}$ is the hidden state at time step t and $o^{(t)}$ is the output at step t [2].

We can see from the equations above that a vanilla RNN has three sets of weights: W, U and V. However, unlike feedforward networks, the RNN shares the parameters throughout the steps, when the loss is calculated, it gets summed over all the sequences using the standard cross-entropy loss function:

$$L(\{x_1, ..., x_t\}, \{y_1, ..., y_t\}) = \sum L^{(t)} \tag{6}$$

$$\begin{aligned} L^{(t)} = E_t(y^{(t)}, \hat{y}^{(t)}) &= -y^{(t)} \log \hat{y}^{(t)} \\ E(y, \hat{y}) &= \sum_t E_t(y^{(t)}, \hat{y}^{(t)}) \\ &= -\sum_t y^{(t)} \log \hat{y}^{(t)} \end{aligned} \tag{7}$$

During the forward propagation step, the RNN unrolls its inputs into a network of size X (X being the size of the RNN cell), thus, if the input is a 5 word sentence, the RNN will be unrolled into a 5-layer network as shown in figure 2.5, the back-propagation is described in the next section.

2.3 Back Propagation Through Time (BPTT)

In this section, we will describe the backpropagation algorithm for RNN and how it differs from the standard backpropagation.

[2]In this particular thesis, the output is only needed from the last cell since we do not map each word to a genre, however, in cases like machine translation, each cell has its own output

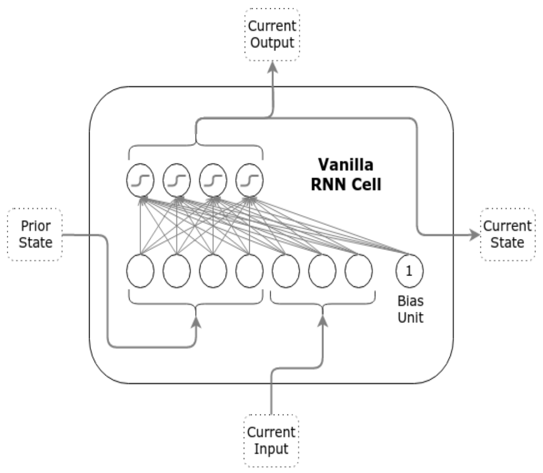


Figure 2.3: A Vanilla RNN layer (source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

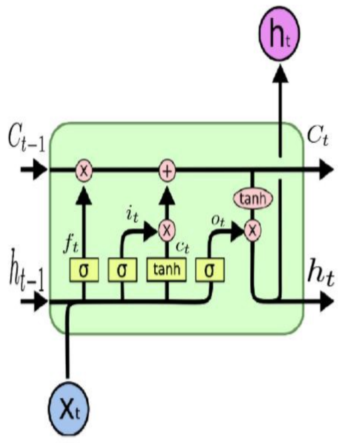


Figure 2.4: An RNN cell (source: https://hackernoon.com/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4)

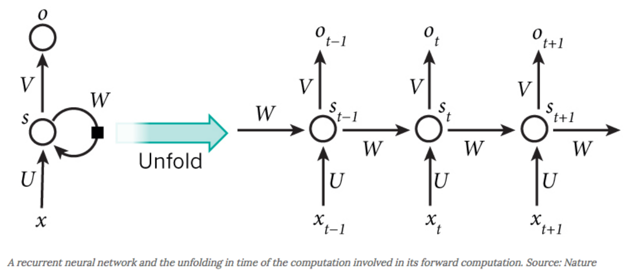

o
V
s
W
U
x
Unfold
W
o_{t-1}
o_t
o_{t+1}
s_{t-1}
s_t
s_{t+1}
x_{t-1}
x_t
x_{t+1}
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

Figure 2.5: RNN unroll (source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/)

Just like the standard backpropagation, we use the chain rule of differentiation, except in the case of RNN, we have three weight matrices to optimize rather than just one in the case of a feed forward network, thus, the equations for backpropagation become:

$$\begin{aligned}\frac{\partial E_t}{\partial V} &= \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial V} \\ &= \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial V} \\ &= (\hat{y}_t - y_t) \otimes s_t\end{aligned} \tag{8}$$

Since V is only dependant on the values from the current time step (see equation 1). In the case of U and W, the updates are more complicated:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial s_t}\frac{\partial s_t}{\partial W} \tag{9}$$

And

$$\frac{\partial E_t}{\partial U} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial s_t}\frac{\partial s_t}{\partial U} \tag{10}$$

Unrolling the above equation results in:

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial s_t}\frac{\partial s_t}{\partial s_k}\frac{\partial s_k}{\partial W} \tag{11}$$

$$\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial s_t}\frac{\partial s_t}{\partial s_k}\frac{\partial s_k}{U} \tag{12}$$

Where each time step contributes to the loss. In summary, the BPTT is very similar to the standard BP, but instead of training the weights of each layer individually, the parameters are shared across the cells and the loss is summed up.

2.4 Long Short-Term Memory Networks LSTM

LSTM were introduced by Hochreiter & Schmidhuber [13]. as a suggested solution to the vanishing and exploding gradient problem, a problem that arises when the RNN is working with a long input sequence. A closer look at Equation 10 shows that the derivative at a time step (t) gets multiplied by the derivatives at time steps (0) to (t-1). In the case where W<1.0, the weights will decrease asymptotically at each time step, causing the phenomena of vanishing gradients, and if W>1.0, the weight will increase asymptotically causing exploding gradients. The limitations of RNNs are discussed in details by Bengio, et al. [14]

The main advantage of the LSTM over the RNN is the memory cell which is shown in Figure 2.6. A memory cell has four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The weight of the self-recurrent connection is 1.0 and ensures that the state of a memory cell can remain without a change in different time step. The input gate can let the incoming signal change the state of the memory cell or block it. Also, the output gate can let the state of the memory cell change other neurons or prevent it. The forget gate can let the cell to remember or forget its previous state, as needed, the equations ruling the LSTM cell are:

$$f^{(t)} = \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) \tag{13}$$

$$i^{(t)} = \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) \tag{14}$$

$$o^{(t)} = \sigma(W_o x^{(t)} + U_o h^{(t-1)} + b_o) \tag{15}$$

$$c^{(t)} = f^{(t)} \cdot c^{(t-1)} + i^{(t)} \cdot \sigma(W_c x^{(t)} + U_c h^{(t-1)} + b_c) \tag{16}$$

$$h^{(t)} = o^{(t)} \cdot \sigma(c^{(t)}) \tag{17}$$

Where:

$x_t \in R^d$: input vector to the LSTM unit

$f_t \in R^h f_t \in R^h$: forget gate's activation vector

$i_t \in R^h\ i_t \in R^h$: input gate's activation vector

$o_t \in \mathbb{R}^h$ $o_t \in \mathbb{R}^h$: output gate's activation vector

$h_t \in \mathbb{R}^h h_t \in \mathbb{R}^h$: hidden state vector

$c_t \in \mathbb{R}^h c_t \in \mathbb{R}^h$: cell state vector

$W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in R^h$: weight matrices and bias vector parameters which need to be learned during training

An example where the forget gate of LSTM can be useful is when dealing with a sentence like "**John** brought his dog, **Jane** brought her cat" where the network is expected to predict the next word, in the case of the aforementioned sentence, the forget gate will allow the LSTM to forget the earlier subject "John", and remember the latest subject "Jane" where it can correctly predict that the next word is "her" rather than "his".

2.4.1 Variants Of Long Short Term Memory

Most of the LSTM variants play on dropping or combining some of the gates that a cell has, one interesting variant was introduced by Gers & Schmidhuber [15] where the cell has a peephole connection that allows the gates to look at the cell state, another very popular variant is the Gated Recurrent Unit (GRU), introduced by Cho, et al. [16]. It combines the forget and input gates into a single update gate. It also merges the cell state and hidden state and makes some other changes. The resulting model is simpler than standard LSTM models. Greff et al. [17] compared some of the most popular variants and found very little difference in terms of performance among them.

2.4.2 LSTM Memory Requirements

In this section, we present the amount of memory needed to construct LSTM networks as it will help the reader understand the practical limitations faced while dealing with

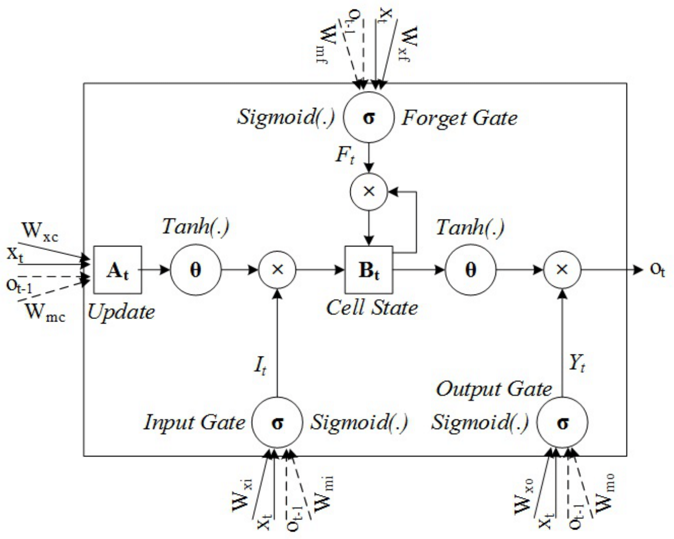


Figure 2.6: LSTM cell

large text sequences.

Looking back at equations 13 - 17 and assuming our input size is m and output size n, we conclude that:

- The weight vector U is of dimensions $n \times m$

- The weight vector W is of dimensions $n \times n$

- and the bias vector b of size n

Considering there are four sets of these parameters, one for each gate as well as an extra set to update the cell status, the total number of parameters sum to 4(nm+n2+n)

In our chosen deep learning framework, tensorflow, the single weight is a float 32 (8 bytes) floating point number, making the weight for a network total to 32(nm+n2+n).

The details of the memory needed for each of our experiments will be states in the corresponding sections.

2.4.3 Attention

Conventional LSTM architectures suffer from the constraint that all input sequences need to be of the same length which imposes difficulties when the sequence length becomes very long. Attention solves this problem by keeping intermediate outputs from each step and training the model to pay selective attention to the input sequences.

The idea of attention was first introduced by Larochelle & Hinton [18] for computer vision tasks. The authors implemented a system for combining glimpses that jointly train a recognition component with an attention component. In their experiment of facial expression recognition, the authors were able to achieve relatively high accuracy by using the attention model, it was also possible for them to show what part of the image the network was paying attention for when classifying an image as positive or negative. Figure 2.7 shows how attention focuses on certain parts of the image while automatically generating a caption.

Vaswani et al. [19] proposed that using attention without LSTM, RNN or CNN can produce better results compared to combining attention with other layers. Their model was composed of N = 6 identical layers, each layer consisting of multi-head self-attention mechanism and a simple fully connected layer in the encoder, and a similar architecture for the decoder with the addition of a third layer that performs multi-head attention on the output. The author's model performed very well in translation, with a BLEU score of 28.4 for English-to-German and 41.0 for English-to-French.

To understand attention better, consider the following example: "The food was amazing" as a positive review, and "The place was terrible" as a negative review. The attention vector will place high importance on the words "amazing" and "terrible" as

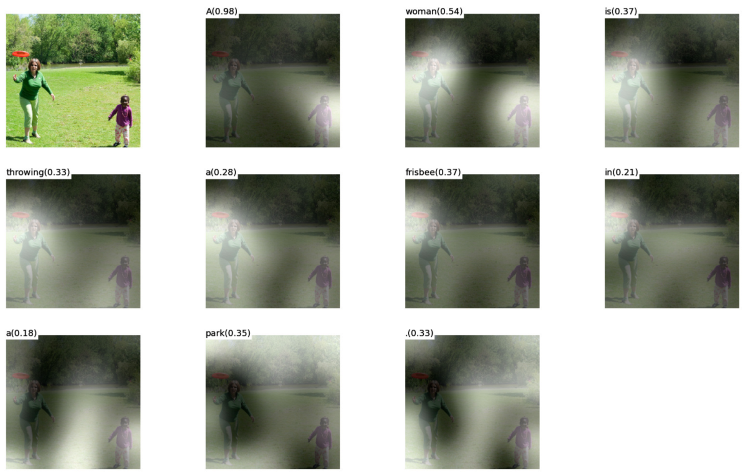


Figure 2.7: A woman is throwing a frisbee in a park." (Image source: Fig. 6(b) Xu et al. [2]
)

they determine the general sentiment of the sentence, as opposed to a standard RNN where it would place similar importance to each word in the input.

2.4.4 Attention Mechanisms

2.4.4.1 Self Attention (AKA Bahdanau or Intra attention)

Self-attention proposed by Bahdanau et al. [20] works by assigning an alignment score between the input as position i and the output based on how much the input affects the output. Self attention works by training a feed-forward networks with a single hidden layer along side the main network, thus, the loss function for attention neural network will be:

$$score(s_t, h_i) = v_a^T \tanh\left(Wa[s_t; h_i]\right) \tag{18}$$

Where W_a is the weight of the attention layer.

2.4.4.2 Luong Attention

Luong et al. [21] proposed the idea of global and local attention, the global attention is similar the aforementioned Bahdanau attention where the attention vector moves freely over the input, while the hard attention is a mix of soft attention and hard attention where only part of the inputs can have the attention at a given time step, the model is preferred over hard attention as it's differentiable.

2.5 Hierarchical Attention Networks (HANs)

HANs consist of stacked recurrent neural networks on word level followed by an attention model to extract important to the classification of the sentence and aggregate the representation of those informative words to form a sentence vector. Then the same procedure is applied to the derived sentence vectors which then generate a vector that carries the meaning of the given document and that vector can be passed further for text classification as shown in figure 2.8

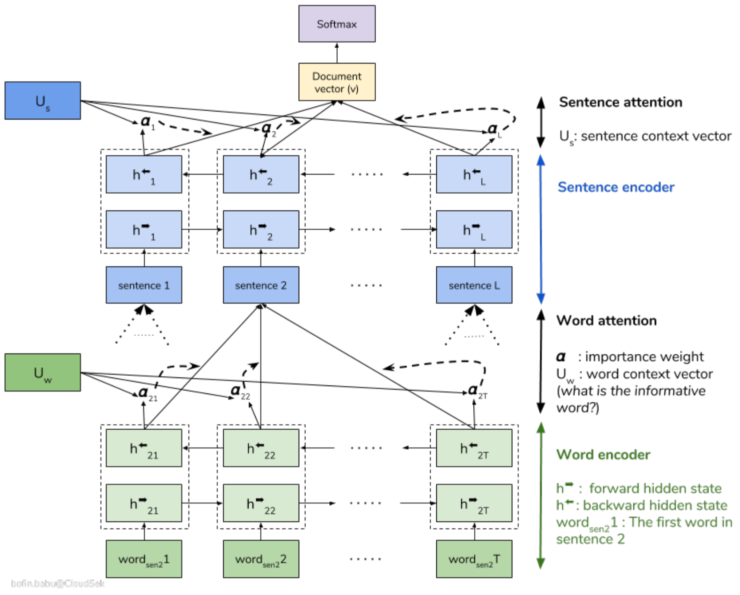


Figure 2.8: HAN structure (image source: Figure 1 Yang et al. [3])

2.6 Word Representation

Machine Learning and Deep Learning architectures are incapable of processing text directly as input. In the case where the input is a text, pre-processing is needed in order to convert the text into numbers. In this section, we present the various methods to do so.

2.6.1 One-Hot Encoding

A one-hot encoding, in general, is a representation of categorical variables as vectors. Each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1, for example, applying One-Hot to an input vector consisting of: ['drama', 'comedy', 'sports'] will result in: [[1 0 0], [0 1 0], [0 0 1]].

While One-Hot encoding does work in the sense that they convert words to numbers, however, they fail to capture the relations between various words, the word "Ottawa" could be represented using to the 1000th column, while the word "Amman" could be the 1st column, although both words represent capitals and should have smaller distance. For this reason, One-Hot encoding is not widely used in NLP.

2.6.2 Word Embedding

A Word Embedding format generally tries to map a word to a vector, one important property of the vector representation of a single word is that its distance from other similar words [3] is less than that of less similar words. The resulting vector has the property that cosine similarity 19 between words is higher for more similar words. This property is very important when training a neural network as it helps the network

[3] Similar here means the words appear in the same context

determine the nature of words it did not see during training.

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta \tag{19}$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{20}$$

2.6.2.1 Word2Vec

Mikolov et al. [22] used skip-gram model to generate embeddings and circumvented some of the training challenges using negative sampling [23]. The main idea behind it is that you train a model on the context of each word, so similar words will have similar numerical representations.

Word2vec model learns the weights by feeding a pair of input word and a target word to a neural network with one hidden layer of size [embedding dimension, vocabulary size] and an output layer of dimension [vocab size] consisting of softmax units. The hidden layer represents the probability that the word it represents will appear in the same context as the target word.

When the network is done training, the output layer is dropped and the hidden layer will be used as the word vector.

2.6.2.2 GloVe: Global Vectors for Word Representation

Pennington et al. [24] Presented the idea of learning embeddings by constructing a co-occurrence matrix (words X context) that counts how frequently a word appears in a context [4].

To better understand how GloVe works, let's follow the example taken from the paper, Let P(k|w) be the probability that the word k appears in the context of word

[4]Context here is user-defined, in the literature it is usually chosen to be whether or not a word appears within X number of words of the target word

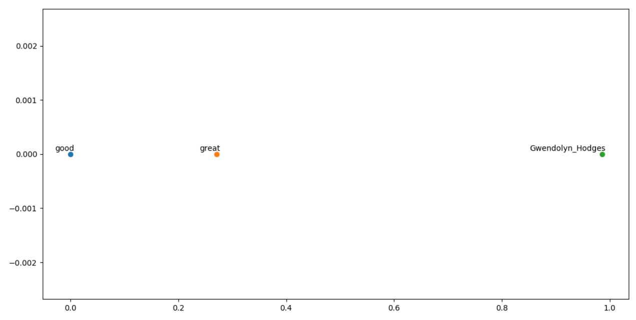


Figure 2.9: Distance between "good", "great" and "Gwendolyn_Hodges" (the least similar word to "good" as per Google's word embedding

w. A word like "ice" is likely to appear in the context of "solid", on the other hand, "solid" is much less likely to appear in the context of "gas" causing the ratio of P("solid" | "ice") / P("solid" | "steam") to be large. If we take a word such as gas that is related to steam but not to ice, the ratio of P(gas | ice) / P(gas | steam) will instead be small. For this reason, the authors use the probability ratio as the weight initializer rather than probabilities. Finally, The resulting co-occurrence will then be factorized and the resulting matrix will be used to represent the word embedding.

2.6.2.3 Importance of pre-trained embeddings

Word embeddings are particularly important when the model sees new or previously unseen words. For instance, if a model trained to detect a sentiment from restaurant reviews sees "The food was excellent" as a positive sample is asked to predict the sentiment of a statement like "great food", where "great" is an unseen word, the network will look up the word "great" from the pre-trained embeddings and determine that is has a similar meaning to "good" as they have similar weights as seen in figure 2.9

2.7 Optimizers

In this section, we cover the Adaptive Moment Estimation (Adam) and Root Mean Square Propagation (RmsPpro) optimizers, the two main optimizers we use to train our models, we also discuss briefly other optimizers such as Gradient Descent and Adagrad as they the base for most other optimizers.

2.7.1 Gradient Descent

Vanilla (batch) gradient descent, computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{21}$$

As we need to calculate the gradients for the entire dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory. Batch gradient descent also doesn't allow us to update our model online.

2.7.2 Stochastic Gradient Descent (SGD)

In contrast to Vanilla Gradient Descent, SGD performs the weight update for each batch of the training examples, changing equation 21 to:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{22}$$

Although it does perform more updates than GD, SGD is actually faster than GD since the computations are performed on a much smaller dataset, moreover, SGD can be used to train online.

2.7.3 Adaptive gradient (AdaGrad)

AdaGrad's basic idea is to adapt the learning rate to the parameters, performing a smaller update, AdaGrad's weight update equation is as follows:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{23}$$

Where $G_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix where each diagonal element i, i is the sum of the squares of the gradients w.r.t. θ_i

2.7.4 Root Mean Square Propagation (RMSprop)

RMSprop was suggested as a random idea by Geoff Hinton in a Coursera Class. The suggestion of RMSprop was a solution to AdaGrad's summing up squared gradients in its denominator leading to the learning rate to becoming exponentially small. RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients, setting the update equation to be:

$$\begin{aligned} E[g^2]_t &= 0.9E[g^2]_{t-1} + 0.1g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned} \tag{24}$$

2.7.5 Adaptive Moment Estimation (Adam)

Adam algorithm was first introduced by Kingma, D. P., & Ba, J. L. (2015) [25] as a method that computes adaptive learning rates for each parameter. Adam's weight update equations are as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{25}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{26}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{27}$$

Where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively.

The authors propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. They show empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms.

2.8 Performance Metrics And Balance Of The Training Data

2.8.1 Loss

The loss function is an important part of a neural network which is used to measure the inconsistency between predicted values and true values. Moreover, the loss function can be used to apply penalties on the weights in order to mitigate overfitting, a technique knows as regularization. The formula of the loss function with regularization is:

$= argmin_\theta \frac{1}{n} \sum_{i=0}^{n} L(y(i), f(x(i), \theta)) + \lambda.\phi(\theta)$

2.8.2 Categorical Cross-Entropy

Categorical cross-entropy is the de-facto loss function for multi-class classification, it measures the variance between the predicted class and the true class. The formula for categorical cross-entropy is:

$L = -\sum_{i}^{\#classes} y_i log(\hat{y}_i)$

Where y is the correct class and $\hat{y}$ is the predicted class.

2.8.3 Performance Metrics

2.8.3.1 Accuracy

Accuracy, simply put, is the most is the ratio of correctly predicted observation to the total observations, in the case of multi-class, the accuracy formula will be:

$$\frac{TruePositive + TrueNegative}{TotalSamples} \tag{28}$$

We can see from the equation above that the balance in the dataset is very important for the accuracy to have a meaningful value, for example, suppose we have a classification problem of three classes, A, B and C and a dataset of six samples of class A, two of class B and two of class C, furthermore, suppose our algorithm always predicts class A as the output. In that case, the accuracy would be:

$\frac{6+0+0+0+0+0}{10} = 60\%$

While an algorithm that produced correct predictions 50% of the time would have an accuracy of:

$\frac{3+0+1+0+1+0}{10} = 50\%$

However, if the input data were more balanced with 4 samples per class, the first algorithm's accuracy would be:

$\frac{2+0+0+0+0+0}{12} \approx 17\%$

and the second algorithm's accuracy would be:

$\frac{2+0+2+0+2+0}{12} = 50\%$

2.8.3.2 Precision

Precision - Precision is the ratio of correctly predicted positive observations of the total predicted positive observations. Having the formula:

$$\frac{TruePositive}{TruePositive + TrueNegative} \tag{29}$$

Low precision is desired when the algorithm is sensitive to false positives.

2.8.3.3 Recall

Recall (Sensitivity) is the ratio of correctly predicted positive observations of all the observations in the actual class. Having the formula:

$$\frac{TruePositive}{TruePositive + FalseNegative} \tag{30}$$

2.8.3.4 F1 Score

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Having the formula:

$$\frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{31}$$

Chapter 3

Methods And Results

3.1 Overview

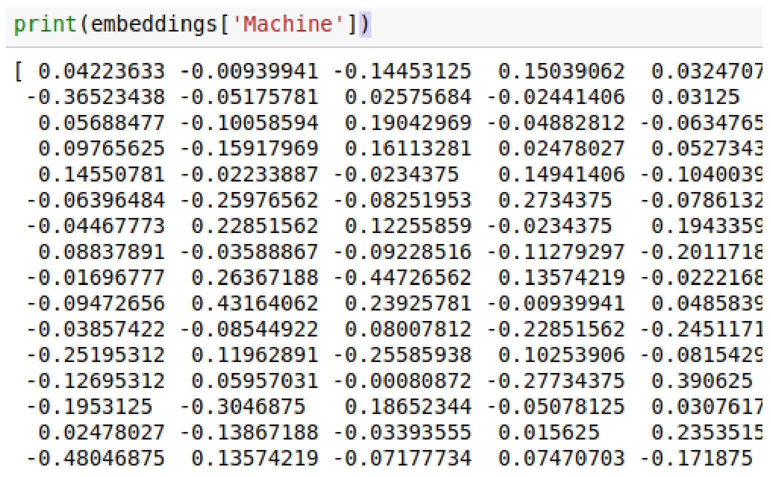
The aim of this research is to build a deep learning model that is able to detect the genre of a certain dialog. The final model should be able to classify a large text corpus into one or more genres with probabilities corresponding to each genre.

In this section, we will showcase the models we tried and provide a summary of the results we accumulated.

3.2 Data Preparation

This section describes building the input layer which converts words into a form of data that the network understands. This layer is the input for all the models described below unless otherwise specified.

In order to convert words into a form the network can understand, we first tokenize the sentences into individual words, which in turn get converted into integers each



```
print(embeddings['Machine'])
```

```
[ 0.04223633 -0.00939941 -0.14453125  0.15039062  0.0324707
 -0.36523438 -0.05175781  0.02575684 -0.02441406  0.03125
  0.05688477 -0.10058594  0.19042969 -0.04882812 -0.0634765
  0.09765625 -0.15917969  0.16113281  0.02478027  0.0527343
  0.14550781 -0.02233887 -0.0234375   0.14941406 -0.1040039
 -0.06396484 -0.25976562 -0.08251953  0.2734375  -0.0786132
 -0.04467773  0.22851562  0.12255859 -0.0234375   0.1943359
  0.08837891 -0.03588867 -0.09228516 -0.11279297 -0.2011718
 -0.01696777  0.26367188 -0.44726562  0.13574219 -0.0222168
 -0.09472656  0.43164062  0.23925781 -0.00939941  0.0485839
 -0.03857422 -0.08544922  0.08007812 -0.22851562 -0.2451171
 -0.25195312  0.11962891 -0.25585938  0.10253906 -0.0815429
 -0.12695312  0.05957031 -0.00080872 -0.27734375  0.390625
 -0.1953125  -0.3046875   0.18652344 -0.05078125  0.0307617
  0.02478027 -0.13867188 -0.03393555  0.015625    0.2353515
 -0.48046875  0.13574219 -0.07177734  0.07470703 -0.171875
```

Figure 3.1: The word embedding vector to the word "Machine"

representing a unique index corresponding to each word present in the corpus. The resulting tokenized sequences were padded to be all of the same lengths, the sequence length is a parameter that will be specified for each model in the corresponding section.

The embedding layer is the other part of the input, which consists of the word embeddings [2.6.2] of the form <unique word index>:<word embedding vector> as shown in 3.1. This is used as a look-up table for the neural network in order to map integers to vectors.

3.3 Models and Models Performance

3.3.1 Convolutions Neural Network (CNN) - 1D

3.3.1.1 Model Structure and hyper parameters

We started out experiments with 1-dimensional ConvNet which is gaining attention in the literature as a competitor to the LSTM networks which are traditionally used in NLP. For this model, we chose the learning algorithm to be Adam with a learning rate of 10^{-4} and a decay of 10^{-6}, although ConvNets traditionally use SGD algorithm, Dozat [26] suggests that Adam optimizer is more effective.

The model consists on an embedding input layer described in the Data Preperation section, the sentences are fed into a one dimension convolutional layer of size 75 followed by a max pooling layer with a pool size of 5. After the first convolution max pool combination, the model consists of 4 stacked convolutional layers with no pooling and filters sizes 60,40 and 25 respectively followed by a max pooling layer after the last convolutional layer, all the convolutional layers in this model use relu activation and stride of size 1. Finally, we added one fully connected layer of 75 with relu activation, and a fully connected layer of size 5 and a softmax activation to be the output layer. The full structure of the model can be seen in Figure: 3.2

3.3.1.2 Model Architecture

A single convolution channel works by multiplying its filters by a single channel of the output, summing the result of these multiplications constructing the first output, the processed is repeated throughout the input length by shifting the layers by the stride size up until the end of the input. The learning process of the convolutional layer works by finding the weights of the filters that minimize the loss.

More concretely, this model runs through the word embeddings generated from GloVe which converts each word into a 100-dimensional vector [1]. The first layer scans the input 75 words at a time and calculates the output in the manner mentioned above, the max-pooling layers extract the outputs with the highest activations. Next, the result to a batch normalization layer and finally, we apply a dropout with keep probability of 0.5 is applied to the output. This output is passed as an input to the consecutive convolutional layers. A flattened representation of the convolution results is passed to a fully connected layer with a relu activation and lastly to an output layer with a softmax activation. Table 3.1 shows a summary of the model's

[1]A good analogy for the word vector is a single pixel consisting of a 100 channels

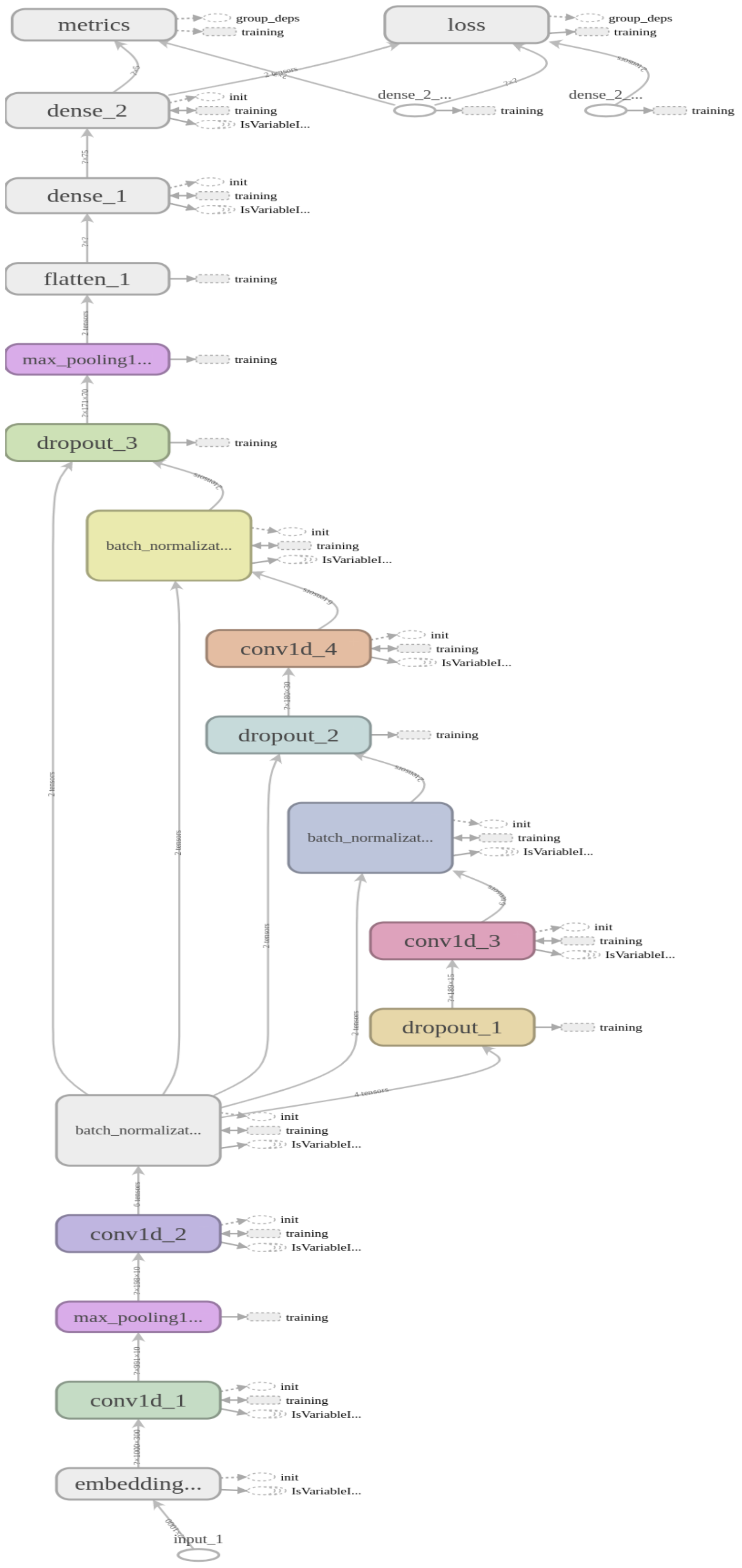


Figure 3.2: CNN model architecture

hyper parameters.

Table 3.1: CNN hyper parameters

Parameter	**Value**
Number of Convolutional Layers	4
Filter Sizes	75, 60, 40, 25
Number Of Channels	10, 10, 10, 10
Embedding algorithm	GloVe
Embedding dimension	100
Total parameters:	3,277,550
Trainable parameters:	18,850
Maximum Sequence Length	1000
Optimizer	Adam Optimizer With Learning Rate=10^{-4} And a Decay Of 10^{-6}
Loss function	Categorical Cross Entropy
Dropout keep probability	0.5
Train/test split	80%/20% yielding 12000/3000 sentences
Model Monitors	Reduce Learning Rate on Plateau, Early Stoppage
Performance metrics	Categorical Accuracy, F1 Score and Precision

3.3.1.3 Model Performance and Analysis

The model did not do very well on our training data, the accuracy did improve over time compared to LSTM models that we will discuss in the next chapter, but it was taking a very long time to train and it was bound to overfit, figure 3.3 shows the accuracy improvement throughout the epochs of this model.

3.3.1.4 Model Improvement: More Channels, Less Filters

After careful analysis of the above model, precisely the representation of our training data. We concluded that the word embeddings consist of multiple channels distributed

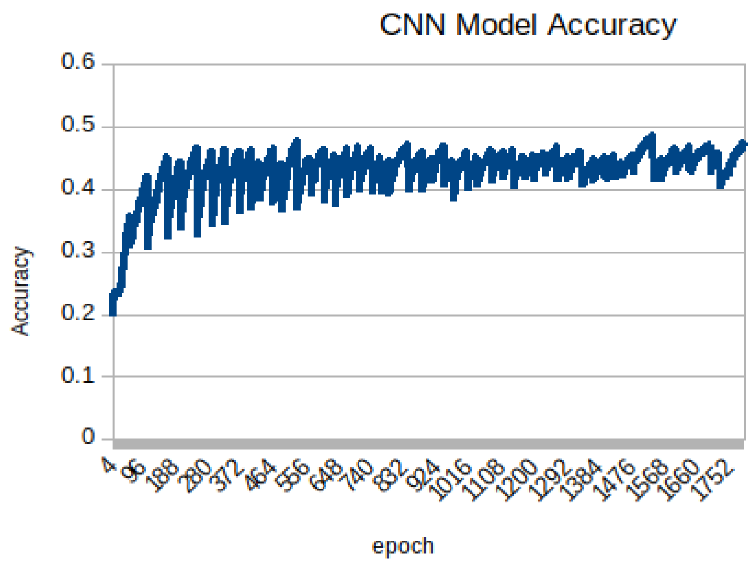


Figure 3.3: CNN model training accuracy

over relatively short sentences as opposed to images which consist of either one (the case of grey scale images) or three (in the case of RGB images). With this note, we decided the redo our model to have a larger number of channels and a smaller filter, to have a valid comparison with the above model, we simply exchanged the number of filters from the previous model to be the number of channels, and fixed the size of the filter to be 10, figure 3.4 shows the structure of the revamped model and table 3.2 shows the updated parameters.

Table 3.2: Updated CNN Hyper Parameters

Parameter	**Value**
Number of Convolutional Layers	4
Filter Sizes	10, 10, 10, 10
Number Of Channels	75, 60, 40, 25
Maximum Sequence Length	1000
Embedding algorithm	GloVe
Embedding dimension	100
Total parameters	4,320,005
Trainable parameters	717,805
Optimizer	Adam Optimizer With Learning Rate=10^{-4} And a Decay Of 10^{-6}
Loss function	Categorical Cross Entropy
Dropout keep probability	0.5
Train/test split	80%/20% yielding 12000/3000 sentences
Model Monitors	Reduce Learning Rate on Plateau, Early Stoppage
Performance metrics	Categorical Accuracy, F1 Score and Precision

3.3.1.5 Improved Model Performance

The revamped model confirmed our theory and did a much better job classifying the input data in significantly fewer epochs, we were able to achieve a training accuracy of 0.870 and a training loss of 0.315, the metrics for the validation were: validation accuracy of 0.75 and validation loss of 0.72 after 975 epochs. Figures: 3.5 to 3.8 show a plot of the aforementioned metrics over time, table: 3.3 shows a summary of the metrics obtained after training the model, and table 3.4 shows the final confusion matrix of this model. Lastly, table 3.5 shows some of correctly and incorrectly classified samples, we can note from this table that the CNN maintains its well-known property of being immune to changes in the order of the input [2], this is shown

[2] i.g: if the image is flipped the CNN can still detect the desired object in the picture

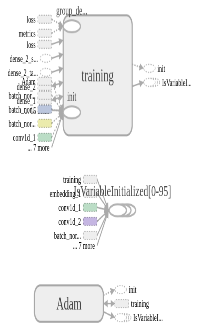
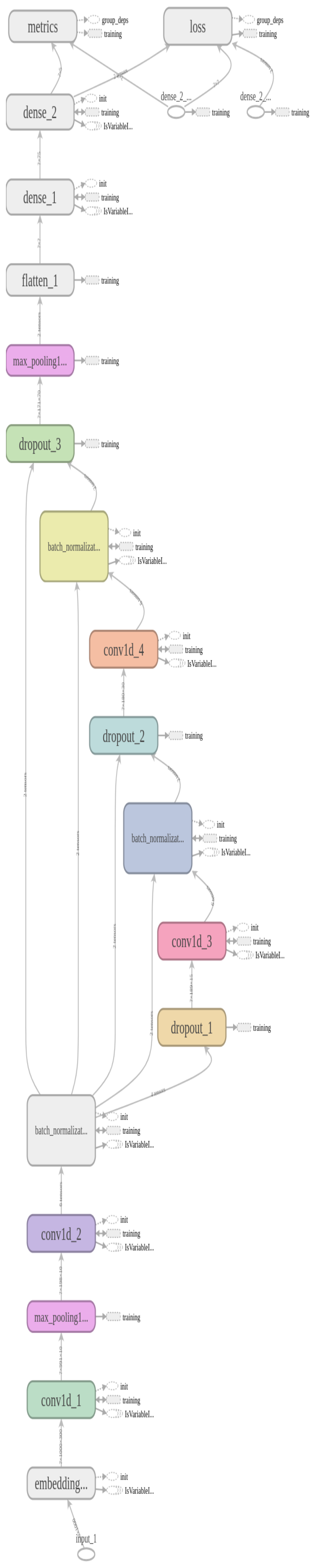



Figure 3.4: CNN model with wide channels

in row 4 of table 3.5 where we created a random permutation of row 3 and the CNN was still able to detect the correct genre with a very similar probability of 0.73 for the original sentence and 0.71 for the permuted sentence.

Given these results, we decided not to invest more time optimizing this model as the HAN model showed a greater promise and hyper parameter tuning for this model didn't lead to a validation accuracy higher than 75%.

Table 3.3: Performance Metrics Of The Improved CNN Model

Metric	**Value**	**Dataset**
Loss	0.315	Train
Accuracy	0.872	Train
Loss	0.72	Validation
Accuracy	0.75	Validation
F1	0.76	Validation
Precision	0.81	Validation
Recall	0.72	Validation

Table 3.4: Confusion Matrix Of The Improved CNN Model

Class	**Action**	**Comedy**	**Drama**	**Politics**	**Romance**
Action	470	25	28	25	10
Comedy	118	405	17	43	47
Drama	71	21	475	38	19
Politics	85	31	39	397	49
Romance	56	43	10	38	440

Table 3.5: Sample prediction results from the improved CNN model

Sentence	**Correct class**	**Predicted class**
well you may ask how may i know when i am in love	romance	romance
...Tammy Two: Hey Jer Bear! What are you doing with these two jabronies? Jamm: Tammy, I've given this a lot of thought, we should break up.Tammy Two: Hahahahaha! What's the matter little boy? The bad people get to ...	comedy	comedy

tony soprano about killing a person you know come ta think of it you never popped your cherry in that regard right bobby baccilieri no tony soprano yet your old man was the *** terminator	drama	drama
a no your know bobby right regard killing soprano come cherry of ta think old you was about that in yet popped never tony your man it soprano terminator tony person you *** the baccilieri [3]	drama	drama
[His voice very hoarse, from his filibuster] There's no compromise with truth. That's all I got up on this floor to say. When was it? A year ago, it seems like....Just get up off the ground, that's all I ask. Get up there with that lady that's up on top of this Capitol dome, that lady that stands for liberty. Take a look at this country through her eyes if you really want to see something. And you won't just see scenery; you'll see the whole parade of what Man's carved out for himself, after centuries of fighting. Fighting for something better than just jungle law , fighting so's he can stand on his own two feet, free and decent, like he was created, no matter what his race, color, or creed. That's what you'd see. There's no place out there for graft, or greed, or lies, or compromise with human liberties. And, uh, if that's what the grownups have done with this world that was given to them, then we'd better get those ...	politics	politics
tony jumps something just touched my foot something's under the couch mcgee maybe it's the uh crime scene fairy tony tony shush i hate halloween,	action	politics
... even in times of trauma we try to maintain a sense of normality until we no longer can that my friends is called surviving not healing we never become whole again we are survivors ...	politics	romance

[3]This row is a random permutation of the row above

Lucille: I just pray it's one of those things where he's unconscious through the whole trial and when he wakes up he gets BIG toy! Michael: Did you do this, Mom? Did you put one of your own sons in a coma so he wouldn't testify?	comedy	romance
This is golden, Tiffany, golden. Two more people. He would have given me two for it, at least one. He would have given me one, one more. One more person. A person, Stern. For this. I could have gotten one more person, and I didn't. And I didn't! Congratulations ... You have been liberated by the Soviet Army on Xmas night!	drama	politics
Morale was deteriorating and it was all Yossarian's fault. The country was in peril; he was jeopardizing his traditional rights of freedom and independence by daring to exercise them.	politics	action

3.3.1.6 Parameter Training

We tried varying the filter size between 10 and 30, the increase in the filter size caused the network to overfit and capped the validation accuracy to about 50% and the training accuracy to about 70%, we recommend 10 to be the filter size when using the embedding of size 100. Increasing the number of channels to 100 75 60 40 had a similar effect where the training accuracy was capped to below 70% and validation accuracy to below 40%.

As for the dropout keep probability, any number between 0.4 and 0.6 had very little effect on the accuracy of the model, finally, we did not adjust the values of the learning rate manually as it is handled automatically by the "Reduce Learning Rate on Plateau" plugin as described in Chapter 4

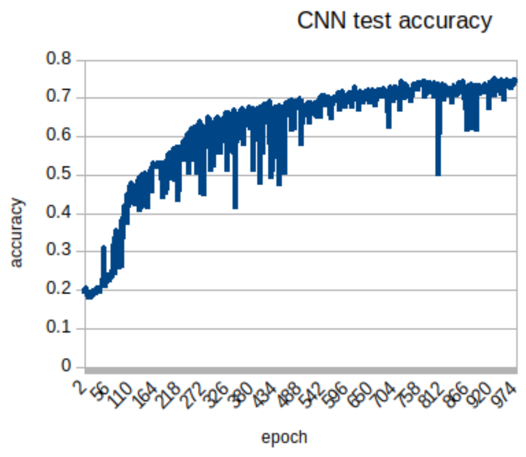


Figure 3.5: Improved CNN validation accuracy (higher is better)

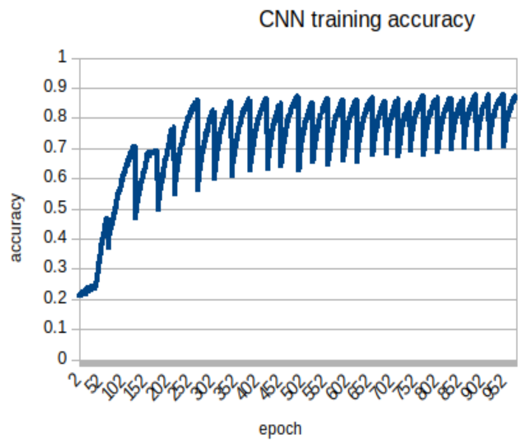


Figure 3.6: Improved CNN training accuracy (higher is better)

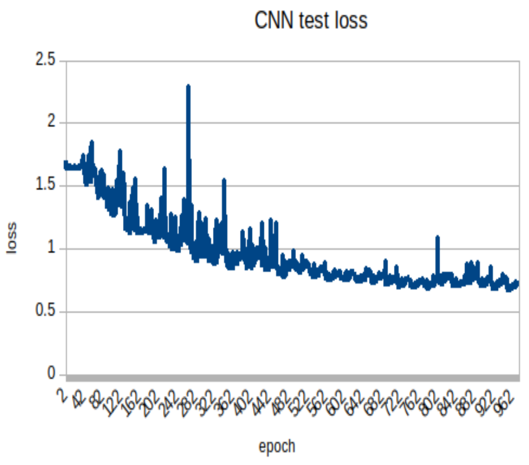


Figure 3.7: Improved CNN validation loss (lower is better)

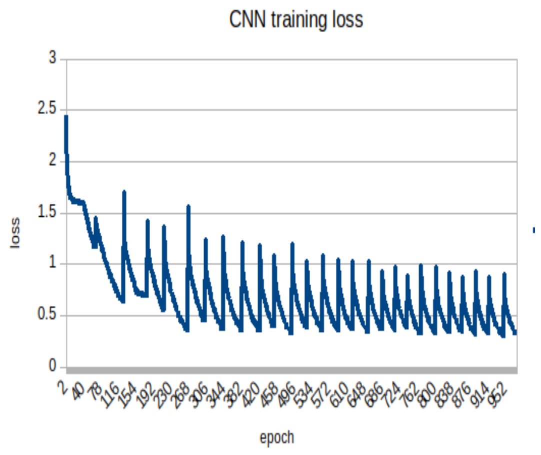


Figure 3.8: Improved CNN training loss (lower is better)

3.3.2 Hierarchical Attention Model

Another architecture that showed great promise in our experiments is the hierarchical attention model. We chose this model after a careful analysis of our training data, which lead us to find that some of the sentences provided as a feature vector to our contribute much more to the particular genre class than the others. For instance, consider the following two sentences taken from the drama genre input:

Example 1: A text from Dr. House

Dr. House: [to Dr. Cameron] Is he Canadian?
Dr. Cameron: He's a low priority.
Dr. House: Is that a yes?

Example 2: A text from Godfather

Don Vito Corleone: [Sobs for a moment before he regains his composure] I want no inquiries made. I want no acts of vengeance. I want you to arrange a meeting with the Heads of the Five Families. This war stops now.

Looking closely at the above examples, it is clear that, not only, the second sentence contributes much more to the drama genre than the first one, but also, there are particular keywords like "sobs", "vengeance" and "war" in the second example that give clearer clues to the drama genre. The Hierarchical attention networks excel in such cases as they utilize two attention vector, the first works as a conventional attention vector described in section 2.4.3 and the other works as an attention vector for the training data itself.

3.3.2.1 Input Data

In order to use a hierarchical attention network, we needed a modification to the way we represent our training data, the previously used representation is not suitable to be used with a time distributed layer which is the main building block for a hierarchical attention network, as the layer expects the data to be of three or more dimensions. The updated input representation is of shape (total number of training data, total number of sentences (with an upper limit), total number of words(with an upper limit)). We chose the upper limit for the sentence length to be 100 and limit our vocabulary to 20000. Another advantage of using this model is that it eliminates the need to pad the training data since it is capable of working with dynamic length sentences.

3.3.2.2 Model Architecture

The model is easier to be explained when thought of as two separate models, the first part (the encoder) consists of conventional recurrent neural network (RNN) built with a bi-directional long short-term memory (BLSTM) layer with size 300 followed by an attention layer of the same size, this model is responsible for encoding the input data, it takes the word embeddings as input, and outputs an encoded representation of the words based on the hidden states of the BLSTM cells. The output of the encoder is then fed to the second part of the model, which in turn starts off with a time-distributed layer, this layer is the core of the HAN network, the purpose of the time-distributed layer is to run a copy of the encoder on each input sentence, the time-distributed layer is followed by a BLSTM layer of sizes 300 and an attention layer of the same size. Finally, the model adds a softmax fully connected layer as the output layer. The full architecture of the model is outlined in figure 3.9.

For this model, we chose GloVe word embeddings of dimension 100 due to the high memory requirements of this model, the optimizer of choice is RMSProp with a learning rate of 10^{-3} and a gradient decay of 0.9. The loss function is categorical cross entropy, and the performance metrics chosen are Accuracy, F1 score and Precision. Table 3.6 contains a list of the model's parameters.

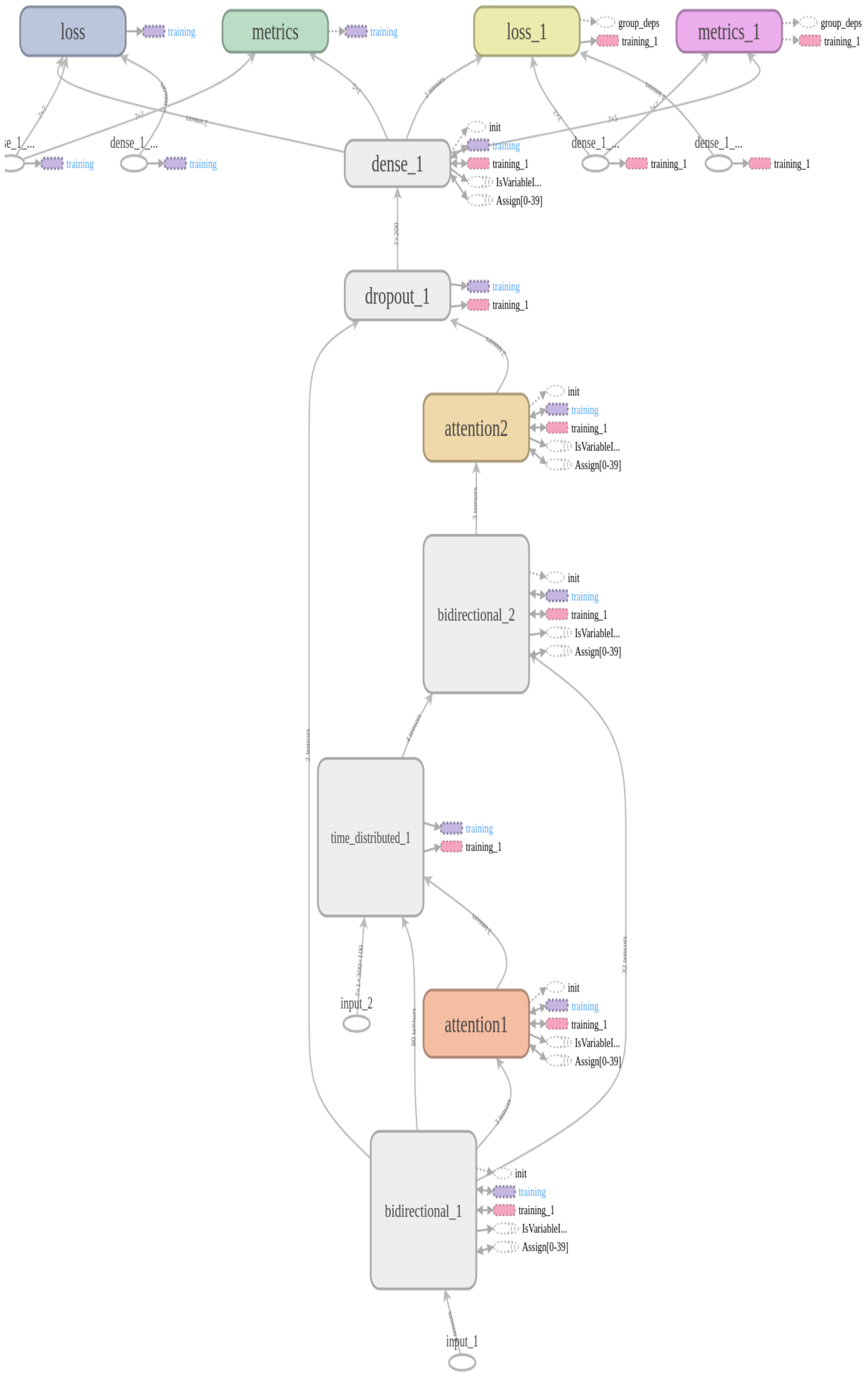


Figure 3.9: Hierarchical attention network architecture

Table 3.6: HAN Hyper Parameters

Parameter	**Value**
Number of BLSTM Units	1
LSTM Hidden Units	300 (The actual size is multiplied by 2 since the layer is bi-directional)
Maximum Sentence Length	300
Maximum Number of words	20000
Embedding algorithm	GloVe
Embedding dimension	100
Total parameters	1,323,600
Trainable parameters	1,323,600
Optimizer	RMSProp Optimizer With Learning Rate=10^{-3} And a Decay Factor Of 0.9
Loss function	Categorical Cross Entropy
Dropout keep probability	0.6
Train/test split	80%/20% yielding 9000/3000 sentences
Model Monitors	Reduce Learning Rate on Plateau, Early Stoppage
Performance metrics	Categorical Accuracy, F1 Score and Precision

3.3.2.3 Model Performance

The HAN model achieved a training accuracy of 0.93 and a training loss of 0.20 after training for 67 epochs, the test metrics values were: test accuracy of 0.89 and test loss of 0.28. Figures: 3.10 to 3.13 show a plot of the aforementioned metrics over time, table 3.7 shows a summary of the performance metrics for this model and table 3.8 shows the confusion matrix of this model, and finally, table 3.9 shows some examples of correct and incorrect predictions of this model.

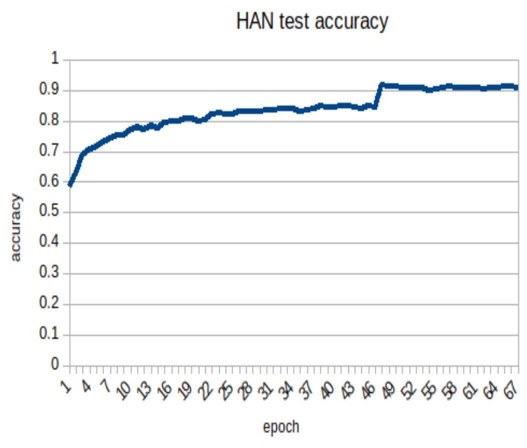


Figure 3.10: Hierarchical attention network validation accuracy (higher is better)

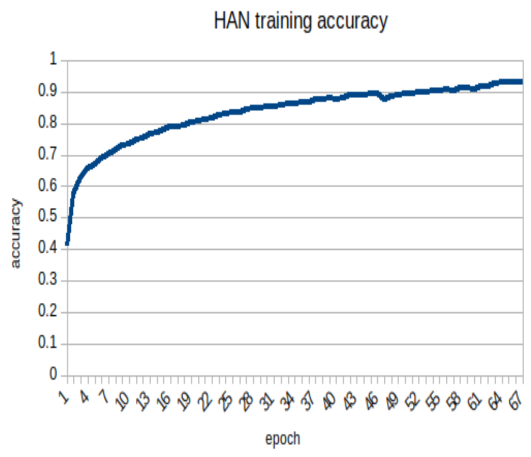


Figure 3.11: Hierarchical attention network training accuracy (higher is better)

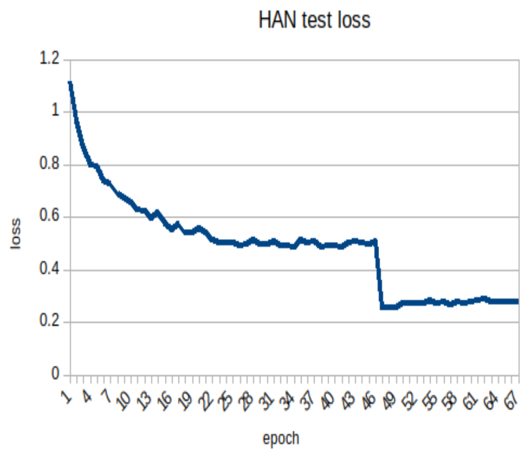


Figure 3.12: HAN validation Loss (Lower is Better)

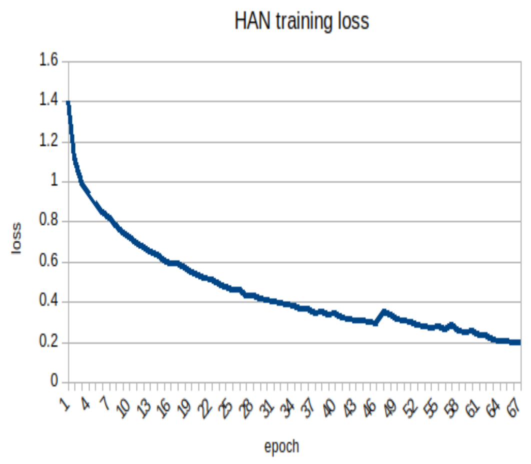


Figure 3.13: Hierarchical attention network training loss (lower is better)

Table 3.7: HAN Model Performance Metrics

Metric	**Value**	**Dataset**
Loss	0.20	Train
Accuracy	0.93	Train
Loss	0.28	Validation
Accuracy	0.884	Validation
Accuracy	0.891	Test
F1	0.883	Test
Precision	0.882	Test
Recall	0.887	Test

Table 3.8: HAN Model Confusion Matrix

Class	**Action**	**Comedy**	**Drama**	**Politics**	**Romance**
Action	**555**	27	6	34	5
Comedy	11	**510**	5	39	41
Drama	7	37	**500**	19	12
Politics	24	39	6	**504**	25
Romance	7	47	7	22	**511**

Table 3.9: Sample prediction results from the HAN model

Sentence	**Correct class**	**Predicted class**
Being with you today is worth all the broken hearts of yesterday.	romance	romance
Leslie: Lucky for me, I've processed all my feelings. And I've gone through the five stages of grief: Denial, anger, internet commenting, cat adoption, African dance, cat returning to the adoption place, watching all the episodes of Murphy Brown, and not giving a flying fart...How many stages it that? I don't know, the point is I'm fine now.	comedy	comedy
flying it anger, And many Lucky internet I've fine I've commenting, Brown, returning giving me, African and of the five a cat stages grief: to my adoption, Leslie: is watching place, for processed episodes now. all cat point don't Murphy adoption the the dance, through fart ... How the that? of feelings. I not stages gone I'm Denial, know, all [4]	comedy	comedy

[4]this is a random permutation of the row above

Rod Serling: Mr. Roger Shackleforth. Age: youthful twenties. Occupation: being in love. Not just in love, but madly, passionately, illogically, miserably, all-consumingly in love, with a young woman named Leila who has a vague recollection of his face and even less than a passing interest. In a moment you'll see a switch, because Mr. Roger Shackleforth, the young gentleman so much in love, will take a short but very meaningful journey into the Twilight Zone.	drama	drama
Having saved a SEAL from being killed by the Chameleon, the Five-0 team are invited into a secret room inside JFB Pearl-Harbor Hickam] Danny: So what, you're not gonna tell me about Operation Strawberry Field? Steve: No. Danny: No, no, 'cause you'd have to kill me if you told me. Steve: [deadpan] Keep that up.	action	action
Prosecutor: The defendant's request for temporary release from federal custody to attend his daughter's wedding is ludicrous. Mr. Sacrimoni is a known member of organized crime at the helm of a vast criminal conspiracy. Defendant: I notice you're wearing a wedding ring, Miss Vaughn. Was your father at your wedding? Prosecutor: My father wasn't awaiting trial on forty seven RICO predicates including murder.	drama	politics
Mike: Hey, when did we become one of those couples who let our rat babies control our lives?	comedy	politics
Lucille: I just pray it's one of those things where he's unconscious through the whole trial and when he wakes up he gets BIG toy! Michael: Did you do this, Mom? Did you put one of your own sons in a coma so he wouldn't testify?	comedy	romance

This is golden, Tiffany, golden. Two more people. He would have given me two for it, at least one. He would have given me one, one more. One more person. A person, Stern. For this. I could have gotten one more person, and I didn't. And I didn't! Congratulations ... You have been liberated by the Soviet Army on Xmas night!	drama	politics
At Yale once, they held an auction. There was this woman and her name was Lulu Landis. Her postcards came up for sale. She had 1400 postcards written to her and I'd never heard of her before but I knew I had to have those cards, I had to know why anyone would get so many messages. I paid sixty-five dollars for them... I got all crazy trying to work it out and first it was just a maze but then I found that her husband killed himself in Dayton, and once I had that, it all began to open, an evangelist had come to Dayton and his horses hit Lulu Landis at the corner of 13th and Vermillion and she was paralyzed. Permanently, and her favorite thing til then had been traveling and all her friends, whenever they went anyplace, they wrote her. Those cards, they were her eyes...	politics	drama

The HAN model just as the CNN, is immune to the order of the words in a sentence, although the attention values do seem to change when the order of the words changes, which is understandable considering that the LSTM cells are order-sensitive, in general, we do not recommend using this model if there is a chance of having un-ordered sentences as inputs. Another important aspect of the HAN is that it is possible to get the attention values associated with each individual word which makes it possible to get a better intuition on how the model decided to classify a particular sentence into a particular genre, Figures 3.14 and 3.15 show examples of the attention acquired when we used the model to classify the comedy and politics example from table 3.9.

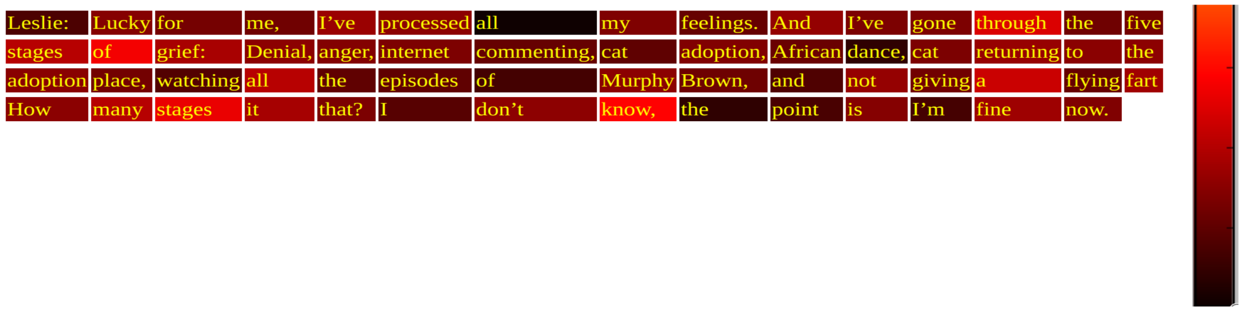


Figure 3.14: Attention values for the comedy genre example

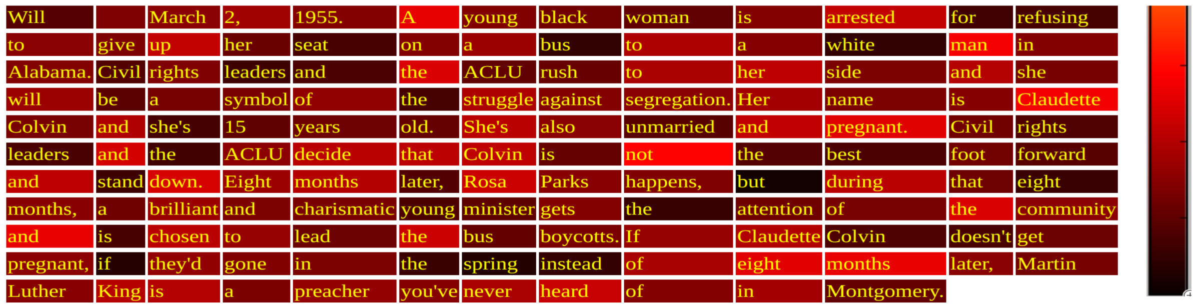


Figure 3.15: Attention values for the politics genre example

Finally, we tested our model on scenes extracted from the movie "**Godfather**" (full script: http://www.dailyscript.com/scripts/The_Godfather.html), the two most dominating genres were: action with 24% of the total scenes and drama with 32% of the total scenes, the movie "**Dumb and Dumber**" (full script: https://www.imsdb.com/scripts/Dumb-and-Dumber.html) had 63% of the scenes belonging to the comedy genre and about 10% for all other genres, finally, the movie "**Lincoln**" (full script: https://www.imsdb.com/scripts/Lincoln.html) had 64% scenes classified as politics and 17% as action.

3.3.2.4 Parameter Training

As opposed to the CNN model, increasing the LSTM size has neither a positive nor negative affect on the model albeit is much more resource intensive, we decided to stick to using 300 units due to that and to be able to skip using the embedding layer. Stacking LSTM layers does help reaching >= 90% accuracy as well and with a smaller number of epochs, this is a good indicator that the model can scale well when adding more data, however, it must be noted that stacked LSTMs as very slow to train and resource intensive, the performance aspects are discussed in more details in Comparison Between HAN and CNN Models.

Lastly, the other parameters had little or no effect on the overall accuracy of the model.

Chapter 4

Models and Scalability Analysis

One of the goals that deep learning models seek is achieving greater accuracy when more data is available, in this chapter, we analyze and make predictions about the performance of the two models we discussed in chapter 3. We also briefly discuss our trials with vanilla recurrent neural networks and how they were not suitable for this kind of classification although they are a standard for text classification and NLP in the literature.

4.1 Comparison Between HAN and CNN Models

Taking a closer look at Figures 3.5 and 3.10. We see that the HAN model achieved a much higher accuracy. Moreover, the fluctuations of the accuracy measure in the CNN and the higher number of epochs it took to achieve such accuracy plays in the favor of the HAN model. Another indicator that supports HAN over CNN is the changes in the learning rate, a learning rate of 10^{-3} was unchanged for almost all the epochs of the model, compared to a more volatile learning rate the CNN model used, keeping in mind that the initial learning rate for the CNN was 10^{-4}. The changes in the learning rate, caused by Kera's callback ReduceLROnPlateau, are an indicator of the model stagnating.

One aspect where the CNN model has a clear advantage is the cost of training. The HAN model is significantly heavier than the CNN model both in terms of memory requirements and training speed, the CNN model takes an average of three minutes to train for one epoch, compared to an average of half an hour per epoch for the HAN model. One reason for this is that the HAN model has such huge memory requirements that it is practically impossible to train it on a GPU, all our epochs for this models were trained using CPUs and were notably slower.

Based on the above analysis of both models, we recommend using HAN networks for greater accuracy, and if training data is expected to scale, the model's accuracy can benefit from the extra training data without any significant changes in the model's architecture. On the other hand, if resources are an issue, CNN models are an efficient and sufficiently accurate method for this use case.

4.2 LSTM Models and Results

We chose to dedicate this section to discuss out experiments with LSTM and attention models, as they are recommended by many machine learning practitioners for NLP and text classification problems. We found that in our particular case, these models did not perform well, this section outlines some of the architectures we experimented with and their performance.

4.2.0.1 Vanilla BLSTM

A vanilla bi-directional LSTM model was one of the first models we experimented with, the reason being that it is one of the most researched and recommended models in the literature for NLP related classifications. The model of interest consisted of 3 BLSTM layers of sizes 75 30 20, the word embedding algorithm was word2vec with an embedding dimension of 300, the dropout keep probability is 0.6. A layer of batch normalization was added after each BLSTM layer. Figure 4.1 shows the full structure of the model.

The performance of the model was below expectations, the model was unable to achieve an accuracy higher than 0.22 after training for over four thousand epochs, moreover, the model was unable to guess the proper distribution of the output classes.

4.2.0.2 Attention BLSTM

The attention BLSTM model works in a similar manner of the vanilla BLSTM model with the addition of an attention vector. This vector is responsible for assigning a set of scores for parts of the input, this attention will ideally be lower for stop words that do not affect the meaning of the sentence and a higher attention for more significant words. In terms of architecture, the model has a similar architecture of the aforementioned BLSTM model with the addition of the attention vector, figure 4.2 shows the full architecture.

In term of performance, the attention model did not perform much better in comparison to the BLSTM, the accuracy 0.2 after 1000 epochs, it did not fix the genre distribution either.

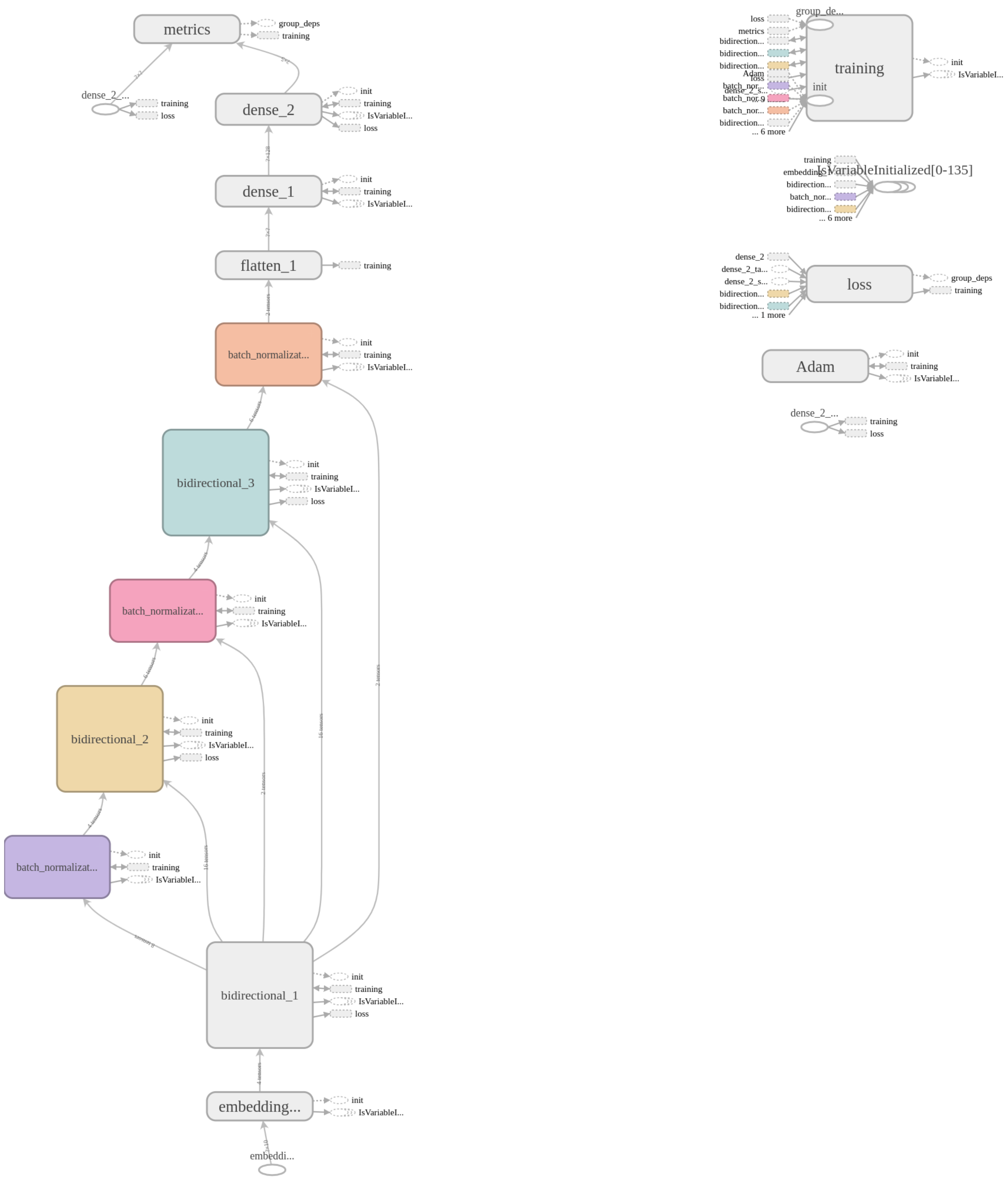


Figure 4.1: BLSTM model architecture

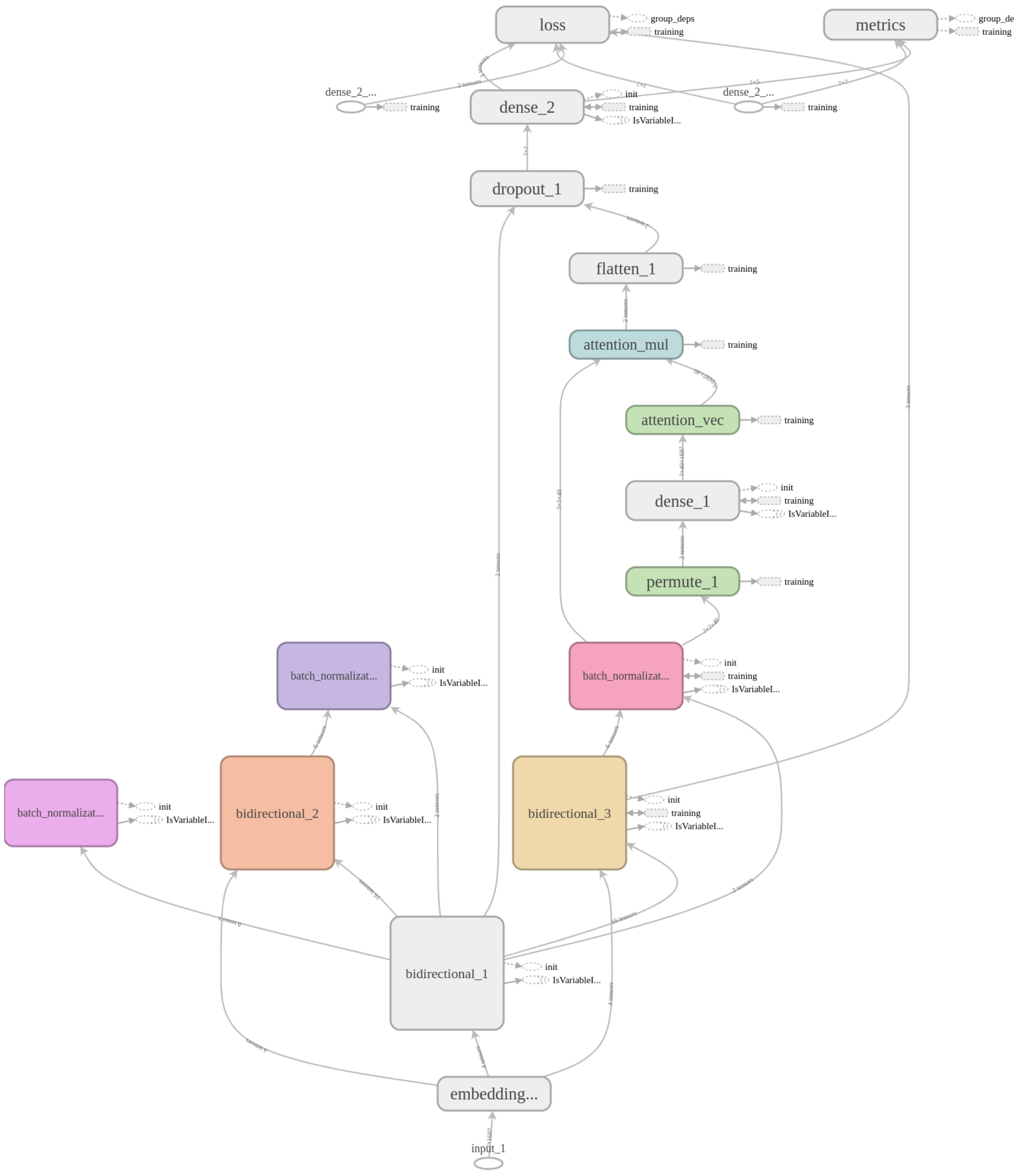


Figure 4.2: Attention model architecture

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this research, we explored a plethora of neural network architectures with the target of building a model capable of predicting a genre of a text corpus. Our dataset consists of quotes from famous movies from each genre taken mainly from wikiquote.com website and complemented from goodreads.com.

We started our research using the industry standard recurrent neural networks which did not perform as good as we expected for this dataset. The same can be said about attention-based recurrent networks. CNN and HAN networks gave much better results.

Comparing HAN and CNN models, we concluded that the HAN model wins in terms of accuracy, we predict it can scale better with more training data without any major changes to the network architecture. In terms of performance, the CNN models is a clear winner, its memory requirements allow the model to be trained on GPU without any issues and the training time is significantly lower.

Finally, looking at the confusion matrices table: 3.4 and 3.8, we see that "action" and "comedy" have high confusion, we suspect that the reason is that, generally, the action movies have comedy as a sub-genre, keeping in mind that the dataset we used being quotes picked up from movies without any sort of manual correction. The same reason would also justify the confusion between comedy and romance as well. In general, the comedy genre is difficult to be detected by machines as it can be very context dependent and it can be a source of confusion. Another notable confusion is between the genres action and drama, which also can be attributed to the dataset as many of the action movies have a drama aspect and vice versa.

5.2 Limitations and Future Work

The main limitations of doing this research were resources and data set. As for the dataset, we depended on the fact that movies labeled as "action" would mostly contain quotes that will belong to the "action" genre, which is true in most cases, however, we found that some of our training data contains quotes that we felt did not belong to the assigned genre, but it was not possible to clean the whole dataset due to its size.

As for the resources, since deep learning is a relatively young field, most frameworks do not support using a mix of CPU and GPU for training. Being resource intensive in nature, we ran into problems deepening our models to a certain extent, one of the models we tried was in fact trained completely on CPU and took a very long time.

Thus, one of our future targets is optimizing our models in order to be more production-ready and faster to train, this could be done by either using a lower level part of our framework of choice (keras) or, if needed, changing the internal structure of the cells.

We also would like to expand our vocabulary to include all the words from our word embedding vector rather than the words appearing in the training set, doing so would allow us to detect the genre of any text corpus by dividing it into logical sections, evaluate each section separately and use the normalized classifications as the final genres.

Another part we would like to dig deeper in is evaluating the models on various other genres, sci-fi and superhero are some of the extra genres we can try.

Finally, We would like to run the same model for foreign films, this is particularly challenging mainly due to the lack of a standard pre-trained word embeddings model for every language. One of the promising projects we found is offered by Carnegie Mellon University.

References

[1] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263. Asian Federation of Natural Language Processing, 2017.

[2] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org, 2015.

[3] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics, 2016.

[4] D. Li and J. Qian. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 471–475, Oct 2016.

[5] Darin Brezeale and Diane J. Cook. Using closed captions and visual features

to classify movies by genre. In *In Poster session of the Seventh International Workshop on Multimedia Data Mining (MDM/KDD2006*, 2006.

[6] Peng-Yu Chen and Von-Wun Soo. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[7] Rada Mihalcea and Carlo Strapparava. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 531–538, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[8] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc., 2015.

[9] Y. Xiao and K. Cho. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. *arXiv e-prints*, January 2016.

[10] Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics, 2016.

[11] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *Computing Research Repository*, abs/1703.01898, 2017.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[15] F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194 vol.3, July 2000.

[16] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

[17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct 2017.

[18] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1243–1251. Curran Associates, Inc., 2010.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473, September 2014.

[21] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics, 2015.

[22] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computing Research Repository*, abs/1301.3781, 2013.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[26] Timothy Dozat. Incorporating nesterov momentum into adam. Stanford University, Tech. Rep., 2015. [Online]. Available: http://cs229.stanford.edu/proj2015/054 report.pdf.

Appendix A

List Of Movies and Series

Action	**Drama**
Die Hard	The Shawshank Redemption
Die Hard 2: Die Harder	The Godfather
A Good Day to Die Hard	The Godfather Part II
The Terminator	The Godfather Part III
Terminator 2: Judgment Day	Schindler's List
The Bourne Identity	12 Angry Men
Face/Off	Fight Club
Lethal Weapon	Seven
Lethal Weapon 2	The Silence of the Lambs
Lethal Weapon 3	Hannibal
Mission: Impossible	Forrest Gump
Mission: Impossible II	The Help
Mad Max: Fury Road	Goodfellas
Casino Royale	Oz
Predator	The Americans
skyfall	The Walking Dead
RoboCop	The Sopranos
300	Lost
Daredevil	The Twilight Zone
Agents of S.H.I.E.L.D.	Sherlock (TV series)
Iron Fist	House
Hawaii Five-0	Dexter
Prison Break	Citizen Kane
NCIS	Moonlight
	12 Years a Slave
	Metropolis
	The Maltese Falcon
	Sunset Boulevard
	The Seven Samurai
	Touch of Evil
	The Color Purple
	Game of Thrones
	The Wire
	Black Mirror
	Breaking Bad

Comedy	**Politics**
Dumb and Dumber	All the President's Men
Dumb and Dumber To	Mr. Smith Goes to Washington
Dumb and Dumberer: When Harry Met Lloyd	Wag the Dog
Rush Hour	The Manchurian Candidate
Rush Hour 2	Frost/Nixon
Rush Hour 3	House of Cards
The Blues Brothers	The West Wing
Raising Arizona	24
Planes, Trains and Automobiles	Scandal
Office Space	Yes, Minister
Anchorman: The Legend of Ron Burgundy	Madam Secretary
The Jerk	Veep
This Is Spinal Tap	Saturday Night Live
The Hangover	The Newsroom
The Hangover Part II	Boardwalk_Empire
The Hangover Part III	Wolf Hall
Tommy Boy	
Superbad	
Silicon Valley	
Friends	
Parks and Recreation	
The Office	
Arrested Development	
30 Rock	
The Office	

Romance
Sex and the City
One Tree Hill
The Love Letter
There's Something About Mary
Love Actually
Remember Me
Casablanca
A Walk to Remember
The Notebook
True Romance
When Harry Met Sally...
It Happened One Night
The Red Shoes
Shakespeare in Love
Midnight in Paris
Before Midnight
Sense and Sensibility
The Little Mermaid
Eternal Sunshine of the Spotless Mind
The Vampire Diaries
Love Story
Titanic
Pride and Prejudice