2018

That Seems Made Up: Deep Learning Classifiers for Fiction & Non Fiction Book Reviews

Clement Manger
Technological University Dublin

That Seems Made Up

Deep Learning Classifiers

for Fiction & Non Fiction Book Reviews

Clement Manger

A dissertation submitted in partial fulfilment of the requirements of

Dublin Institute of Technology for the degree of

M.Sc. in Computing (Data Analysis)

24th January 2018

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analysis), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institutes guidelines for ethics in research.

Signed:

Date:

Abstract

The thesis aims to take the first step towards automated extraction of the information found in book reviews, by using machine learning tools to assign a label of fiction or non fiction to the text. The thesis makes use of neural networks and performs experiments around architecture, hyper-parameters and text processing from which an optimized model is produced. The thesis enjoys certain successes; it was possible to match the state of the art achieved by (Kim, 2014) and computation was sped up considerably from the default to the optimized model by 13.8 seconds per 50 steps. Further it is confirmed by the thesis that labelling a sequence as fiction or non fiction can be performed most accurately with LSTM architectures and that contrary to (Reimers & Gurevych, 2017) every considered hyper parameter had a considerable impact on results.

Keywords: Long Short-Term Memory, Text Classification, Word Embeddings

Acknowledgments

- Dr Pierpaolo Donido for his support, critical eye and good humor.
- Dr Robert Ross for his deep learning expertese and generosity in sacrificing personal time for someone not his supervisee.
- Dr Luca Longo for his support and understanding throughout personal difficulties.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background

Consumers face decisions every day and make these decisions based on an understanding of what will best serve their needs and tastes. Whether it be media, clothing, consumables or otherwise: the wealth of peer review information available for consumers has made review information one of the main factors that influences the final decision a consumer makes. The form that most online reviews take, for example on Amazon.com, is simply an abstract numerical rating (such as star rating) and/or a review text. When a product may have many thousands of reviews, this leads to a problem for many consumers. A star rating provides a very coarse understanding of how suitable the product is; as it is abstracted to the point that no direct product features are discernible. Meanwhile review text has the opposite problem, that it is overly rich and not possible to absorb without a concerted effort to read each text.
For many consumers this is not practical and the review text, which contains a reflection closer to the ground truth of peers opinions on the product, is simply ignored. The aim of this thesis will be to look at applying machine learning techniques aimed at gaining insight into this richer ground truth held in aggregate review texts, by assigning a label of fiction or non fiction to book reviews. Amazon Book reviews are chosen as they offer an especially rich dataset, having been recorded since 1996 and further the dataset is available for research use at http://jmcauley.ucsd.edu/data/amazon/.

1.2 Why we need a Fiction vs Nonfiction label

Preliminary examination of the reviews was carried out to see what kind of motifs are present in the dataset that a human reader might take an interest in which will guide the aims of the research. The dataset are JSON including the review text and certain metadata (see example below)


```
{"reviewText": "I bought this for my husband who plays the piano. He is
    having a wonderful time playing these old hymns.
  The music is at times hard to read because we think the book was published
      for singing from more than playing from. Great purchase though!",
  "overall": 5.0,
  "reviewerID": "A2SUAM1J3GNN3B",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"}
```


Explanation: "reviewText" - full text of the review including all characters, unfiltered and includes non UTF Characters, overall - Abstract numerical rating out of a possible 5, reviewerID" - ID allowing for a single reviewer to be identified, reviewerName - name, helpful - [LH digit - number of respondents that found this helpful, RH digit - number of respondents overall], summary - a summary title for the review, unixReviewTime - time of review in 24hr with minutes, reviewTime

When a human reads a review, they are looking to find features that they are interested in and an accompanying opinion that will guide their decision making. Immediately visible is that the content and therefore feature space varies enormously by the category a book might belong to. For example, non fiction books reviews might comment on the accuracy/utility of the book whilst a review of a novel might comment on the quality of the plot. Further to this the type of language used between different categories of book varies, with non fiction books generally attracting reviews with a more serious tone. Useful insight into products therefore requires that feature classification

is carried out within a specific category of book. Unfortunately the meta data above do not include labels for category and the first step in providing useful insight becomes dividing the book reviews into fiction or non fiction categories.

1.3 An Intuitive look at Fiction vs Nonfiction Labelling

The dataset is re-examined to discern which intuitions we as humans use when determining if a review belongs to a fiction or non fiction book.

1.3.1 Phrases and Terms

In many cases, a review can be determined as fiction or non fiction by examining short phrases or specific terms.

- *"The third **novel in this series**, 'Catcher Redknapp' **tells the story** of police officer Dale Whitaker in his pursuit of Chicago gangland boss Forest Fenton"*
- *"Jodie Westford has already written **stories of great complexity** and this is no different"*
- *"A practical and well written **guide to carpentry**"*
- *"Although many of the above reviews disagree, I found Gary Ipswitchs **explanation of thermodynamics** to be edifying and useful*

For a human reader it is these phrases, combined with a humans contextual knowledge of the terms used that reveals the answer.

1.3.2 Long Term Dependences

In some cases, a review cannot be determined as fiction or non fiction without reading the entire text and finding long term dependencies that reveal the answer.

- *"This should be sold as fiction. It's obvious the author spent no time researching or caring enough about the subject."*
- *"The author of this book has taken considerable artistic licence when telling this tale. In this readers opinion any departure from the facts does not help the quality of the book."*

Interestingly, a human reader might be confounded by a phrase that implies one category but in fact the wider context and dependencies reveal it to be the opposite (the following are hypothetical examples).

- *"Unreadable, the author should have picked up a* ***guide to carpentry****, rather than waste his time on such wooden prose."*
- *"A complex book, though to enjoy it one hardly needs a full* ***explanation of thermodynamics****"*

1.3.3 Further Intuitions

Reviews vary greatly in length with the longest at 2382 words and the shortest at 23. Initial examination suggests that non fiction book reviews are more prone to be longer as their subject matter can be concerned with concepts of great detail, however this may not be the case in aggregate. Frequent use of proper nouns seems to be associated with a higher likelihood of the book being fiction (as characters and places are often given in a brief summary). Again however this might confound a human as certain non fiction books also have a higher prevalence of proper nouns, for example biographies. With these intuitions in mind, related literature is considered to help frame the research. Literature will be considered that offers machine learning frameworks in natural language processing that best suit the dataset, task intuitions outlined above.

1.3.4 Literature review summary that informs the research question, objectives, methodology, scope and limitations

The literature review revealed that non linear machine learning methods are more effective than linear as manual feature selection is avoided (which risks missing important features difficult to discern as a human). Further two popular architectures are found that reflect the intuitive formations in the text outlined above: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs). Binary sentiment analysis is found to be a comparable and well researched field, with established science as to the optimal methods, hyper-parameter settings and variety of text preprocessing. Vector space representations of words are revealed as an important consideration in this type of natural language processing, with the degree dimensionality and the training set for vectors important factors. A wide variety of architecture considerations with associated hyper-parameters have been the subject of recent research, for example the use of a drop out layer to prevent overfitting. The computational cost of these processes is also of concern to the scientific community, with wide ranges in training times across models. For full details refer to chapter 2

1.4 Research Questions

From this point it becomes possible to frame the research. There are three primary research questions:,

1. Can deep learning models be made to assign a label of fiction or non fiction to a book review text with the same accuracy as state of the art accuracies in long sequence binary sentiment analysis?

2. Are the hyper-parameters found by (Reimers & Gurevych, 2017) to be most important for accuracy in long sequence sentiment analysis the same as those in assigning a label of fiction or non fiction?

3. Does computational speed necessarily come at a cost to a accuracy?

1.5 Research Objectives

1. Build a model which may match the absolute accuracy in binary classification found in (Kim, 2014).

2. Build a model which may match the absolute accuracy in binary classification found in (Barry, 2017).

3. Use experimental results to build an efficient model that produces as high accuracies as possible in as fast a time possible.

1.6 Research Methodology

The research will be carried out using Google Tensorflow, in which a series of models will be built and recorded at different stages of training. Each model will serve as part of an experiment around architecture, hyper-parameter tuning and text preprocessing. Models will be compared based on their accuracy and speed. For full details refer to chapter 3

1.7 Scope and Limitations

- During model training is was discovered that depending on when in the row of models a model is trained, the model appears to be affected. Our suggested hypothesis to explain this is that the GPU is affected by the state of the VRAM, or that the GPU heats up during training. This introduces a confounding variable into the measurement of speed and may also affect accuracy. Possible controls are suggested in section 6.5. however it is beyond the scope of this thesis to control for.

- Further during model training it was discovered that training might be affected by blips in accuracy where, in spite of convergence, accuracy drops again. Our suggested hypothesis is that the model is being faced with a more ambiguous batch of reviews than in other iterations. Again, this is a confounding variable that we may be measuring rather than the intended variables of experimentation. A possible control is suggested in section 6.5. However this is beyond the scope of the thesis to control for.

- K-fold cross validation of the models has not been performed due to time constraints, this would help to confirm the validity of each model and would be the first consideration in future work.

- Manual labelling of the dataset also results in a smaller than ideal size for the sample, however there are examples in literature with smaller datasets: (Kim, 2014) make use of a 3775 size sample in classifying product reviews and (Pang, Lee, & Vaithyanathan, 2002) make use of a 2053 size sample. Furthermore any bias introduced will be lessened by stratified sampling as close to an equal sample of fiction and non fiction reviews as possible.

- A vector space representation of words can be trained on any corpus of texts and this might have been an option for the results of this thesis, however this is not carried out within the thesis for the reason that pre-trained vectors are available from Google and Stanford that are trained on far larger corpuses (over 3 billion word corpus in the case of Google Word2Vec) which we can safely assume will give rise to a more accurate representation of a word than our smaller sample of just book reviews.

- The thesis will not make comparisons between linear and non linear machine learning methods, this is because there is a clear distinction in reviewed literature showing outperformance of deep learning methods. Instead the focus of experiments will be between deep learning methods and their associated hyperparameters and input preprocessing.

1.8 Chapter Summary and Readerś Guide

This chapter gave an overview of the research and explored the motivation behind undertaking the research. For a condensed view of the thesis, the following reader strategy is advised. Reading the format I literature review, the method chapter experimental practice and model design then skipping ahead to the analysis chapter, using the method chapter experiment list and results chapter for consultation when a part of the analysis chapter needs clarification or a figure is referenced. The analysis chapter contains all discussion and reference to the figures found in the results chapter. The following chapter presents the literature that was considered to help frame the research.

Chapter 2

LITERATURE REVIEW

2.1 Literature Review Aims

In approaching the literature review a series of questions needed to be answered:

1. What are the similar problems faced in the field of NLP?
2. Is linear or non linear machine learning suitable in this field?
3. What kind of non linear ML architectures are used and what are their relative merits?
4. What kind of data preprocessing is done in similar fields?
5. How are dependencies in text expressed?
6. How is accuracy reported?
7. What sample size is needed?
8. What is the state of the art?
9. What are the hyper parameter concerns in non linear ML that will affect our experimentation and how do these decisions above affect computational cost?

The literature review is presented in two formats. First, papers are grouped by their implications for the questions above and a conclusion is reached about how this topic

will be explored and experimented on by this thesis. Second, each relevant paper is presented as an outline of its research results with comment offered as to the implications for the thesis.

2.2 Format I - Answering the Literature Review Aims

2.2.1 What are the similar problems faced in the field of NLP?

Labelling a sequence of text with a binary classification is a common problem in natural language processing. (Kim, 2014), (Barry, 2017), (Zaremba, Sutskever, & Vinyals, 2014), (Reimers & Gurevych, 2017), (Komninos & Manandhar, 2016) (Tang, Qin, Feng, & Liu, 2015) look at the classification of a sequence based on binary sentiment - whether a sequence is positive or negative. The intuitions surrounding binary sentiment classification has a great deal in common with a fiction or non fiction label. For example certain terms can be used as an indicator of positivity or negativity but further to this it is possible for subtleties in the text to cloud this. Longer term dependencies such as a negating term earlier in the sequence might reverse a positive word later in the sequence. Much research is concerned with shorter text classification. (Kim, 2014) considers sentence level where (Komninos & Manandhar, 2016) look at tweets. Longer sequences of the kind considered by this thesis are examined by (Barry, 2017) & (Reimers & Gurevych, 2017).

2.2.2 Is linear or non linear machine learning suitable in this field?

Almost every relevant paper involving sequence classification considered makes use of non linear (deep learning methods). Only one paper, (Pang et al., 2002) makes use of solely linear methods. Three papers offer a comparison with linear ML. (Kim, 2014)

offers an extensive comparison of convolutional neural networks against linear methods, scoring 2.1% higher than the SVM (the peak linear ML approach) on a binary movie review sentiment assignment test. (Komninos & Manandhar, 2016) compare SVMs, CNNs and LSTMs, consistently finding the SVM the lowest scoring, the most accurate SVM model scoring 7% lower than the highest deep learning method. (Barry, 2017) directly compares a bag of words SVM to LSTM, finding an 8% increase in accuracy in the LSTM approach with GloVe vetors. (Pang et al., 2002) achieve an 81% accuracy using SVMs, much lower than those of more recent works. Three of the final chapters of discussion in (Pang et al., 2002) are devoted to the problems posed by n-grams, parts of speech and crucially positioning of words in a sentence. All three of these problems are in some way addressed by using non linear ML approaches. For these reasons the thesis will not pursue a linear ML approach. The time will be better spent honing the non linear architecture and engineering hyperparameters.

2.2.3 What kind of non linear ML architectures are used and what are their relative merits?

Architectures used in related research generally fall into two categories, Convolutional Neural Networks (CNNs used by (Collobert et al., 2011), (Zhang & Wallace, 2015), (Denil, Demiraj, Kalchbrenner, Blunsom, & de Freitas, 2014)) and Long Short Term Memory (LSTM used by (Barry, 2017), (Zaremba et al., 2014). Some examples expand on these two by combining them or adding extra layers (Zhou, Sun, Liu, & Lau, 2015), (Li, Jurafsky, & Hovy, 2015)). The convolutional approach makes use of a sliding window that moves over a text and attaches weights to a series of groups (or n-grams) of words. Mid and high level features are found by the conjunction of these n-grams and then an output layer provides the final classification. (Collobert et al., 2011) make use of this configuration and comment on its high accuracy and quick running time, with their system training itself in only 4 seconds and achieving 97% accuracy on a simple Part of Speech tagging task. The LSTM approach makes use of a sophisticated cell chain structure that passes information and weights through a sequence, updating

weights as needed with new information. This allows for long term dependencies in a text to be captured in the model. (Barry, 2017) makes use of these in binary sentiment classification, achieving 96.5% accuracy. (Komninos & Manandhar, 2016) offer a comparison of the two approaches and find the CNN to train more quickly and have higher accuracy in classifying short texts, however intuition would suggest this may be different when the task involves longer term texts. For this reason these two architectures will be experimented with during this thesis.

2.2.4 What kind of data preprocessing is done in similar fields?

Many papers apply some kind of data preprocessing, although only one varies the preprocessing as part of the experimentation. (Clark, 2003) is the first to state that this is explicitly necessary and gives compelling arguments that when content is user generated it is flawed in ways that inhibits machine learning. (Clark, 2003), (Pang et al., 2002), (Kim, 2014), (Reimers & Gurevych, 2017) all make use of stop word removal and lemmatisation. Many papers follow the processing performed by (Kim, 2014) as this achieved state of the art measures in many tasks ((Zhou et al., 2015), (Barry, 2017), (Tang et al., 2015). (Haddi, Liu, & Shi, 2013) experiment with a series of pre-processing procedures: text cleaning (white space removal, expanding abbreviations found in tweets) stop word removal and stemming. These preparations were decided on due to the nature of the text being considered (social media content). Similar intuitions will be applied to the pre-processing of reviews. As such stop word removal stemming will be experimented with. As mentioned in the introduction, proper nouns as a reoccurring motif in book reviews and processing these into a marker will also be experimented with.

2.2.5 How are dependencies in text expressed?

A basic approach to representing text is simply to express words as numbers that are looked up in a dictionary. This is known as a bag of words approach as no information about the dependencies or context between words is stored. However an

area of considerable research in recent years is storing a word as a semantic map of its co-occurrences. The two landmark words on this are (Mikolov, Chen, Corrado, & Dean, 2013) and (Pennington, Socher, & Manning, 2014). (Mikolov et al., 2013) sets out the Word2Vec framework, in which a given word is predicted based on the window of words either side (its semantic context) or a window of words is predicted based on a given word. (Pennington et al., 2014) present the GloVe vector representation, that adds to this concept by adding context from elsewhere in a paragraph rather than its immediate context. Every relevant paper in this field since these works have included some kind of vector representation using one of these methods or an adapted version. Each of these pieces of research have published vector representations for large corpuses of text; Google News and Wikipedia respectively. Alternatively it is possible to train a domain specific vector representation on a corpus of text. This thesis will experiment with different vector representations, but will not train its own vector representations. The reason for this is that those made available by (Mikolov et al., 2013) and (Pennington et al., 2014) are trained on a far larger dataset and outperform by at least 1% in (Barry, 2017).

2.2.6 What sample size is needed?

Sample sizes vary based on the task being performed. Within binary classification (used in this thesis), the smallest is 2053 (Pang et al., 2002) and the largest is 10662. (Kim, 2014) makes use of a 3775 sample for of customer reviews. As this thesis will need to manually label reviews, there will be a time trade off against performing other tasks. The sample will be larger than 2053 of (Pang et al., 2002), with the customer reviews set used by (Kim, 2014) 3775 as the target.

2.2.7 How is accuracy reported in this field?

Absolute accuracy (% correctly classified), F1 measures and AUC are used to report the accuracy of a model. Many papers simply use absolute accuracy as a default: (Kim, 2014), (Zhang & Wallace, 2015), (Reimers & Gurevych, 2017), (Barry, 2017).

Whilst other measures are used, absolute accuracy seems the standard approach. This thesis will use absolute accuracy and avoid other measures.

2.2.8 What is the state of the art?

Papers that consider examples of sequence classification achieve accuracies ranging between 81.5% and 96.5% when using non linear ML techniques. (Kim, 2014) achieves 81.5% on an IMDb dataset of 10662 positive and negative review sentences and 85% on a dataset of 3775 customer reviews of products. (Barry, 2017) achieves 96.5% in binary sentiment classification of a long sequence using LSTMs and GloVe vector representations of words. If the results of this thesis can reach or exceed 81.5% accuracy then we will consider the state of the art matched.

2.2.9 What are the hyper parameter concerns in non linear ML that will affect our experimentation? How do these decisions affect computational cost?

(Reimers & Gurevych, 2017) as well as (Zhang & Wallace, 2015) offer extensive discussion of hyperparameter engineering. The premise is that whilst non linear machine learning avoids the problem of hand crafted features, practitioners must now specify an exact model architecture that draws from very many variables. From (Zhang & Wallace, 2015):

"To the uninitiated, making such decisions can seem like something of a black art because there are many free parameters in the model. This is especially true when compared to, e.g., SVM and logistic regression. Furthermore, in practice exploring the space of possible configurations for this model is extremely expensive. (1) training these models is relatively slow, even using GPUs. (2) The space of possible model architectures and hyperparameter settings is vast."

Thus a major part of experimentation in the thesis will be concerned with hyperparameter engineering. The list of hyperparameters considered within (Reimers &

Gurevych, 2017) will be experimented upon, with other factors examined if there is sufficient time. The list of hyperparameters include: the type of word embeddings, the optimizer, the dropout, lstm hidden layers minibatch size.

2.3 Format II - Full Literature Review

Pang and Lee (2002) Thumbs up? Sentiment Classification using Machine Learning Techniques

In an early example of binary classification of a sequence, (Pang et al., 2002) apply a variety of linear machine learning techniques to an IMDb archive to classify a review as positive or negative. The sample is 2053 reviews of which 1301 are positive and 752 negative. This is a much smaller size than found in other papers but the results appear to justify the sample size. Feature selection is either a bag of words approach, bigrams (words taken as pairs) or a position marker. The support vector machine is found to be the most effective at 82.9% accuracy, with both unigrams and bigrams taken as features. Considerable discussion is made as to the problems capturing semantics in a sentence such as parts of speech and positioning. These results are less favourable than those found in other papers such as (Kim, 2014) or (Barry, 2017) and suggest for this thesis that linear ML is not the state of the art approach.

Mikolov and Zweig (2012) Context dependent recurrent neural network language model

(Mikolov & Zweig, 2012) offers an early application of vector representations to feed into a recurrent neural network. The objective is to predict the 51st word (given 50 predecessors) in the Penn Treebank portion of the Wall Street Journal corpus. This paper offers an initial perspective on applying vector representations instead of simple bag of words to the classifier. However LDA has been superseded by skip gram and CBOW methods, as is shown by the dimensionality of the vectors used (40 for LDA in this paper vs 300 used by (Mikolov et al., 2013). Further the problem is different as this thesis aims to classify sequences rather than predict terms in a sliding window.

Collobert et al 2011 Natural Language Processing (Almost) from scratch

(Collobert et al., 2011) aim to apply non linear machine learning methods to allow important elements in natural language processing to be found in an unsupervised way. The intuition is that non linear machine learning methods will allow the feature selection process to be avoided. The four tasks chosen are part of speech tagging, chunking, named entity recognition and semantic role labelling. The type of neural network uses a window approach and is a convolutional neural network in all but name. In part of speech tagging and chunking, the approach achieves close to state of the art performance compared with the linear ML approach. This paper suggests that the thesis should consider convolutions as a model architecture.

Hochreiter and Schmidhuber (1997) Long Short-Term Memory

(Hochreiter & Schmidhuber, 1997) gives the first description of the Long Short Term Memory (LSTM) architecture. In traditional RNN neural networks, recurrent back-propogation on an error in a sequence either blows up or tends to nothing, depending on the weights applied. By using a combination of memory cells and gate units, each step can be used to update the cell state which is preserved in a back-flow that runs up the entire chain. As a result, longer term dependencies in sequences can be preserved and ran at speed in a way which was previously not possible or highly expensive. This paper offers the main means by which the longer term dependencies found in our fiction/non fiction labelling dataset can be captured and trained against, an LSTM architecture.

Tang et al 2015 Document Modelling with Gated Recurrent Neural Network for Sentiment Classification

A combination of CNNs and LSTMs are used to learn sentence level representations that are then used with a recurrent neural network to create a document level representation which is used to provide a binary classification of sentiment classification. Longer term semantic dependancies within the document are supposed to be captured

by the LSTM approach and this is supported by the outperformance of the proposed approach vs other state of the art approaches. The method is tested against four sentiment classification datasets, the Yelp Challenge sets 2013-2015 and the IMDB set. A series of baseline comparisons are offered involving SVMs and preset features and further a majority assignment that simply assigns each classification as the majority. The LSTM approach is found to consistently yield the highest accuracy (F-measure) over linear machine learning and convolutional neural networks. This is highly relevant to the aims of this thesis for several reasons. First, the length of a document in each of the sets consists of a series of sentences, with the semantic connection between sentences a potential driver of the final classification. Further it offers several baseline comparisons that might be used to compare the accuracy of the model, assigning a majority classification might be used also in the dataset used by the thesis. Finally the type of classification is simple binary (positive or negative, fiction vs non fiction). However the paper considers a multitude of options that can be comfortably avoided in this thesis, for example the use of simple recurrent neural networks and svms. The fact that LSTM architectures consistently achieve the highest accuracy supports the intuition that LSTMs are the best approach for longer level semantic dependancies and document level classification.

Ilya Sutskever and Oriol Vinyals (2015) Recurrent Neural Network Regularization

Overfitting it found to be a problem with RNNS that cannot be solved in the same way as feed forward networks (dropout) This paper proposes a method to alleviate the problems of overfitting with a dropout that is applied to a single cell in the LSTM chain without affecting the weights of many time steps/cells in the past. The difference between test and validation set word-level perplexity is at its lowest when using dropout to regularise LSTMs in instances with a single LSTM and multiple LSTMs where an average classification is used. This paper looks at language modelling (predicting the next word given an input chain) and is therefore different to the approach of this thesis in which a document is assigned a binary classification. However overfit-

ting is still likely to be a problem given the high dimensionality of the dataset (vector representations of words). As a result the dropout method proposed in this paper will be used in hyper parameter engineering stage.

Zhou et al (2015) A C-LSTM Neural Network for Text Classification

(Zhou et al., 2015) propose a novel approach that combines the CNN and LSTM architectures to heighten Neural Network performance in Text Classification. They posit that the weaknesses of CNN (an inability to capture longer term dependencies that have semantic value in classifying a text) can be fixed by combining this with an LSTM. The LSTM is moreover less suited for learning local features such as N-grams as each step is taken sequentially. By feeding the output from a single layer CNN into the inputs of an LSTM, a C-LSTM architecture is formed. Experiments are conducted on the SST movie set (sets of sentences and sentiments attached) and classification is performed binary and with 6 classes. Binary in which degree of sentiment is removed and reduced to just positive/negative. The same set is used within (Kim, 2014), (Tai, Socher, & Manning, 2015). In binary classification, the proposed CLSTM model does achieve results comparable to the state of the art found in those papers (87.7% (Zhou et al., 2015), 88.1% (Kim, 2014)). In 6 class classification there is a clear increase (94.6% (Zhou et al., 2015), 92.2% (Kim, 2014)), suggesting that more nuanced shorter texts with more classes see an increase as a result of the CLSTM. In deciding the architecture that will best suit the classification task of this thesis, a combined approach of LSTM and CNN is an option. However given the debatable gain in accuracy in binary classification of sentiment analysis (which is closer to the binary classification in this thesis), the time will be better spent tuning hyperparameters such as vector representations and language markers with text cleaning etc. For this reason a combined approach will not be pursued within this thesis.

Reimers and Gureyvch (2017) Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks

(Reimers & Gurevych, 2017) set out to perform a similar task as (Zhang & Wallace, 2015) but in an LSTM architecture, which presents very different concerns and hyperparameters to a CNN architecture. In keeping with other papers on hyperparameter tuning, the emphasis is on the fact that tuning is often a black art requires unwritten rules of thumb or brute-force search to find optimal settings. A series of different tasks such as Part of Speech Tagging, Named Entity Recognition and chunking are performed and hyperparameters including the use of pretrained embeddings, the optimizer, the classifier, the use of a dropout layer, the mini-batch size and the number of LSTM layers are tested. Of the parameters tested, Word Embeddings, the Optimizer, the Classifier and Dropout were all found to have a high impact on the performance of the model; whilst LSTM units and Mini Batch size were found to have a medium impact on performance. Despite the fact that document classification is not one of the tasks this paper tests against, this paper is highly relevant to the thesis as it offers a contemporary view on hyperparameter tuning in an LSTM architecture. Hyperparameters will be tested against to see how their application affects the model output in predicting fiction or non fiction.

Kim (2014) Convolutional Neural Networks for Sentence Classification

Making use of a simple one layer CNN architecture and pre trained word vectors, (Kim, 2014) achieves a state of the art classification rate for sentence level classification across a range of datasets ranging from binary up to 6 classes (the same sets used in (Zhou et al., 2015)). Hyperparameters unique to CNNs such as windows and stride are tested, with dropout rate and mini batch held constant. Further, (Kim, 2014) makes use of pretrained vectors trained on 100 billion words from Google news (as described in (Mikolov et al., 2013)). State of the art F1 measures are achieved by some variant of the CNN architecture proposed by (Kim, 2014) in three datasets: the Stanford Sentiment Treebank (binary version), the Rotten Tomatoes Movie review set and the Hu/Liu customer review dataset. The use of pretrained vectors in this

paper dramatically increases performance and add to the well-established evidence that unsupervised pre-training of word vectors is important in deep learning NLP. This confirms the relevance of testing word vector strategies in optimising the model for this thesis.

Barry (2017) Sentiment Analysis of Online Reviews Using Bag-of-Words and LSTM Approaches

(Barry, 2017) aims to demonstrate the relevance of word order in sentiment analysis by comparing the accuracy of LSTM neural networks with bag-of-words and SVMs. In keeping with the intuition on LSTMs, the argument is that longer term dependencies such as negation or compound phrases cannot be captured when words are taken as individual entities. Furthermore vector representations taken from (Mikolov et al., 2013) and (Pennington et al., 2014), the Word2vec and GloVe word representations, are tested to measure the increase in classification accuracy. The classification task is basic binary sentiment classification at a document level (multiple sentences) and two datasets are considered; the Amazon Fine Foods review set and Yelp challenge dataset. The model with highest accuracy is found to be an LSTM with GloVe embeddings for the amazon dataset, and an LSTM with Word2Vec embeddings for the Yelp dataset (94.1 and 94.8 respectively). As expected the LSTMs consistently outperform the support vector machine and naive bayes with bag of word approaches. (Barry, 2017) performs a series of experiments that are highly relevant to this thesis as we are testing for a vary similar set of long term dependancies in text, so the same concerns as to vector representations and model construction apply. Further we can surmise from this that the neural network approach is the state of the art.

Denil et al., (2014) Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network

(Denil et al., 2014) aim to summarise review text by applying a paradigm from computer vision to document modelling. Namely that by taking convolutions at lower levels, document level semantics can be derived as they are made up of groups of sen-

tences which are themselves made up of groups of n-grams. Pooling of convolutional outputs is performed at a max pooling layer which is then used by the classifier. The most important of these sentences from pooling are then brought together to form a summary. The efficacy of these summaries is tested with a novel approach, a bag of words naive bayes classifier is trained against the sentiment of a summary generated in this way vs a summary generated by selecting sentences at random. Testing in this way, there is a clear increase in the accuracy of the bayes classifier (as much as 9% increase when only a 25% sample of the text is taken for summary). Whilst no direct comparison is made between this approach and an LSTM, intuition would suggest that a CNN that is trained on N-grams will be less effective at capturing very long dependancies in a text, as the N-gram will sample smaller phrases. Whether CNN or LSTM architecture is more effective will be part of the experimentation performed in this thesis.

Lund and Burgess (1996) Producing high-dimensional semantic spaces from lexical co-occurrence

Improving on the original construction of vector representations of words, (Lund & Burgess, 1996) offer an automated fashion by which words distances might be described by vectors. Where previously words are assigned a position on an axis by human auditors, the HAL method outlined by this paper produces a space based on co occurrence in the text. This type of method is the first example of an automated means by which words are represented by more than just a value in a dictionary. Although the work is too early to have a means of implementing using Python. Further work on this is found in Mikolov et al (2013 and Pennington et al (2014). Efficient Estimation of Word Representations in Vector Space

Mikolov et al (2013) Efficient Estimation of Word Representations in Vector Space

This paper outlines two novel means by which words might be represented in vector space with comparatively little computational expense to previous methods. The

continuous bag of words (CBOW) and continuous Skip Gram, which are in effect inversions of each other. CBOW involves predicting a word from a given context of words, and Skip Gram predicts a context given a word. These models form the basis of Word2Vec and GloVe vector learning methods that have become mainstream in natural language processing (Kim, 2014), (Zhou et al., 2015), (Barry, 2017), (Tang et al., 2015). Further this paper directly shows that the proposed methods are more effective than Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) which were previously the state of the art of learning word representations. Training time varies greatly depending on the architecture used, vector dimensionality and the size of the training dataset (between 0.3 and 2.5 days). However after training is complete a vector space with intuitive uses is made. The example given in the paper is that the cosine vector derived from combining (Man, King, Queen, x) will arrive at woman for x. The vectors trained on googles news page are made available alongside this paper. This is a landmark paper and gives the two vector learning methods that will be used in testing for this thesis. Further the pretrained vectors offered by (Mikolov et al., 2013) will be tested in this thesis.

Tang et al. (2015) Target-Dependent Sentiment Classification with Long Short Term Memory

(Tang et al., 2015) posit that a further problem for sentiment classification is the relatedness of a polarity word with the actual target of the text, i.e it might be possible to have a word that expresses great sentiment but does not refer to the actual target of the text. In order to address this a novel LSTM approach that takes a target from the middle of a sequence, building backwards and forwards to model the relatedness of that target to its context. Intuition suggests this approach might be suitable for this thesis, as it might be possible to refer to fictional notions within a non fictional review in an unrelated manner. However the implementation of this architecture is expensive as it involves two LSTM models running in parallel, with implications for run time and questionable return on the time invested. For this reason this architecture will not be tested against in this thesis.

Le and Mikolov (2014) Distributed Representations of Sentences and Documents

This paper offers more evidence as to why distributed vector representations of words are more effective than bag of words or bag of n-gram approaches. Also proposed is a paragraph vector algorithm that (much like CBOW from (Mikolov et al., 2013)) predicts a word given a context of words around it. This method expands on CBOW by sampling many contexts from the same paragraph (a wider window than looking at the words either side or in the same sentence used by CBOW). This paper reports the error rate rather than accuracy % and demonstrate that paragraph vector approach gives a lower error rate on a commonly used dataset (IMDB movies) used by (Zhang & Wallace, 2015) and (Zhou et al., 2015). Training specific vector representations for paragraphs is beyond the scope of this thesis as the error rate is only improved by 1.2% in the IMDB dataset vs word level vector representations. Further training vectors on the dataset adds time and complication vs using pretrained vectors available from GloVe and google Word2Vec.

Zhang and Wallace (2016) A Sensitivity Analysis of (and Practitioners Guide to) Convolutional Neural Networks for Sentence Classification

This paper aims to shed some light on the supposed black art of hyper-parameter engineering in sentence classification using CNNs. The paper investigates the effect of adjusting the input word vectors, filter region size, feature maps, max pooling strategy and dropout. The same text preprocessing is examined as in (Kim, 2014). The paper closes by offering specific advice around each of these hyperparameters to practitioners using CNNs for sentence classification. Two of the hyperparameters addressed by this paper (vector representations and dropout layers) are also used by LSTMs and the advice offered here will be implemented in the thesis i.e that experimentation is necessary on both and that higher dropout rates than 0.5 are generally found to be more effective. However the domain area is different as sentence classification is a shorter sequence than the review length found in the dataset of this thesis, lending more weight to the argument that CNNs are less likely to be effective. Should CNNs

prove an effective architecture, the other hyperparameters considered in this paper will be experimented with.

Komninos and Manandhar (2016) Dependency Based Embeddings for Sentence Classification Tasks

Expanding on the work of (Mikolov et al., 2013), the Skip Gram model by which vector representations of words are derived is expanded using dependency contexts. Where the original simply takes a window of 5 words and attempts to use the 2 either side to target the middle 3rd word, this dependency model uses context markers instead of the actual words (meaning stop words might be skipped). The results show that for some datasets this type of vector representation is more accurate and benefits linear ML classifiers greatly as it gives a source of structural information (that otherwise is only captured by non-linear CNNs or LSTMs. This type of dependency vector representation would require training specific word embeddings and parsing the corpus for each sentence to give a relative dependency, for this reason the method will not be used in the scope of this thesis. Results are compared across models using simple accuracy measures which gives a clear indication of the top performers.

Gal and Ghahramani (2016) A Theoretically Grounded Application of Dropout in Recurrent Neural Networks

The tendency of RNNs to overfit their training sets is a major problem in deep learning and this paper aims to arrive at rules that prevent this overfitting. Dropout has been used before (Zaremba et al., 2014) but this paper expands by adding a Bayesian theoretical element. By applying the same dropout throughout layers in the sequence rather than a random dropout at each layer, weights adjust to relying too heavily on specific sequence that drive the overfit. This paper provides further evidence that dropout will be very important to addressing any problem with overfitting that will be encountered in the thesis and will be an area for experimentation.

Ma and Hovy (2016) End-to-end Sequence Labelling via Bi-directional LSTM-CNNs-CRF

By combining word and character level representations, aim to improve sequence labelling that is derived from either levels working alone. CNNs are used to derive character level features and a bi-directional LSTM with word embedding vectors is used to perform two NLP tasks: named entity recognition and part of speech tagging. Absolute accuracies are reported in conjunction with the results from other papers. The character level model is shown to be effective and this paper again cites dropout and correct choice of vector space representation important to prevent overfit.

Pennington et al (2014) GloVe: Global Vectors for Word Representation

In this paper, word representations are improved by adding global co-occurence statistics to local window occurrences used by the Skip Gram model of (Mikolov et al., 2013). The result is the GloVe model of word representations and is tested against the same word analogy, similarity and named entitiy recognition sets used by (Mikolov et al., 2013) and has a greater accuracy at each training time compared with CBOW and Skip-Gram models. Vectors are trained against a corpus of Wikipedia entries from 2014 with a 400,000 word vocabulary. The availability of pretrained vectors that perform so highly with a state of the art performance is highly useful for the means of this thesis and will be used in experimentation alongside vectors derived by Word2Vec.

Maas et al (2011) Learning Word Vectors for Sentiment Analysis

Commenting on the effectiveness of word vector representations in specific applications, this paper suggests that a domain specific approach is needed in sentiment analysis to capture the subtleties of sentiment orientated texts. The model uses part supervised methods, attaching values between 0 and 1 alongside the context of a word to inform the vector space derived. The use of context specific vectors has been explored before and does bear some weight however such a method is unnecessary for the scope of this thesis as it would add greater time in labelling for the supervised methods to be

applied.

Abadi et al (2016) Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

This paper sets out the open source TensorFlow API designed for large-scale distributed deep machine learning. The system is scalable and can be operated by smaller domestic systems and larger distributed commercial ones, including certain domestic GPUs. Tensorflow works as a series of nodes connected by operations in what is called a graph. Each node performs some operation on the multi-dimensional arrays that are sent through it. These multi-dimensional arrays are known as tensors (thus Tensor flow). Further a companion visualisation tool called Tensorboard that is run from a browser allows for training to be viewed and the graph of the session to be viewed and thereby understood better. As a freely available machine learning platform that can support both linear and non-linear machine learning, Tensorflow will be used as the tool for development, training and testing of the models for this thesis.

Chetlur et al (2014) CuDNN: Efficient Primitives for Deep Learning

A library that makes use of parallel architectures and provides routines for GPUs. The thesis will make use of a domestic Nvidia GPU unit with CUDA options available, use of this library will speed up training and deployment of Tensorflow especially in expensive tasks such as LSTMs.

Chilimbi et al (2014) Project Adam: Building an Efficient and Scalable Deep Learning Training System

Traditional machine learning optimisers do not scale well as they are not designed to be parallelised. Specifically built distributed hardware has been used in the past but this is not a cost-effective option. This paper offers a new optimiser that parallelises operations, giving asynchronous parameter updates across its network that speeds up training time and accuracy. As mentioned in (Reimers & Gurevych, 2017), the type of optimiser used in the non linear architecture has a high impact on training and

performance. Both this paper and (Reimers & Gurevych, 2017) suggest that Adam will have the strongest performance, but the context of this thesis means this will need experimentation to confirm.

Haddi et al (2013) The Role of Text Pre-processing in Sentiment Analysis

(Haddi et al., 2013) aim to experiment on the types of data fed into machine learning classifiers by holding the classification technique constant and adjusting the types of pre-processing applied to data. Techniques experimented on include cleaning text of non utf characters, stemming, stop word removal and expanding abbreviations. Across all feature selection methods, accuracy is improved by applying some kind of text preprocessing. Whilst text preprocessing is less of a deep learning paradigm (in which the depth of the architecture is trusted to find features), there may be a gain in accuracy by streamlining the feature space in which the neural net is searching for features. Many of the papers considered such as (Kim, 2014) in which many state of the art performances are made, include text preprocessing. As a result text preprocessing will be experimented within the thesis.

Clark (2003) Pre-processing very noisy text

Publicly created content often has many semantic and syntactic errors, such as spelling, grammar and incorrect use of terms. (Clark, 2003) argues that these have a considerable impact on any sequence model as terms and features that are in essence similar may be mistakenly taken as different. Techniques such as stop word removal and lemmatisation decrease the degree difference these errors make by removing the dimensionality in which these errors are made. Further alterations such as spelling error correction and similar term compression (segmentation) might also be used. For example 10 different terms for Ha are found in the Penn treebank set that (Clark, 2003) argues might be comfortably collapsed into a single. In keeping with (Haddi et al., 2013), the preprocessing applied to the text will be an area of experimentation in this thesis.

2.4 Chapter Summary

In this chapter we have presented relevant literature that explains the technologies used and places the work of the thesis in its wider research context. The following chapter explains the method by which models were built as well as listing the experiments and giving the specification of the dataset. Again if a condensed view is desired, it is recommended to read experimental practice, model process and dataset before skipping ahead to the analysis chapter.

Chapter 3

DESIGN & METHODOLOGY

The study will consist of a series of models constructed in Tensorflow. The models will have varying parameters and by judging evaluative metrics it will be possible to deduce which parameter settings give optimal performance. The two metrics by which a model will be judged are absolute accuracy (% of examples correctly predicted) and time to perform 50 iterations. Where parameter categories are a sliding scale, a baseline will be compared against two other settings (High, Mid and Low), examples of this include the minibatch size or number lstm hidden layers. Where parameters are discrete (on or off) including the parameter will be compared to a baseline, examples of this include lemmatising text preprocessing vs non lemmatising.

3.1 Experimental Practice

Models in tensorflow undergo training iterations, with each step feeding in a batch of examples and updating weight rules accordingly. Consequently models can be judged at checkpoints during their training.

- Every 50 steps - % Accuracy against the training set and the time to perform 50 steps recorded
- Every 1000 Steps - Model saved for testing against a 10% test set

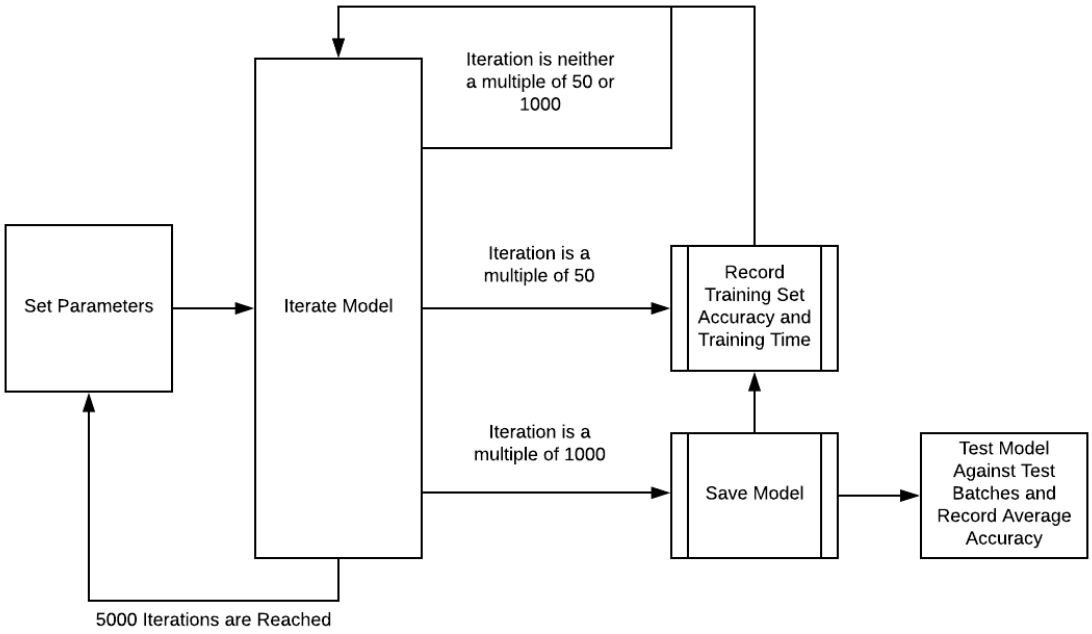

Figure 3.1: Experimental practice, depending on the iteration number different actions are taken

- Every model will run for 5000 steps - giving 100 evaluative metric recordings and 5 models

Each model is designed to report accuracy for a given test batch, therefore average accuracy across batches will be reported (sixteen batches at 16 size, eight batches at 32 size, four at 64 size). For more details on batching see section 3.3. After experimentation a final model with optimal parameters will be trained.

3.2 Model Process

The model follows 12 steps in which data is imported, pre-processed, batched and sent into TensorFlow where the model itself is induced through iterations of batches sent through a neural network.

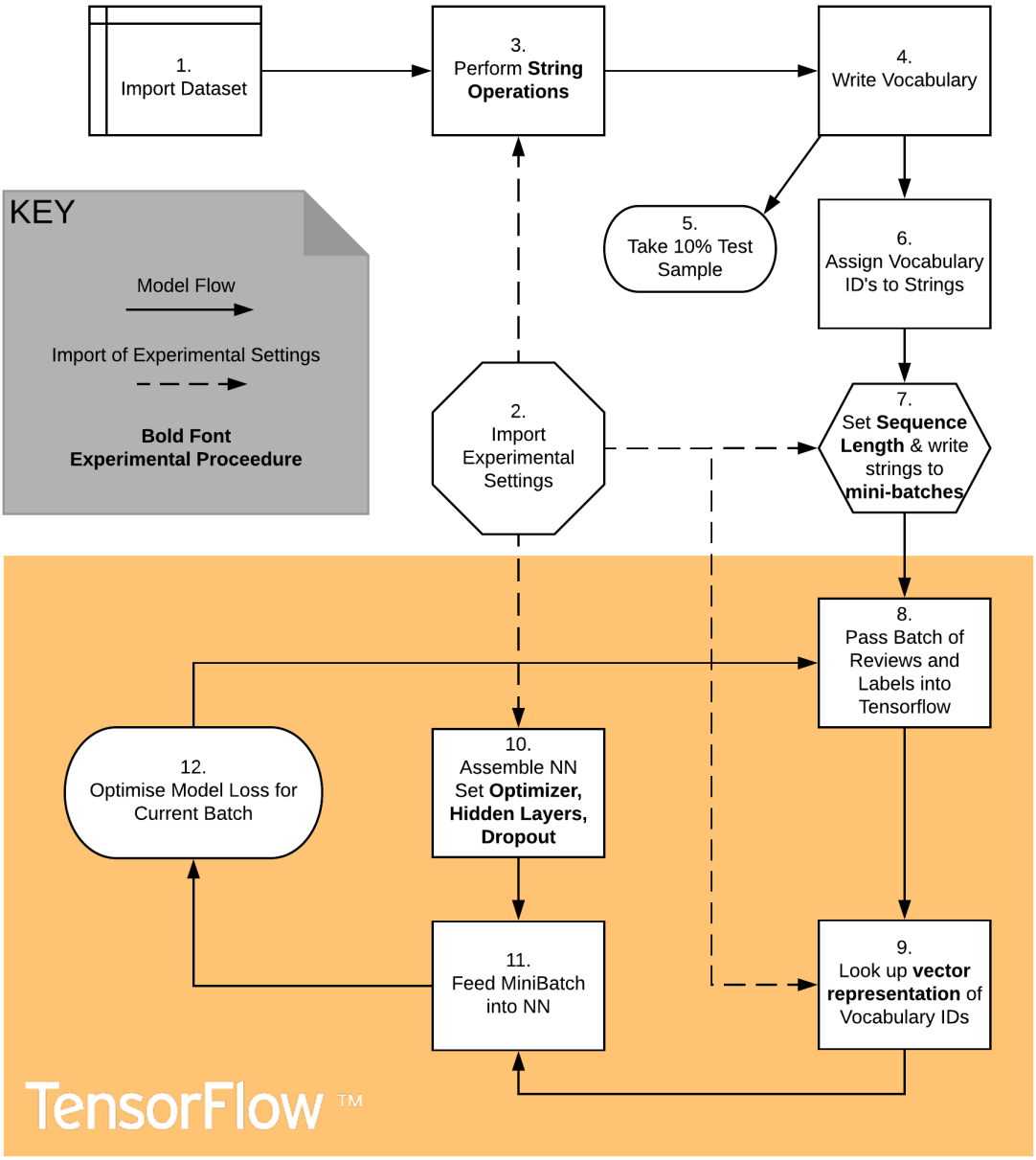

Figure 3.2: Model Process, for description see section 3.2

1. Import of the dataset in CSV
2. Import of experimental settings from a JSON object.
3. String operations are the first experimental parameter imported from JSON and performed on the dataset
4. The vocabulary of the resultant set is determined
5. A 10% Test sample of string is held aside
6. Strings are rewritten as a series of vocabulary indices
7. Sequence is padded (truncated) and sequences as formed into Mini Batches

Entering TensorFlow Environmet

8. A Mini Batch is passed into TensorFlow
9. Vector Representations are looked up for each ID in the given sequence
10. The Neural Network is constructed according to parameters taken from the settings JSON (Optimiser, Hidden Layers Dropout)
11. The batch is passed into the Neural Network
12. The Neural Network optimises its loss function along with the new information from the batch Steps 8,9,11,12 repeat until 5000 iterations are reached.

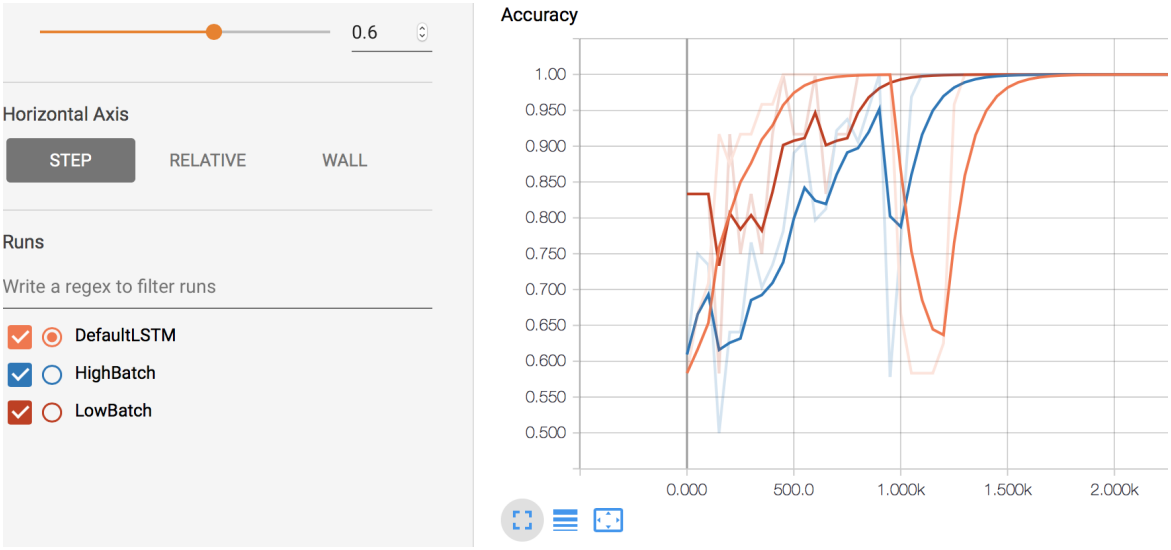

Figure 3.3: Example Tensorboard, showing training set accuracy as the model iterates for the Mini Batch size experiment

3.3 Example Results

Tensorboard is a browser based tool for use in tandem with TensorFlow that allows a user to visualize the learning process for the model and any metrics of interest. This will be used to display model training against the train set. See figure 3.3 as example. **Bar plots** will be used to show accuracy against the test set at each of the 5 training checkpoints. See figure 3.4 for example. **Violin plots** will be used to the distribution of time taken for 50 steps to be performed between models. See figure 3.5 for example. **Tables** will be used to show figures behind the plots. *Scipy.stats.describe* used for speed result statistics.

Spec	Min	Max	Mean	Variance
HighBatch	30.2	33.7	30.4	0.2
DefaultLSTM	28.2	33.4	28.5	0.41
LowBatch	27.6	31.3	27.8	0.2

Table 3.1: Example Table showing Batch Size Speeds (Seconds) in Performing 50 Steps

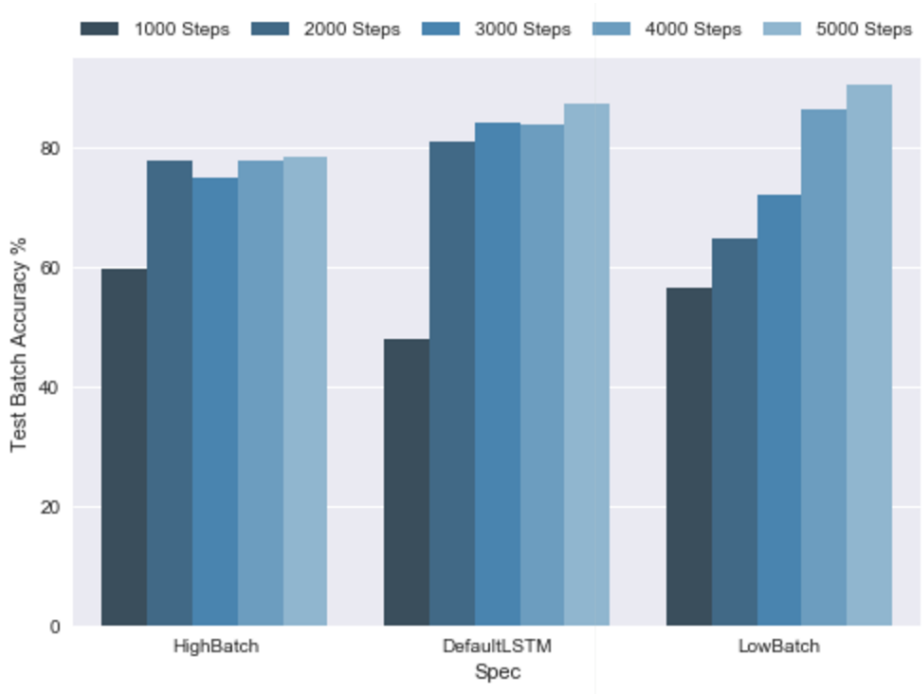

Figure 3.4: Example Bar Plot, showing test set accuracy at each checkpoint for the Mini Batch size experiment

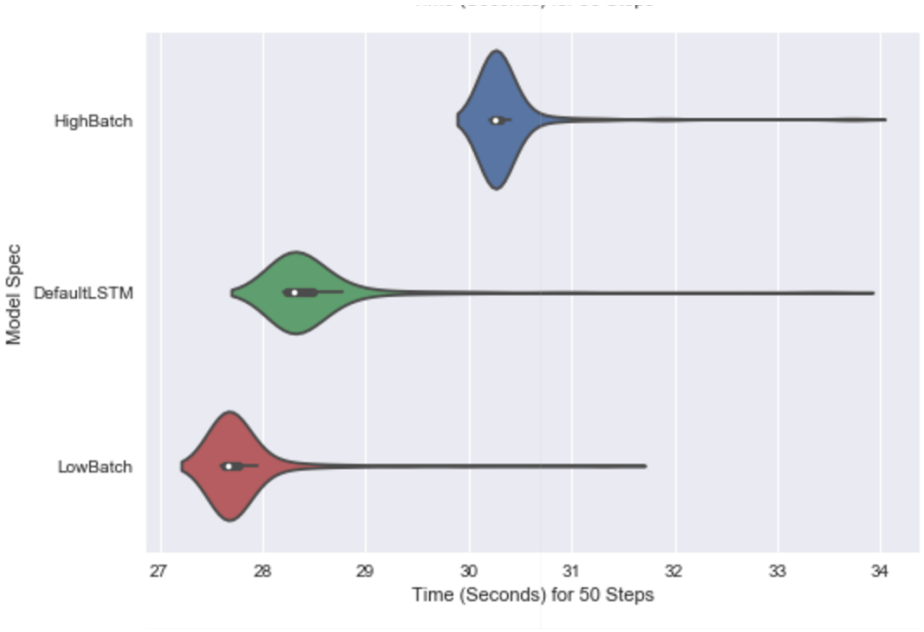

Figure 3.5: Example Violin Plot, showing the distribution of time taken for 50 steps to be performed for the Mini Batch size experiment

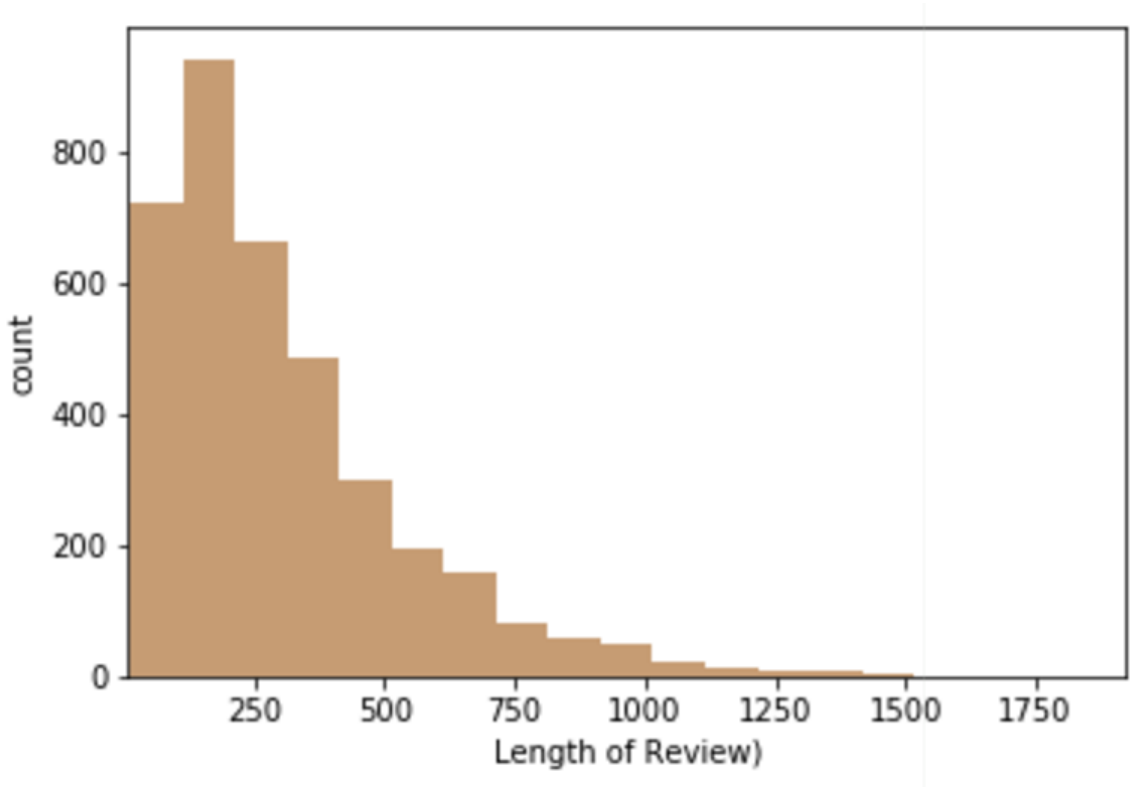

Figure 3.6: Histogram of Review Lengths, with a clear positively skewed distribution
Quartiles: 145, 267, 446
Mean Length: 306

3.4 The Dataset

The dataset consists of 3699 review texts and a boolean fiction label (true indicates fiction) 3.6. The data are manually labelled by a team of 3 auditors, with a sample of 200 overlapped for agreement metrics. The kappa value of inter agreement is used for which this labelling process had a value of 0.88, (Ganu, Elhadad, & Marian, 2009) cite a kappa value of 0.8 as very good agreement. This sample size is 1646 larger than that used by (Pang et al., 2002) and 76 fewer than the size used by (Kim, 2014). The split is 1967 non fiction and 1723 fiction reviews (47:53), this ratio is more favourable that that used by (Pang et al., 2002) in which 752 negative and 1301 positive reviews (37:63) are used. Further the lengths of reviews follow a clear sloped distribution without local maxima, which gives us confidence that the sample size is adequate.

3.5 Experiment List

Experiments are grouped by category and consist of the full parameter specification and commentary on the value of the experiment in answering the research question.

3.5.1 Architecture Experiment

Spec	Default LSTM	CNN
Sequence Length	300	300
Batch Size	24	24
Hidden Units	32	32
Dropout	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam
String Operations	None	None

Table 3.2: Architecture Experiment Parameters

CNN Specific Parameters - Window : 10, Stride : 5

As described in chapters 1 and 2, the architecture used to induce the model will be the subject of experimentation. The intuition behind this experiment is that certain forms of text from which a fiction or non fiction label can be seen, might be captured in a series of N-grams (the CNN method). At the same time others forms of text require an LSTM approach to be captured. By experimenting with architectures we can deduce which forms of text are more prevalent and important in review texts. For more information on architectures, see (Hochreiter & Schmidhuber, 1997) (LSTMs) and (Denil et al., 2014) (CNNs) in the literature review. The question that can be answered by this experiment is: is the more computationally expensive LSTM approach necessary?

3.5.2 Optimizer Experiment

Spec	Default LSTM	SGD Optimizer
Sequence Length	300	300
Batch Size	24	24
Hidden Units	32	32
Dropout	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300
Optimizer	**Adam**	**SGD**
String Operations	None	None

Table 3.3: Optimizer Experiment Parameters

The optimizer is a part of the architecture that gives the procedure by which weights are adjusted to new information. The choice of optimizer is found by (Reimers & Gurevych, 2017) to be highly important in binary sentiment analysis. For more information on the Adam optimizer see (Chilimbi, Suzue, Apacible, & Kalyanaraman, 2014) in the literature review. The question that can be answered by this experiment is: is the choice of optimizer as important in this problem vs binary sentiment analysis?

3.5.3 Hidden Units Experiment

Hidden units serve as the number of weights stored by the network as it unrolls a sequence. Higher numbers mean a more fine grained recording is made of the weights found by the network, but this in turn might increase the tendency to overfit. The question that can be answered by this experiment is: do book reviews need as fine a grain representation as sentiment analysis to capture the fiction or non fiction element or does this cause the model overfit? Is this as important as found in binary sentiment analysis?

Spec	High Hidden Units	Default LSTM	Low Hidden Units
Sequence Length	300	300	300
Batch Size	24	24	24
Hidden Units	**64**	**32**	**16**
Dropout	0.5	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam	Adam
String Operations	None	None	None

Table 3.4: Hidden Units Experiment Parameters

Spec	Long Sequence	Default LSTM	Short Sequence
Sequence Length	**350**	**300**	**250**
Batch Size	24	24	24
Hidden Units	32	32	32
Dropout	0.5	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam	Adam
String Operations	None	None	None

Table 3.5: Sequence Length Experiment Parameters

3.5.4 Sequence Length Experiment

Neural networks require inputs to be of fixed dimensions to allow the same operations to be performed on them. As a result, the length of a sequence is set at a maximum and strings are either padded or truncated to reach this length. Intuition suggests that longer sequences will increase computation time, but truncating longer sequences may remove a crucial part on which the correct labelling might hinge. The question that can be answered by this experiment is: can we afford to truncate sequences for the uplift in computational speed, or does this cost too greatly in accuracy?

3.5.5 Batch Size Experiment

Spec	High Batch Size	Default LSTM	Low Batch Size
Sequence Length	300	300	300
Batch Size	**64**	**24**	**12**
Hidden Units	64	32	16
Dropout	0.5	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam	Adam
String Operations	None	None	None

Table 3.6: Batch Size Experiment Parameters

Mini Batch size denotes the number of examples used by the network to establish weights in a single iteration. Larger batch sizes make special cases in a batch less highly weighted but do mean that the model is exposed to more rules more quickly. (Reimers & Gurevych, 2017) find batches between 1-32 in size to be optimal. The question that can be answered by this experiment is: do book reviews have similar subtleties to sentiment analysis that require the model to be trained slowly (many smaller batches rather than fewer large batches) to capture? Is this as important as found with binary sentiment analysis?

3.5.6 Dropout Wrapper Experiment

As propounded by (Zaremba et al., 2014), the correct use of a dropout wrapper helps to alleviate many problems to do with overfitting that arise from NNs especially LSTMs. The question that can be answered by this experiment is: is overfitting a large concern in this problem and does a large drop out need to be used to preserve accuracy? Is this as important as found in binary sentiment analysis?

Spec	High Dropout	Default LSTM	Low Dropout
Sequence Length	300	300	300
Batch Size	24	24	24
Hidden Units	32	32	32
Dropout	**0.75**	**0.5**	**0.25**
Vector Representation	Word2Vec300	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam	Adam
String Operations	None	None	None

Table 3.7: Dropout Wrapper Experiment Parameters

Spec	Glove300d	Default LSTM	Glove50d
Sequence Length	300	300	300
Batch Size	24	24	24
Hidden Units	32	32	32
Dropout	0.5	0.5	0.5
Vector Representation	**Glove300d**	**Word2Vec300**	**Glove50d**
Optimizer	Adam	Adam	Adam
String Operations	None	None	None

Table 3.8: Vector Representation Experiment Parameters

3.5.7 Vector Representation Experiment

Vector representations offer a way of giving context rather than a bag of words approach. However there is choice within the type of vector representation used. Here three different representations are considered. A 50 dimensional Global Vectors (Pennington et al., 2014) representation, a 300 dimensional Word2vec (Mikolov et al., 2013) representation and finally a 300 dimensional Global Vectors representation. Intuition suggests that these will be in increasing computational cost: the 50d option being the cheapest and the 300 GloVe the most expensive (Word2Vec stores only win-

dow contexts wheras GloVe takes from elsewhere in the paragraph). The question that can be answered by this experiment is: can we afford to use a lower dimension representation for the uplift in computational speed, or does this cost too greatly in accuracy? Is the choice of vector represntation as important here as found in binary sentiment analysis?

3.5.8 String Operations Experiment

Spec	Remove Stop Words	Lemmatize	ProperNoun
Sequence Length	300	300	300
Batch Size	24	24	24
Hidden Units	32	32	32
Dropout	0.5	0.5	0.5
Vector Representation	Word2Vec300	Word2Vec300	Word2Vec300
Optimizer	Adam	Adam	Adam
String Operations	**RemStop**	**Lemmas**	**ProperNoun**

Table 3.9: String Operations Experiment Parameters

In keeping with (Clark, 2003) and (Haddi et al., 2013), text preprocessing methods are experimented with. Removing stop words may lead to efficiency gains as sequences are made shorter and potentially unnecessary terms are lost. However one intuition suggests that longer reviews that are more verbose and use more stop words may be more technical in nature and therefore be more likely to be non fiction. Replacing terms with their lemmas will shorten the dictionary of terms as similar terms are compressed to a root term. This lessens the dimensionality of the feature space and may improve model fit and may improve training speed. An intuition found from the original examination of the dataset was that names of authors, places and characters often occur in reviews. As these names are unique to each book they will extend the dictionary of terms significantly without giving great indication as to whether the review is fiction or non fiction. By replacing these names with a ProperNoun marker, it

should be possible to reduce dimensionality that may improve fit and increase training speed. The question that can be answered by this experiment is: which string operations improve the fit and function of the model in this case?

3.6 Hardware. Coding and Platform

Models will be built using Tensorflow for python version 3.6 with GPU support. The hardware platform will be a domestic GeForce GTX 1050 Ti card with Nvidia cuda settings, with 4gb of VRAM. Initial models run on CPU took 220 seconds to run 100 steps of the default LSTM architecture. The GPU supported platform sped this up to 120 seconds. Code is written in python with settings parsed from a JSON dictionary object.

3.7 Chapter Summary

In this chapter we have presented the full methodology by which models are built, the variables for experimentation and the dataset specification. The following chapter details the full results of each experiment. Discussion and analysis of results are offered in the analysis chapter.

Chapter 4

IMPLEMENTATION & RESULTS

Results are presented in two formats. First as tables and secondly as graphics. The first set of tables displayed correspond to the blue bar charts e.g. table 4.18 & 4.26. The second set of tables displayed correspond to the violin plots e.g. table 4.10 & 4.3. The tensorboards do not have accompanying tables.

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
CNN	72.1	73.5	73.6	74.8	74.8

Table 4.1: Architecture Experiment Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
SGD	51.2	48.5	50.6	49.8	49.0

Table 4.2: Optimizer Experiment Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
HighLSTMUnits	73.4	78.3	76.8	66.3	82.375
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
LowLSTMUnits	59.6	72.9	70.1	69.9	69.5

Table 4.3: Hidden Units Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
SequenceLong	72.6	82.4	83.8	85.4	85.8
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
SequenceShort	49.5	49.5	78.8	78.4	78.4

Table 4.4: Sequence Length Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
HighBatch	59.6	77.9	75.1	77.8	78.5
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
LowBatch	56.6	64.8	72.0	86.5	90.5

Table 4.5: Batch Size Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
HighDropout	60.9	81.0	80.2	86.0	89.3
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
LowDropout	49.4	48.0	51.6	49.1	52.2

Table 4.6: Dropout Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
Glove 50d	49.0	72.0	46.4	51.4	48.7
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
Glove 300d	77.5	86.0	85.5	88.7	91.1

Table 4.7: Vector Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
Remstop	69.2	76.6	82.4	86.3	85.0
lemmatize	64.8	80.6	79.4	81.3	82.0
Propernoun	69.2	80.6	88.4	86.3	88.0

Table 4.8: String Ops Test Accuracy %

Spec	1000 Steps	2000 Steps	3000 Steps	4000 Steps	5000 Steps
DefaultLSTM	47.9	81.0	84.2	84.0	87.3
FinalLSTM	72.7	86.0	88.5	87.7	89.0

Table 4.9: Final Settings Test Accuracy %

Spec	Min	Max	Mean	Variance
DefaultLSTM	28.2	33.4	28.5	0.4
CNN	12.6	16.7	13.0	0.2

Table 4.10: Architecture Experiment 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
DefaultLSTM	28.2	33.4	28.5	0.41
SGDOptimizerLSTM	28.1	31.3	28.2	0.1

Table 4.11: Optimizer Experiment 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
HighLSTMUnits	30.1	33.1	30.2	0.1
DefaultLSTM	28.2	33.4	28.5	0.41
LowLSTMUnits	27.3	30.5	27.4	0.1

Table 4.12: Hidden Units 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
SequenceLong	32.9	36.6	33.0	0.2
DefaultLSTM	28.2	33.4	28.5	0.41
SequenceShort	11.4	20.1	11.8	1.0

Table 4.13: Sequence Length 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
HighBatch	30.2	33.7	30.4	0.2
DefaultLSTM	28.2	33.4	28.5	0.41
LowBatch	27.6	31.3	27.8	0.2

Table 4.14: Batch Size 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
HighDropout	28.1	32.4	28.3	0.3
DefaultLSTM	28.2	33.4	28.5	0.4
LowDropout	28.2	32.6	28.4	0.3

Table 4.15: Dropout 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
Glove 50d	24.9	28.6	25.3	0.2
DefaultLSTM	28.2	33.4	28.5	0.41
Glove 300d	26.7	30.6	26.9	0.2

Table 4.16: Vector Test 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
DefaultLSTM	28.2	33.4	28.5	0.41
Remstop	26.7	29.9	26.9	0.1
lemmatize	26.8	29.7	26.8	0.1
Propernoun	27.7	30.9	27.9	0.1

Table 4.17: String Ops 50 Step Speeds Seconds

Spec	Min	Max	Mean	Variance
DefaultLSTM	28.2	33.4	28.5	0.41
FinalLSTM	14.1	20.1	14.7	0.46

Table 4.18: Final Settings 50 Step Speeds Seconds

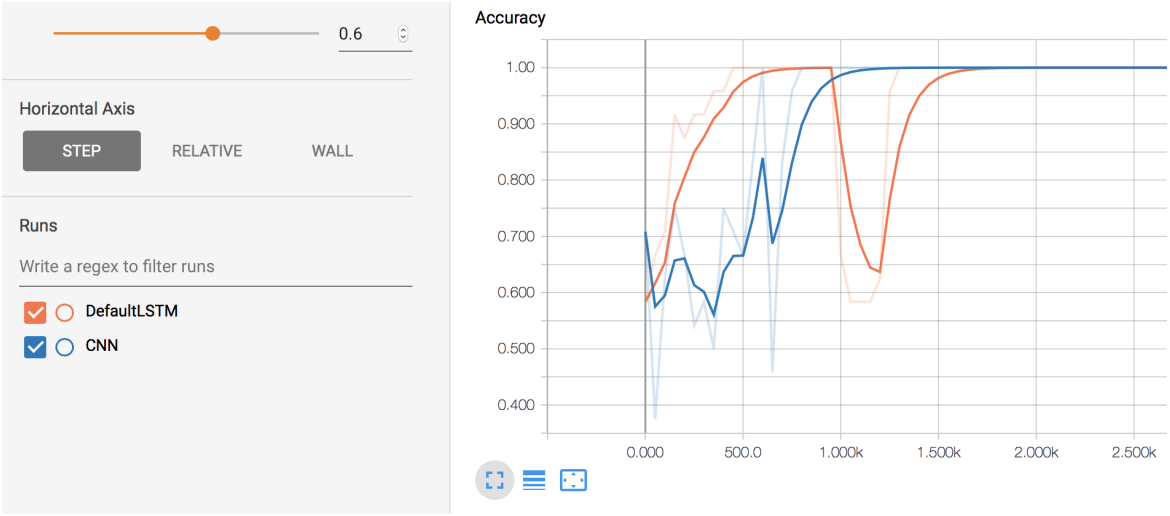

Figure 4.1: Architecture Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

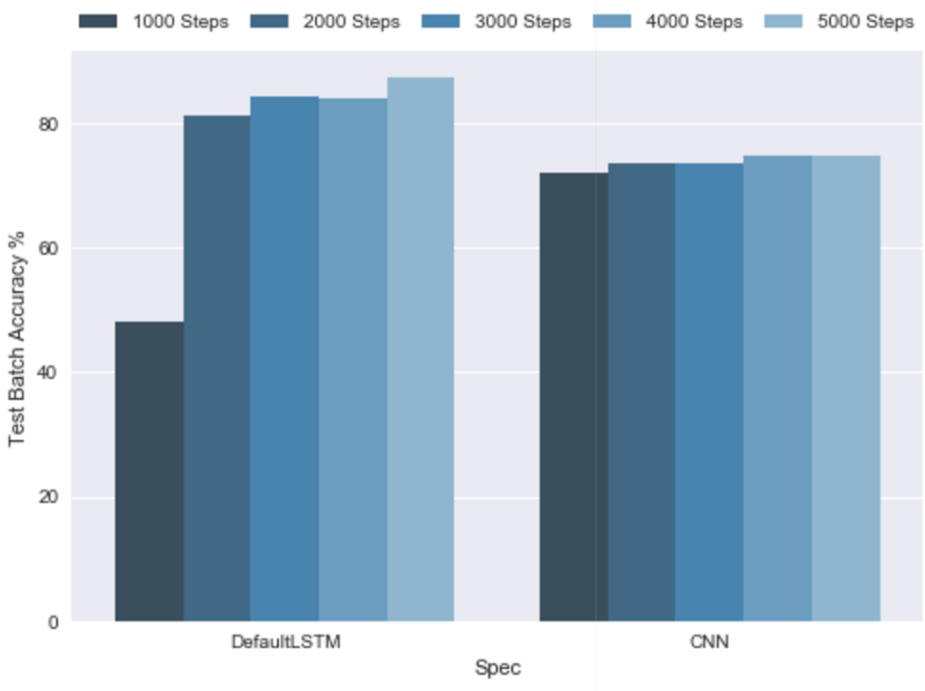

Figure 4.2: Architecture Size Experiment Test Accuracy

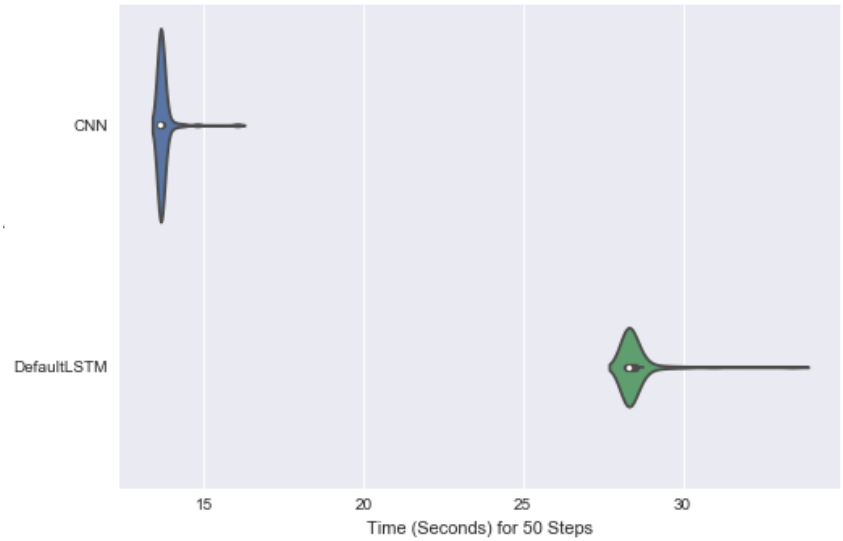

Figure 4.3: Architecture Size Experiment Violin Plot

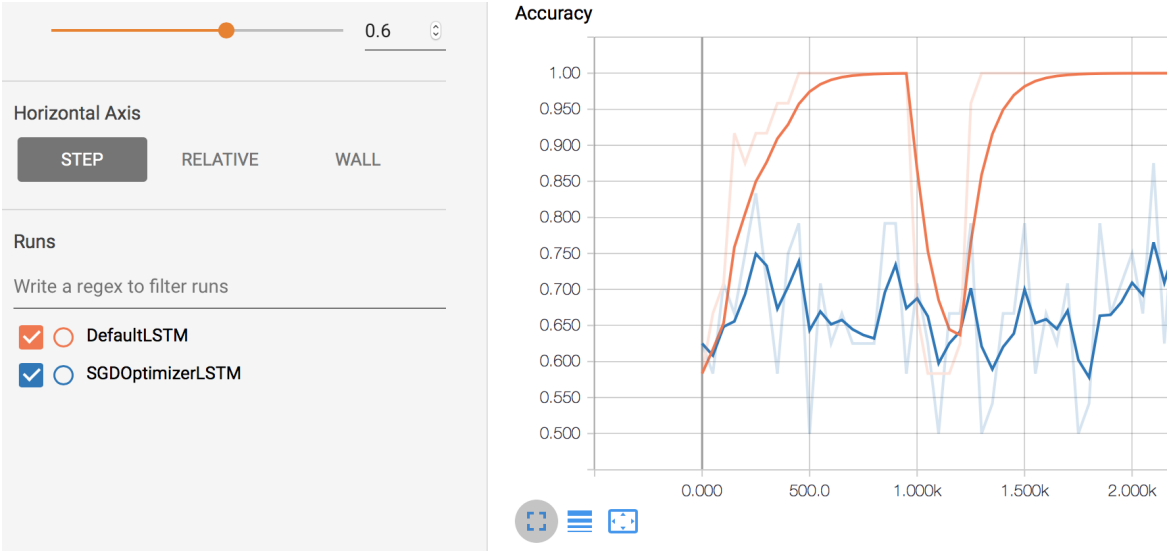

Figure 4.4: Optimizer Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

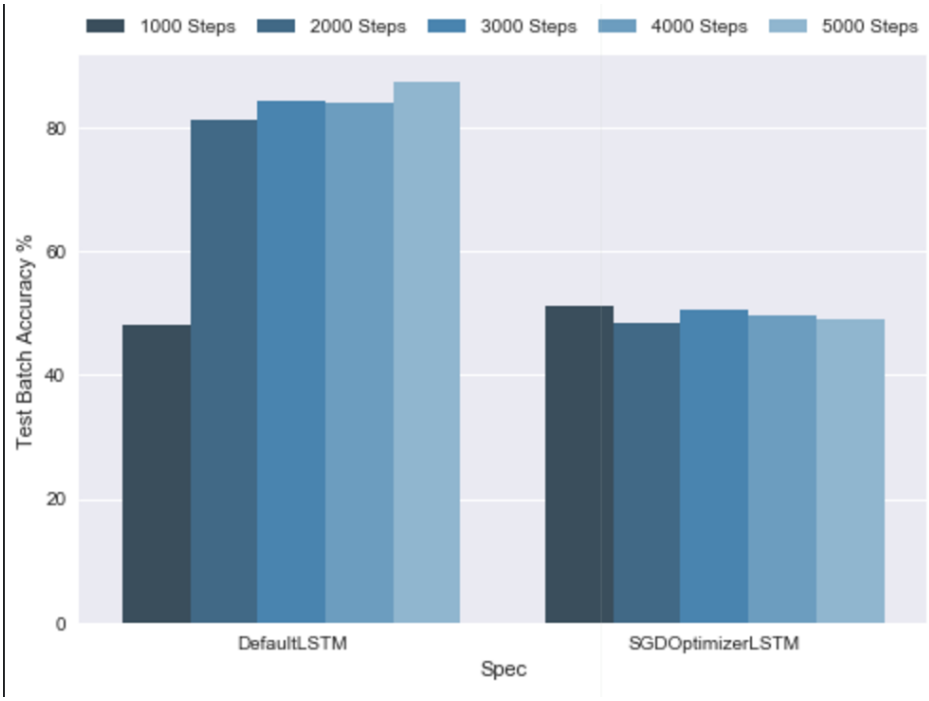

Figure 4.5: Optimizer Experiment Test Accuracy

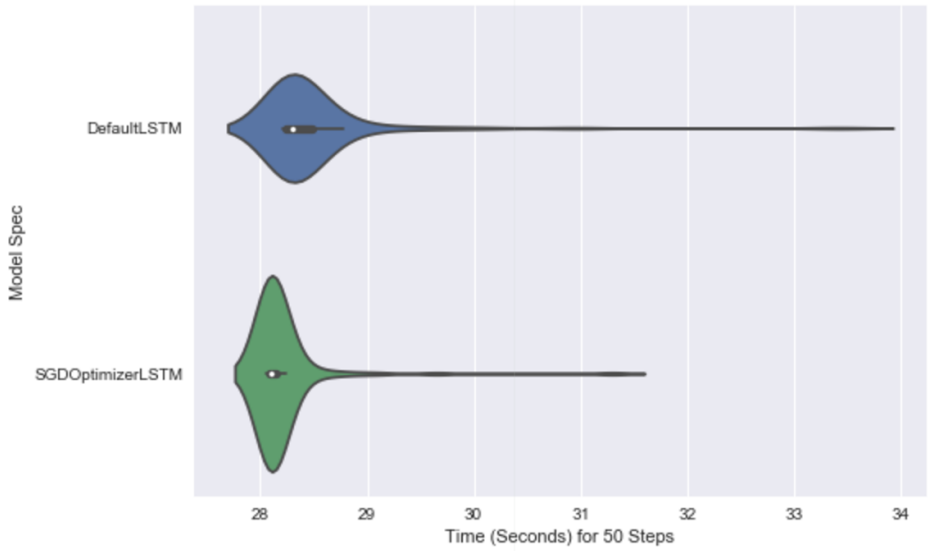

Figure 4.6: Optimizer Experiment Violin Plot

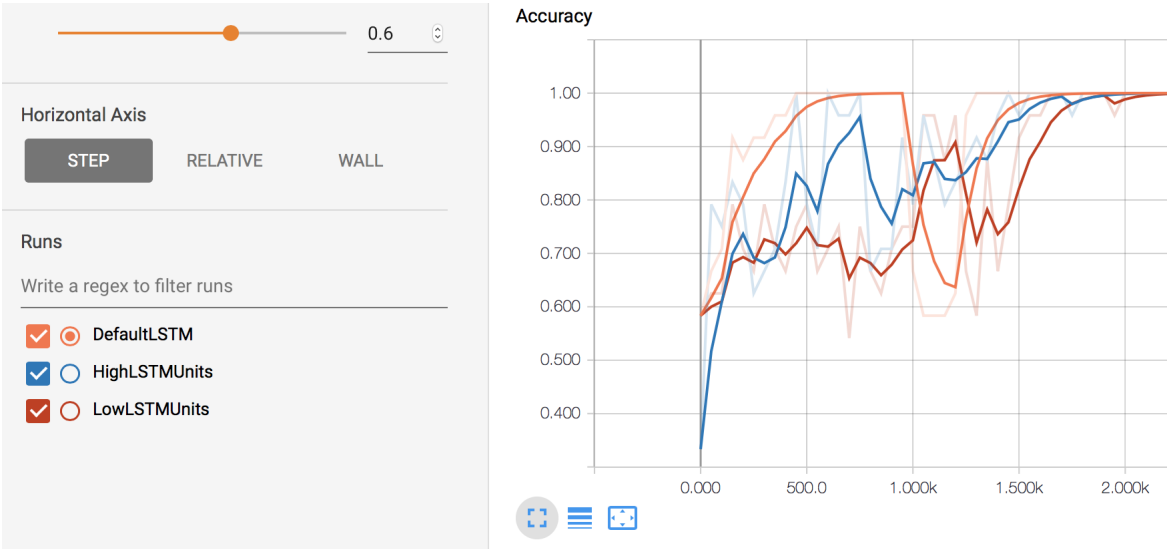

Figure 4.7: Hidden Units Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

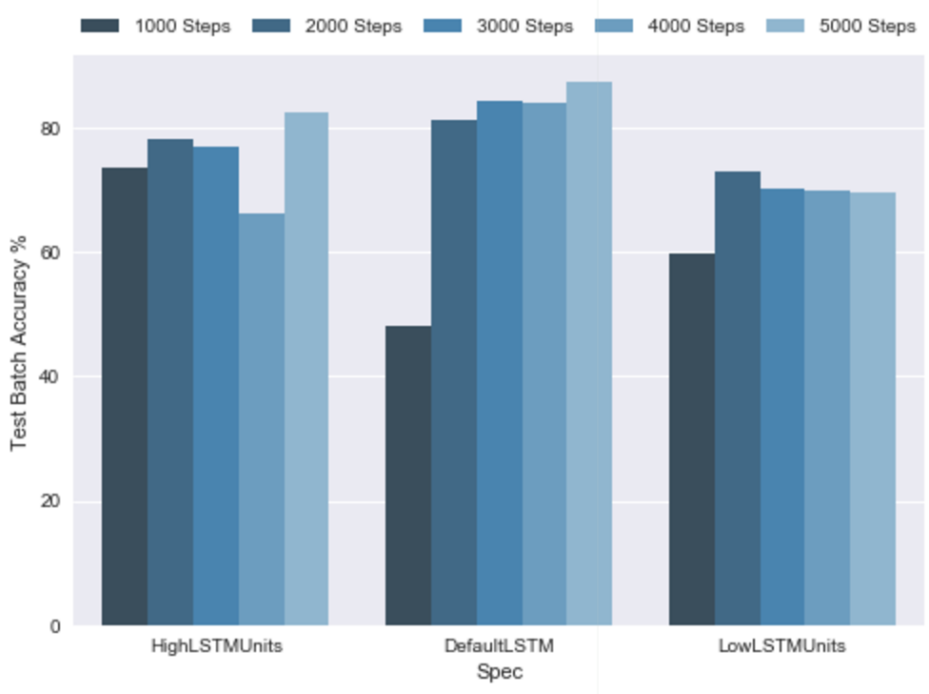

Figure 4.8: Hidden Units Experiment Test Accuracy

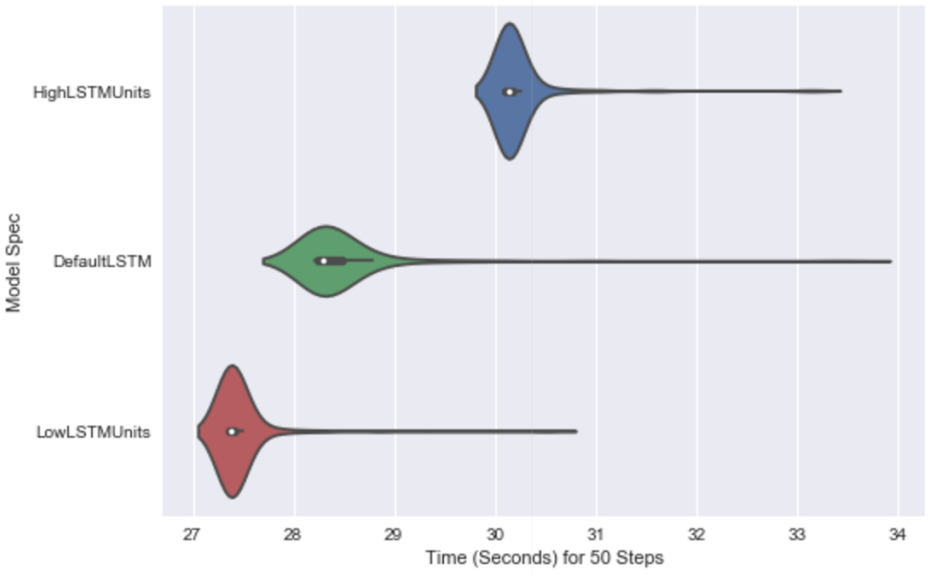

Figure 4.9: Hidden Units Experiment Violin Plot

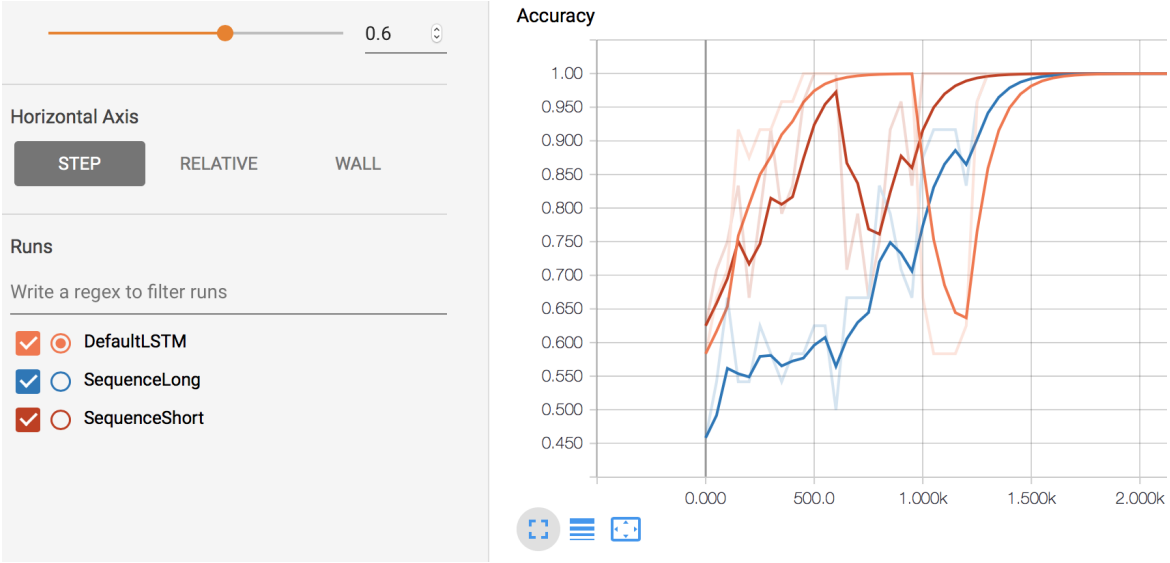

Figure 4.10: Sequence Length Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

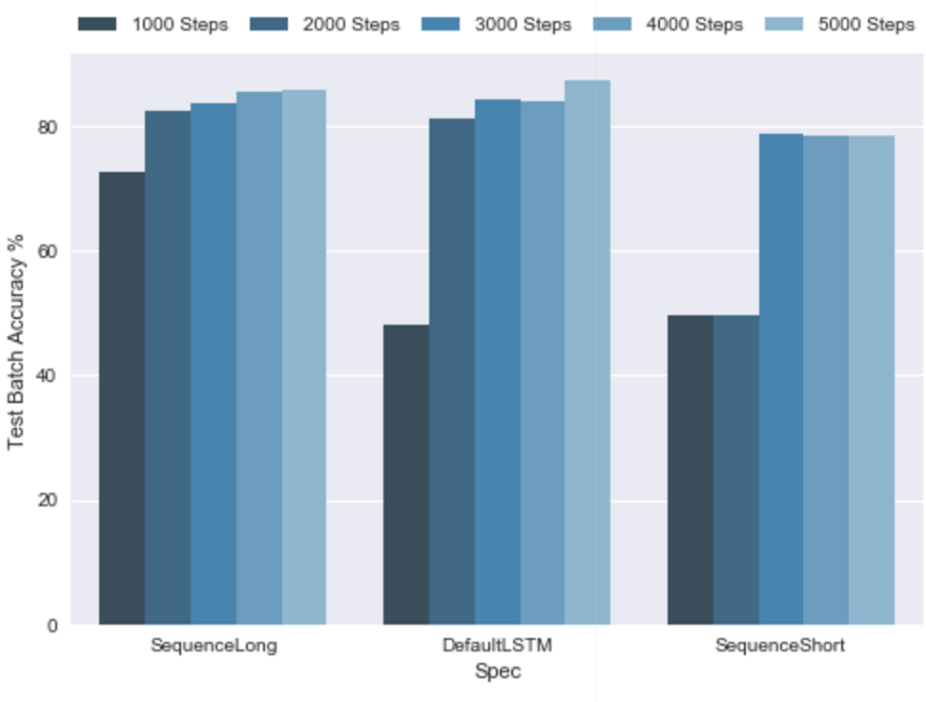

Figure 4.11: Sequence Length Experiment Test Accuracy

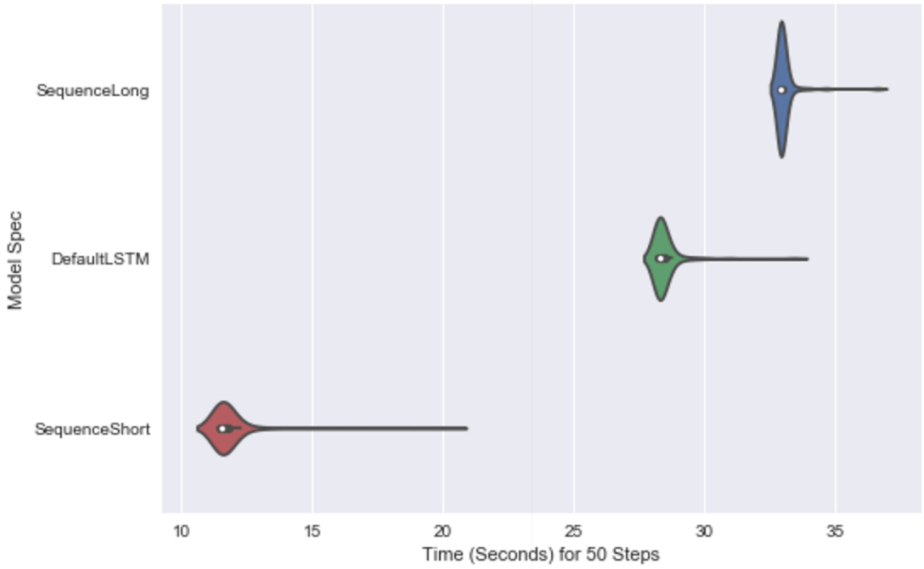

Figure 4.12: Sequence Length Experiment Violin Plot

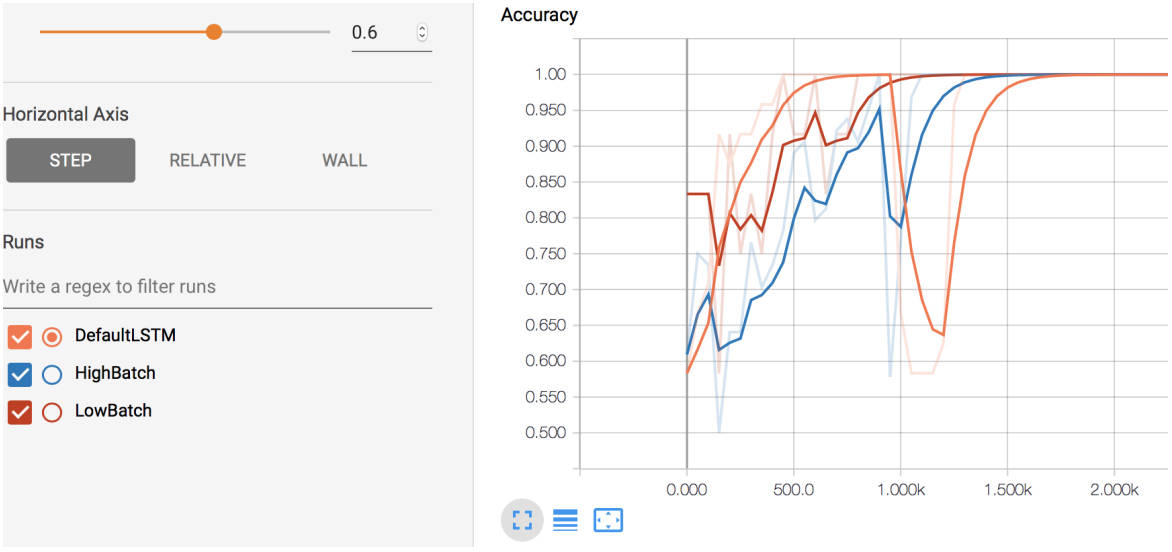

Figure 4.13: Batch Size Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

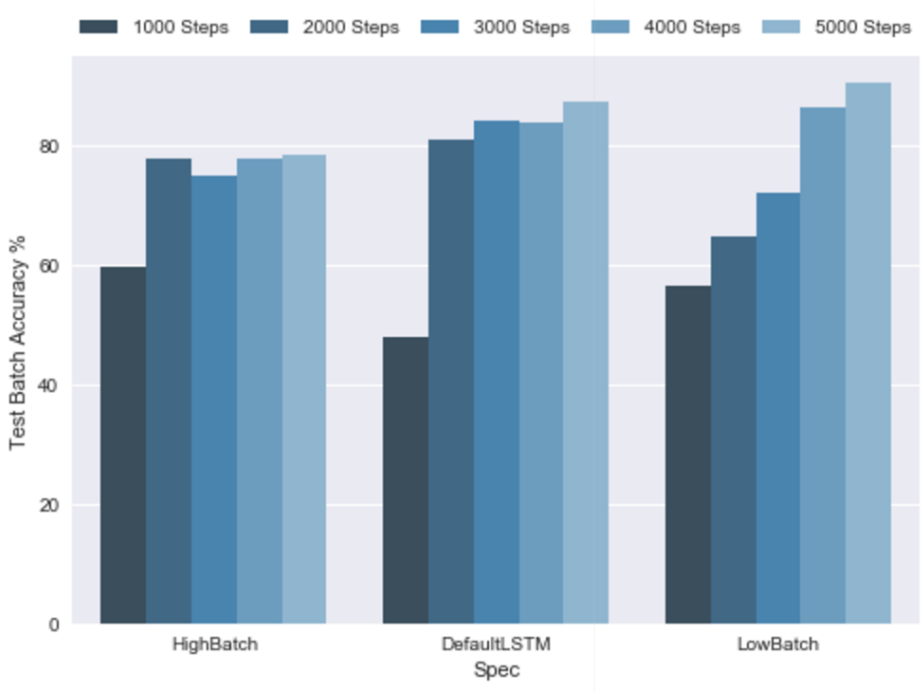

Figure 4.14: Batch Size Experiment Test Accuracy

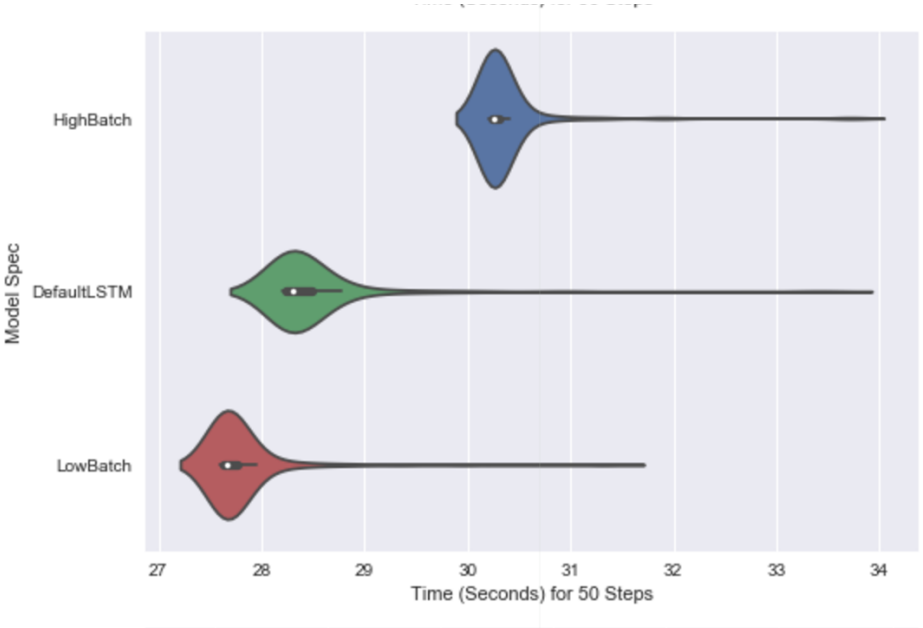

Figure 4.15: Batch Size Experiment Violin Plot

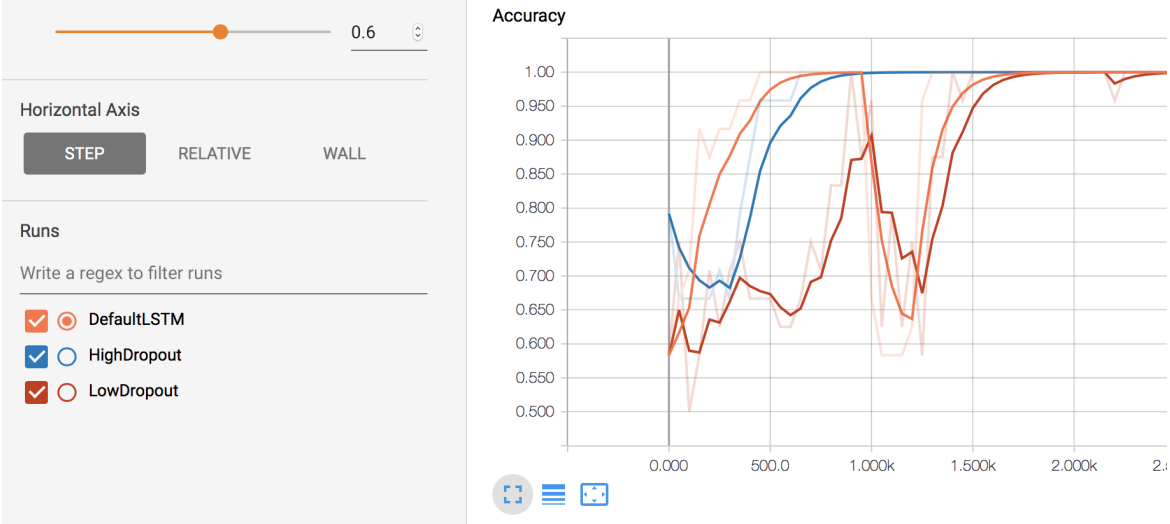

Figure 4.16: Dropout Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

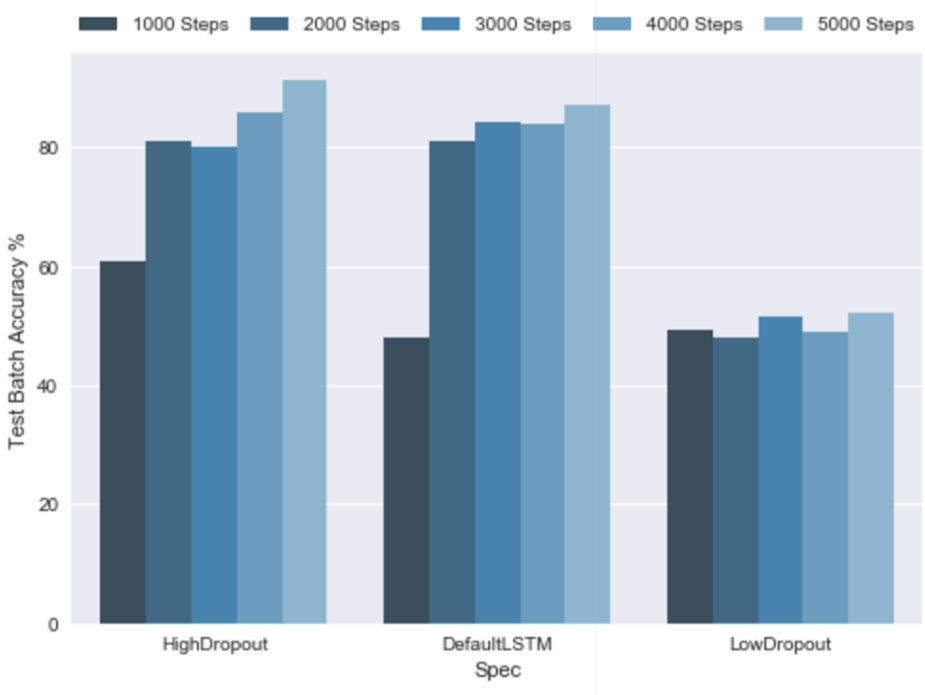

Figure 4.17: Dropout Experiment Test Accuracy

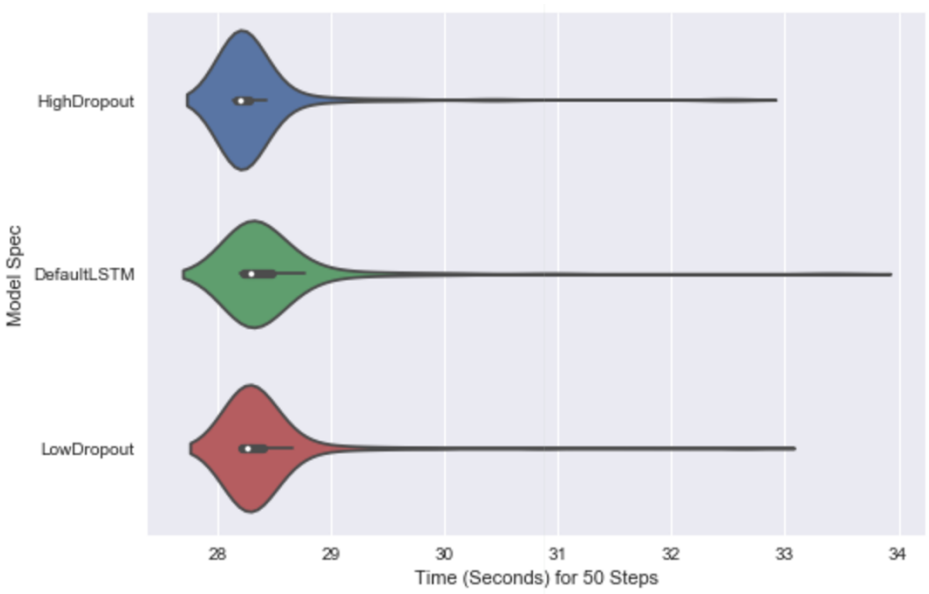

Figure 4.18: Dropout Experiment Violin Plot

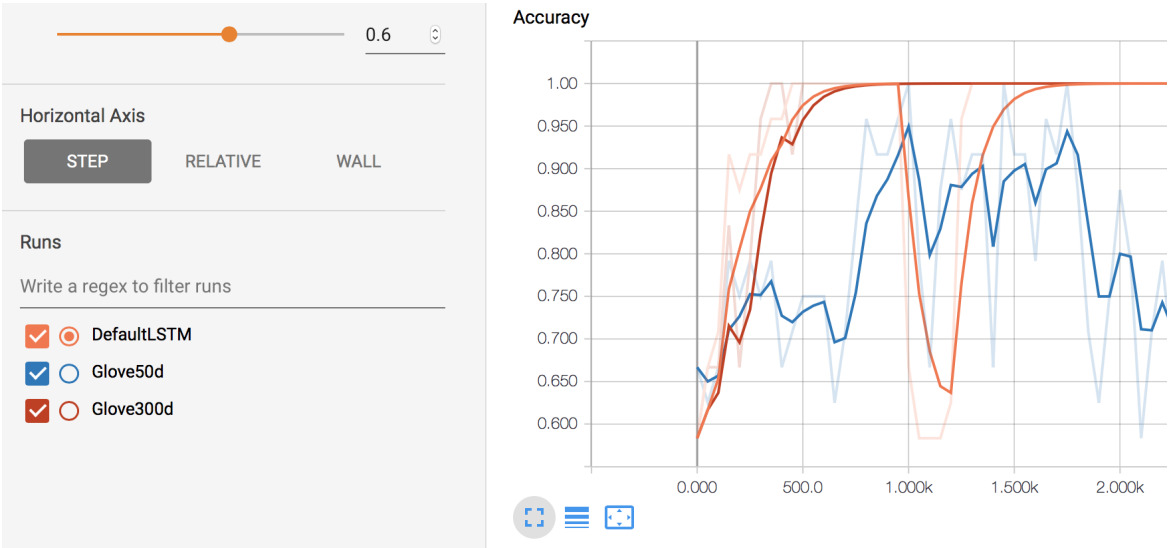

Figure 4.19: Vector Representation Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

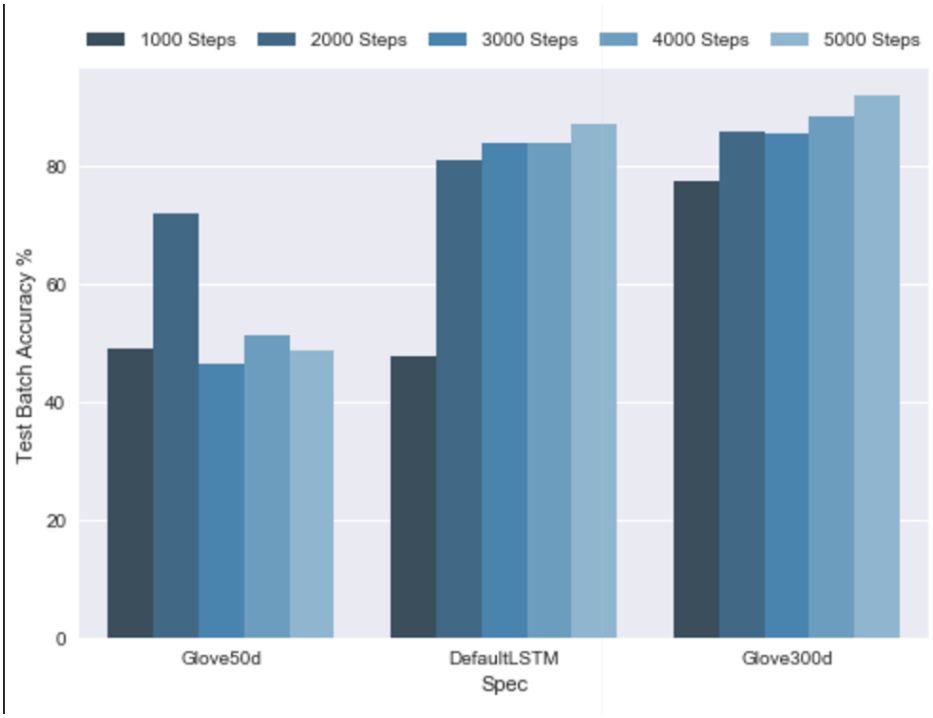

Figure 4.20: Vector Representation Experiment Test Accuracy

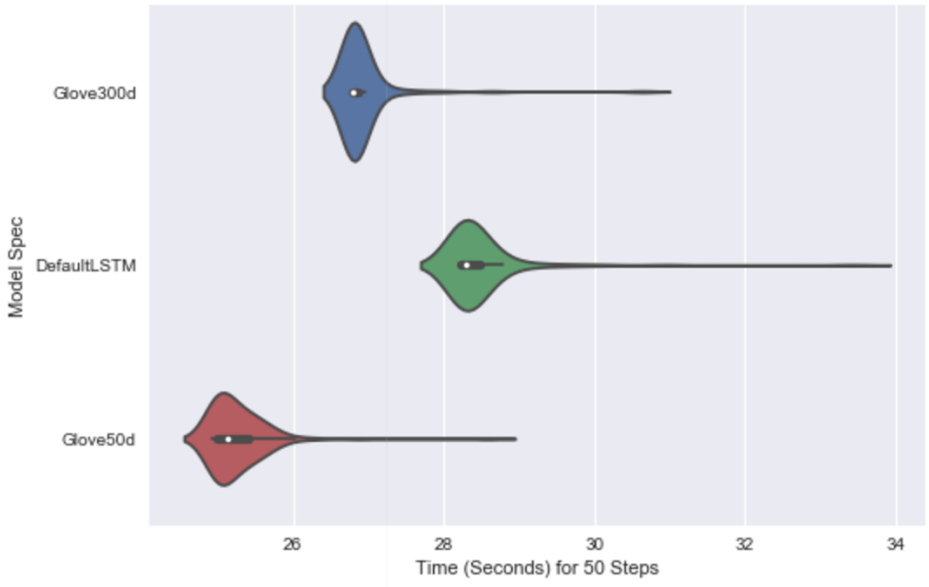

Figure 4.21: Vector Representation Experiment Violin Plot

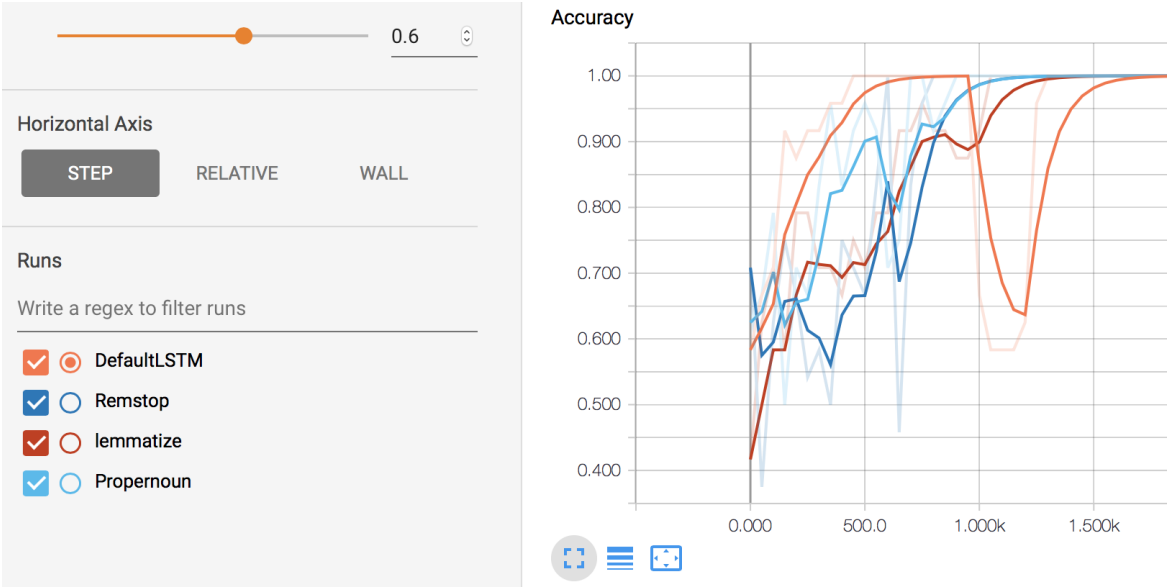

Figure 4.22: String Operations Experiment Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

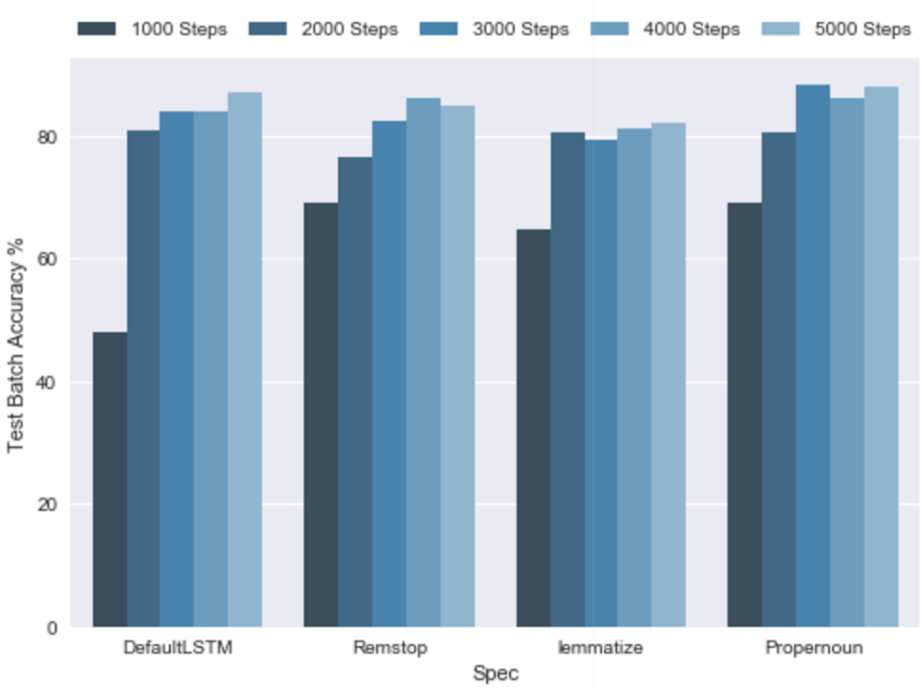

Figure 4.23: String Operations Representation Experiment Test Accuracy

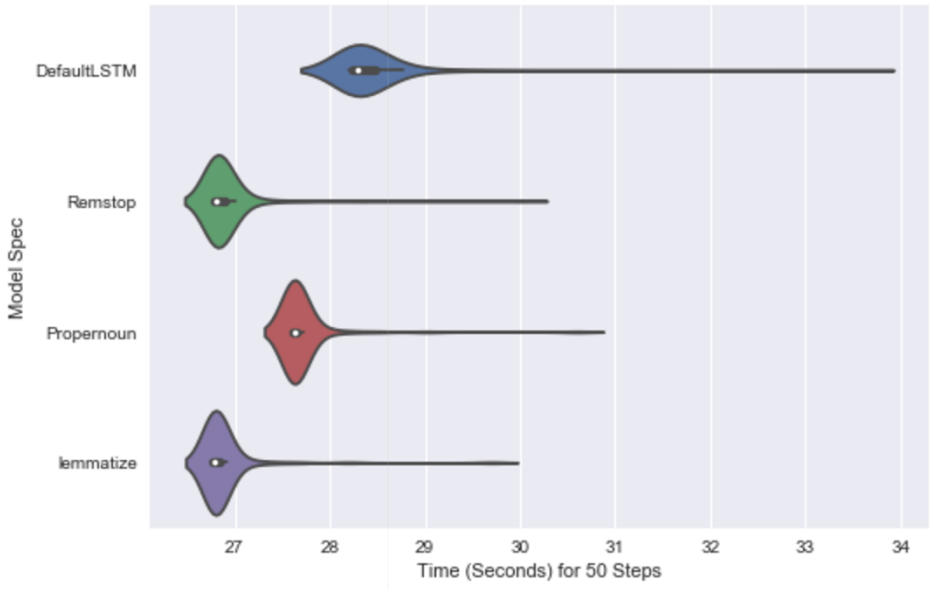

Figure 4.24: String Operations Representation Experiment Violin Plot

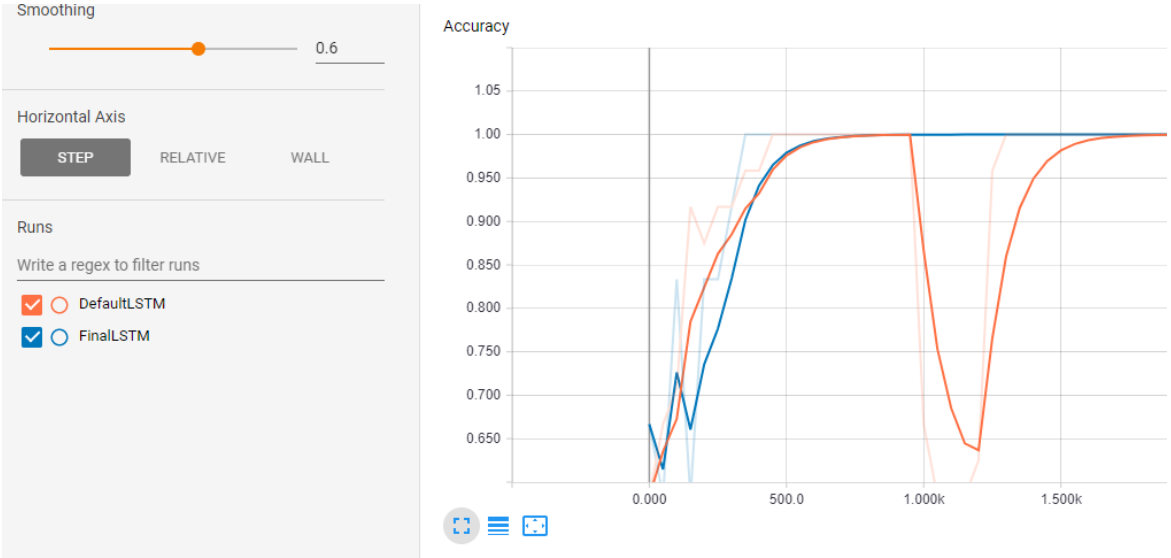

Figure 4.25: Optimal Settings Tensorboard Showing Accuracy of the Model during Training **Against the Training Data**

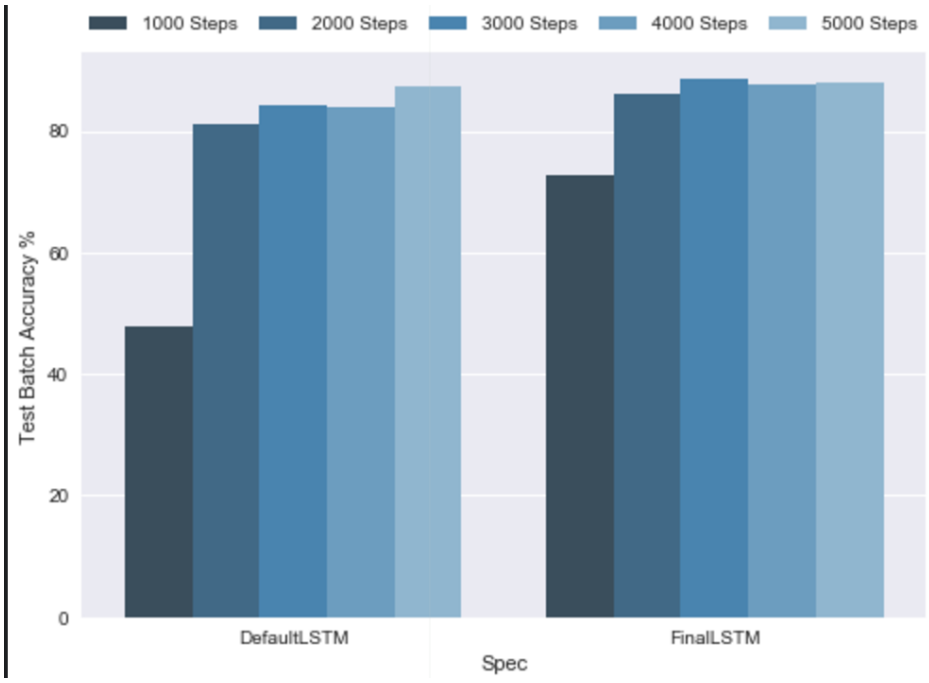

Figure 4.26: Optimal Settings Test Accuracy

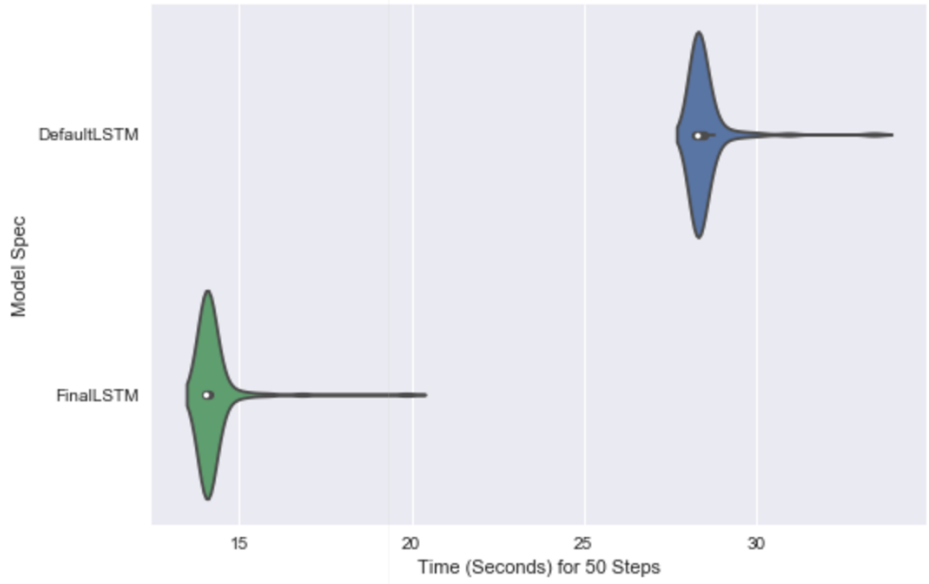

Figure 4.27: Optimal Settings Violin Plot

Chapter 5

ANALYSIS, EVALUATION & DISCUSSION

Analysis is presented in three formats. First, relevant observations are offered that became apparent during the practice of experimentation. Second, each experiment is considered in turn with comment given as to the results of each in terms of training, accuracy and speed. Finally each of the three research questions are reflected on in the light of these results.

5.1 General Observations

"Difficult" Batches and Stability

As mentioned in section 1.7, there are irregularities in training. For example the default LSTM suffered at the 1000 step mark (4.1). This might be explained by models facing "Difficult" batches, in which there is greater ambiguity amongst terms or a term is used in a way that counters previous weightings. This randomness amongst batches gives rise to blips in which a converged model may drop out of 100% accuracy amongst the training set (4.1, 4.7, 4.16).

How vulnerable a model is to 'difficult' batches gives rise to a new way of judging models; how stable they are over the course of training. Figure 4.25 shows the default

LSTM model with the final optimized model for comparision, showing a much more stable training profile.

Convergence

The majority of models have converged by 2k steps and increases in accuracy are marginal after this point. Overtraining does not seem to be an issue. Only the LowLSTM (4.8) and Glove 50d (4.20) models peak at earlier training iterations.

Convergence in a model does not mean the model isnt updating, it simply means it has arrived at weights that can give correct predictions for all the training sample it has currently been exposed to. A given model is unlikely to have been exposed to all of the training data by 1000 iterations but this tends to very likely by 5000 steps. All models see change in test accuracy between points of convergence (4000 vs 5000 steps for example).

Further, convergence doesnt mean accuracy. The CNN architecture converges quickly (by around 700 steps) but doesnt manage to get higher than 75% accuracy. Converged models with low test accuracy may be examples of overfitting the data which can be suggested intuitively; for example low dropout causes overfit because model weights are enforced more rigidly.

Training Time Irregularities

As mentioned in section 1.7, there are some irregularities in training times. For example, the Final LSTM has no reason to be slower than the short sequence LSTM (all of the parameter changes are seen in previous experiments to be speed accelerating) and yet it is (4.27 and 4.12). A possible explanation is that the Final LSTM ran on a GPU that had been freshly restarted whilst other models were ran alongside one and other in a row of 16. If this is the case then a confounding variable is introduced by the hardware state. Section 1.7 suggests a possible control for this but putting this into practice is outside the scope of this thesis.

5.2 Experiment Results

5.2.1 Architecture Experiment Analysis

Figures: 4.1, 4.2, 4.3

The CNN is quick to train and quick to converge, however it has a low accuracy ceiling of 74.8% compared to the default lstm at 87.3%. The LSTM continues to improve test accuracy up into the 4000+ wheras the CNN doesnt seem to have much uplift after the 2000 mark. From the CNN miss rate it looks like around 25% of texts have long term dependancies that are difficult to capture using n grams. Whilst it might be possible to improve on the CNN by optimising around hyperparameters, for the means of this thesis and in order to match the 96.5% accuracy achieved by (Barry, 2017), a 12.5% difference between the two architectures is enough to justify abandoning the CNN architecture and moving to experiment with just the LSTM model.

Optimal Decision - LSTM Architecture

5.2.2 Optimizer Experiment Analysis

Figures: 4.25, 4.26, 4.27

The SGD optimizer never converges where the adam optimizer does despite both being set at the same learning rate (0.01). One possible explanation of this is that the learning rate is too high for SGD and so fails to find a minimum. A further possibility is that the SGD has arrived at a local minimum and so weight updates are too small to fit the information found in new batches. There is no uplift in speed and the SGD does not get above 50.6% accurate. In keeping with (Reimers & Gurevych, 2017) this factor has a high impact on performance.

Optimal Decision - Adam Optimizer

5.2.3 Hidden Units Experiment Analysis

Figures: 4.7, 4.8, 4.9

The experiment produced some interesting results. Low lstm units had less effective results, losing 3.4% between 2000 and 5000 steps. This is likely due to underfit. The higher lstm unit spec cost on average 1.7 seconds longer to perform 50 steps, without convincing evidence of an uplift in accuracy. Conversely to low lstm this is likely due to overfit. (Reimers & Gurevych, 2017) state that this is of medium importance but it appears in this instance to be highly important to strike a balance.

Optimal Decision - Default Hidden Units

5.2.4 Sequence Length Experiment Analysis

Figures: 4.10, 4.11, 4.12

The experiment produce interesting results. Longer sequences are slower to converge, whilst shorter converges very quickly and is subsequently stable. As expected, truncated sequences are much quicker to run but with high cost to accuracy. The inverse is true of longer sequences, however an uplift in accuracy vs default is only the case at 1000, 2000 and 4000 steps. At 5000 steps the default outperforms long sequences by 1.5%. Truncated sequences cost 8.9% in peak accuracy but save 16.7 seconds.

Sequence length, more than any other hyper parameter, offers the greatest opportunity for accelerating training. For more discussion on this see section 5.3 optimal model.

5.2.5 Batch Size Experiment Analysis

Figures: 4.13, 4.14, 4.15

Smaller batches converge quickly (by 1000 steps) but overfits at earlier times. Higher batches also has a difficult batch around the 1000 step mark.

Smaller batches are both quicker to train and reach higher accuracies at 5000

iterations. Intuitively small batches would take longer to be exposed to the entire sample, which would account for the steadily increasing accuracy. Larger batches take longer to train and then produce lower accuracies. Batch sizes are important in and this is in keeping with (Reimers & Gurevych, 2017), who also recommend smaller batch sizes.

Optimal Decision - Small Batches

5.2.6 Dropout Experiment Analysis

Figures: 4.16, 4.17, 4.18

The experiment produced clear results. High dropout is much more effective at producing stable high results. Low dropout is both slower to converge and causes overfit. Furthermore, high dropout converges more quickly, gives stability to training and boosts accuracy at the highest iterations. High dropout is 2% more accurate than the default at 5000 steps. Neither high nor low changes training speed (0.1 seconds difference with the default). This parameter gives us a way of boosting accuracy and stability without costing accuracy.

Optimal Decision - High Dropout

5.2.7 Vector Representation Experiment Analysis

Figures: 4.19, 4.20, 4.21

The 50 and 300d glove vector representations are on average faster to train, by 3.2 and 1.6 seconds respectively. This might be explained by size of the file that the program must search through for a representation (the glove300d is 1.1gb, 50d is 260mb, the google word2vec is 3.7gb). The Glove 300d vector representation with default other settings achieves highest accuracy overall with 91.1% at 5000steps. The glove 50d representation struggles to converge, only managing by around 3400 steps. The glove 300d produces a more stable convergence, facing only minor blips at 2500 and 3000 steps. Use of the glove 300d representation should result in a more accurate and stable model whilst also raising speeds. The selection of vector representation is

very important in keeping with (Reimers & Gurevych, 2017), who also find a Glove representation the most effective in their instance.

Optimal Decision - Glove 300d Vector Representation

5.2.8 String Operations Experiment Analysis

Figures: 4.22, 4.23, 4.24

String operations are all found to have significant results. There is slight acceleration with all string methods, with lemmatizing speeding up the most on average at 1.7 seconds. Propernoun is the only spec to give convincing accuracy increase, of 0.7% at 5000 steps compared with the default. All 3 train fairly similarly, converge by the 1000 step mark and are stable. The concern with using string operations is that the wealth of the text is lessened in some way that damages the accuracy of the model, this does not appear to be the case with the three operations experimented on.

Optimal Decision - Use all String Methods Considered

5.3 Optimal Model Selection

- Architecture : LSTM
- Optimizer : Adam
- LSTM Hidden Units : 32
- Dropout : 0.75
- Sequence Length : 250
- Batch Size : 12
- Vector Representation : Glove300d
- String Operations : Removed Stop Words, lemmatizing and use of Propernoun Markers

A note on sequence length

Using default length sequences still could not improve on the accuracy of the default model with 300d glove vector representation. The optimal model uses truncated sequences as it was found there was an accuracy ceiling at around 91% that could not be exceeded even with average sequence lengths.

5.3.1 Optimal Model Results

Use of small batches, high dropout and Glove 300d representations make the model less vulnerable to 'blips'. Use of truncated sequences rapidly improves training time. The optimal model is 1.7% more accurate at 5000 steps than the default and trains 13.8 seconds faster for 50 steps. Further, the model is quicker to converge (by 400 steps) and more stable after point of convergence. Smaller test batches mean the optimal model overfits at 1000 steps but this is soon corrected with much higher accuracy by 2000 steps. 5.2 shows the optimum model compared with other models, showing the improvement in the two dimensional space of speed and accuracy.

5.4 Research Question 1 Revisited

Can deep learning models be made to assign a label of fiction or non fiction to a book review text with the same accuracy as state of the art accuracies in long sequence binary sentiment analysis?

Kim (2014): 81.5%

Thesis accuracy: 91.1%

Barry (2017): 96.5%

Peak accuracy exceeds that of (Kim, 2014), but does not match those of (Barry, 2017). Experimenting with architecture, hyper-parameters and text preprocessing could not give a correct prediction of the final 8.9%. The next step for further research is in depth error analysis so see if there are commonalities amongst the errors that might be accounted for in future modelling. Full accuracy comparison can be seen in figure

5.1.

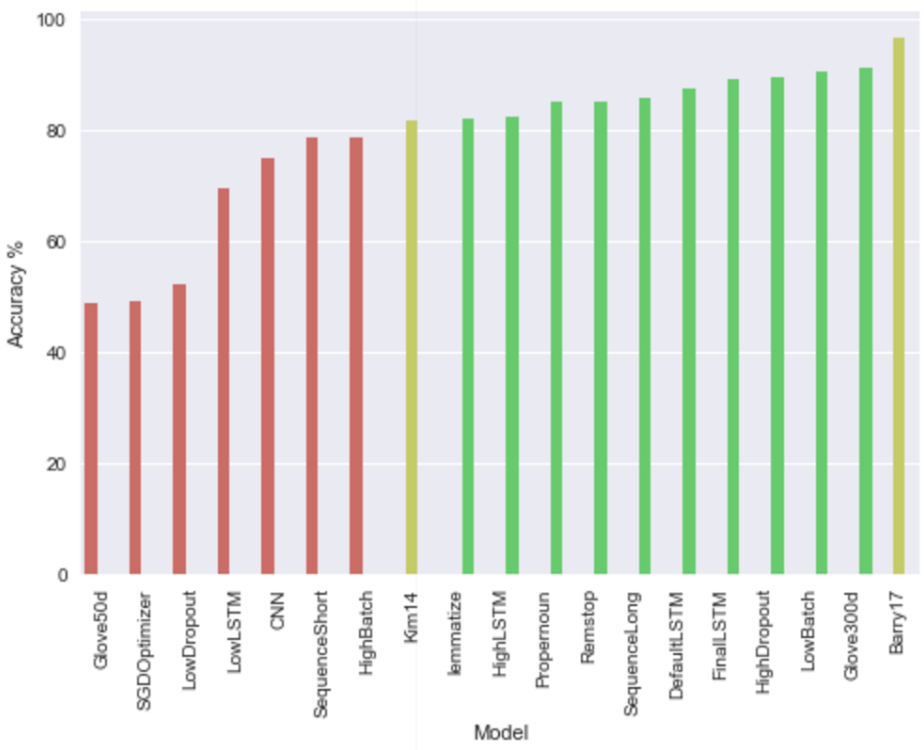

Figure 5.1: Peak performance of each model with (Kim, 2014) and (Barry, 2017) results for comparison, green models exceed performance of (Kim, 2014)

5.5 Research Question 2 Revisited

Are the hyper-parameters found by (Reimers & Gurevych, 2017) to be most important for accuracy in long sequence sentiment analysis the same as those in assigning a label of fiction or non fiction?

Contrary to (Reimers & Gurevych, 2017), every variable for experimentation was important to accuracy. There are two main differences between the findings of this thesis and those of (Reimers & Gurevych, 2017). First, (Reimers & Gurevych, 2017) find the number of LSTM hidden units to be of low importance. The results of this thesis show a 16.8% range in accuracy for this parameter, which we argue is significant and therefore of at least medium impact. Second the results of the thesis show the lowest range in accuracy to be in the batch experiment. (Reimers & Gurevych, 2017)

found this to be more important than other factors including the optimizer and the number of LSTM units. The results of this thesis suggest that the fiction or non fiction domain suggest that batching is less important than the optimizer or number of lstm units. Full results of comparative t-tests between accuracies are displayed in table 5.1.

Variable	MinAccuracy	MaxAccuracy	Z Score	p-Value
LSTM Hidden Units	0.6953	0.8729	18.56	0.000
Optimizer	0.4904	0.8729	35.31	0.000
Batch Size	0.7853	0.9053	14.27	0.000
Dropout	0.5221	0.9029	36.18	0.000
Vector Representation	0.4871	0.9110	39.74	0.000

Table 5.1: Proportion t-test performed on each comparable Hyperparameter *n=3699*

5.6 Research Question 3 Revisited

Does computational speed necessarily come at a cost to a accuracy?

As mentioned in section 5.3, there appears to be an accuracy ceiling that the thesis experimentation could not breach at around 91%. As a result it is possible to accelerate training without accuracy suffering too much, this forms the basis of the optimized model offered in section 5.2. We argue that computational speed does not necessarily come at a cost to accuracy and it is in fact more readily achievable to speed up a model of reasonable accuracy than to reach toward the higher 90 percentiles in accuracy. 5.2 shows the optimal model in comparison to other specifications.

5.7 Chapter Summary

In this chapter we have examined the experimental results for meaningful insight, revisited the research questions to measure the successes of the thesis and built an optimized model. The following chapter concludes the thesis by assessing the contributions of the thesis and offering suggestions for further work.

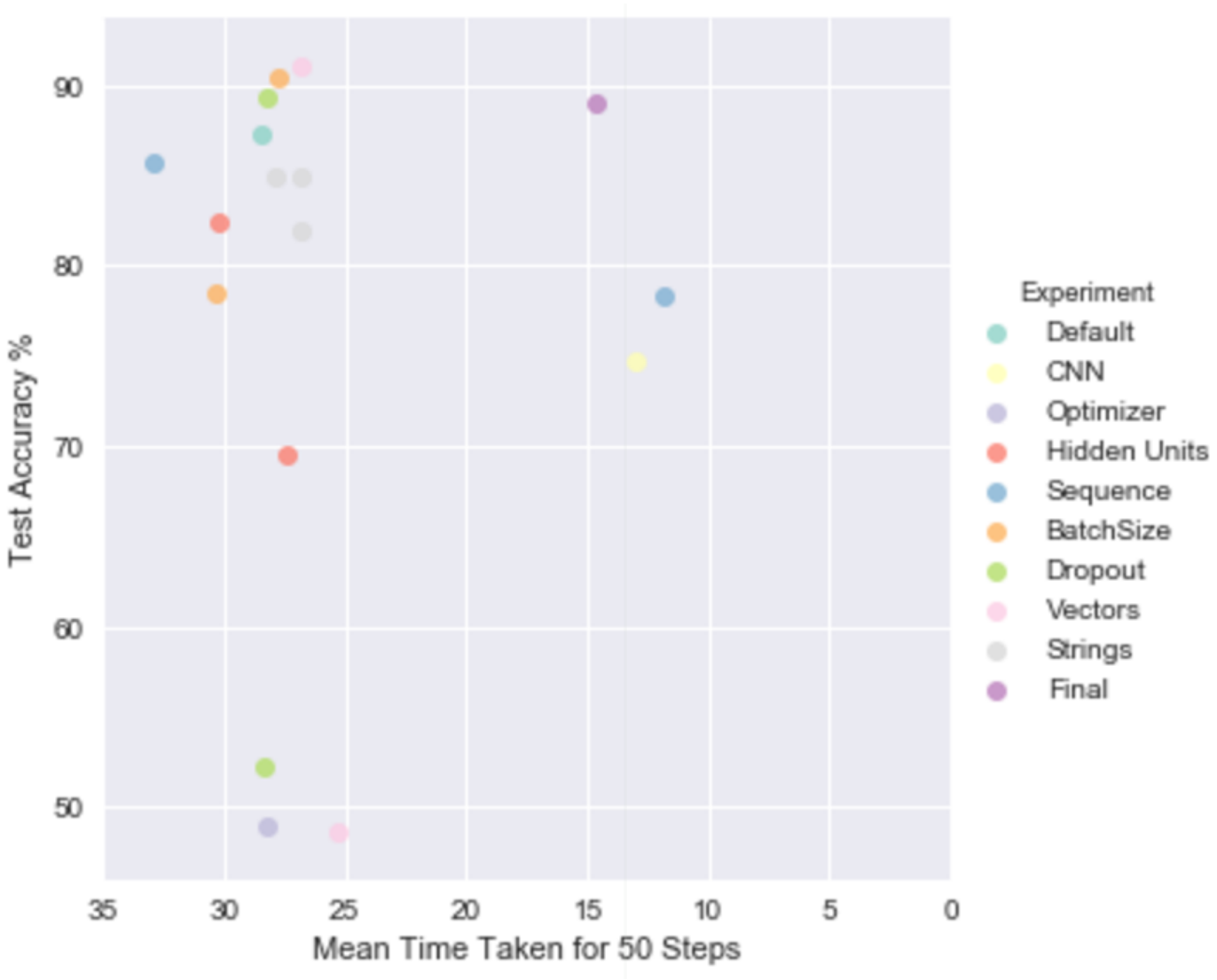

Figure 5.2: ScatterPlot with FinalLSTM

Chapter 6

CONCLUSION

6.1 Research Overview

The thesis aimed to apply machine learning techniques to book reviews to take the first step towards automated extraction of the information found in these reviews, by assigning a label of fiction or non fiction to the text. The thesis made use of neural networks and performed experiments around architecture, hyper-parameters and text processing from which an optimized model was produced. The thesis enjoyed certain successes; it was possible to match the state of the art achieved by (Kim, 2014) and computation was sped up considerably from the default to the optimized model by 13.8 seconds per 50 steps. Further it was confirmed by the thesis that labelling a sequence as fiction or non fiction can be performed most accurately with LSTM architectures and that contrary to (Reimers & Gurevych, 2017) every considered hyper parameter had a considerable impact on results.

6.2 Problem Definition

The problem faced by the thesis was efficient binary classification of a long sequence (a review text). The suitability of the model solution was judged in terms of the absolute accuracy a model can achieve on the test set and on the average time taken to perform 50 training iterations.

6.3 Design/Experimentation, Evaluation & Results

The thesis experiment design took the form of a series of neural network models built in Tensorflow, altering hyper-parameters whilst holding others constant to measure the effect of a given hyper parameter. Results were largely intuitive and could be put to use building an optimized model that ran both faster and more accurately than the default settings. For continuous parameters, experiments were performed between three arbitrary High, Mid & Low settings. This might be a flawed approach if there is a nonlinear relationship between the continuous parameter and accuracy or speed. We hypothesise further that two confounding variables introduce bias into the study; the difficulty of a batch and the state of the hardware in which models are run. Potential controls are suggested in section 6.5

6.4 Contributions and Impact

The chief contribution of this thesis is an optimized model for binary classification of a long sequence, that achieves 89% accuracy and is trained to 5000 steps in 24 minutes. The thesis has suggested several means by which models may be trained more quickly, increasing the feasibility for such models in a commercial envirionment where time is a constraint. This thesis has helped to shed light on the review text dataset and has added further experimental data to long sequence binary classification. Assigning a fiction or non fiction label to texts will allow the next stage of sentence level classification to be modelled; from which point genuine insights can be drawn from modelling as to the aggregate value of opinions on a given book. The research is potentially relevant across different fields as it suggests that binary sentiment classification techniques can be used outside of sentiment classification.

6.5 Future Work & Recommendations

- K-fold cross validation of the models would help to confirm the validity of each model.

- Performing the same set of experiments on a larger sample size would allow us to see if the sample size changes the results.

- Building on the work of the thesis the next step is to proceed to sentence level classification, for which a large labelled dataset and similar experiments around architecture, hyper parameters and text preprocessing would be needed. Intuition might suggest that the CNN architecture would be more suitable as sentence level is a shorter sequence.

- Further to this the work of the thesis might be verified by examining continuous parameters at more points than simply High, Mid Low points.

- Testing the hypothesis that instability is introduced by random batching. If confirmed, this might be controlled by stratifying batches based on an ambiguity rating, thereby ensuring that models face a certain proportion of ambiguous examples per iteration. However this would require the labelling procedure to be yet more time consuming.

- Testing the hypothesis that instability is introduced by model order and GPU state. If confirmed, this might be controlled in one of two ways; training models simultaneously on the same GPU with equal prioritising, which whilst it would slow training down would do so equally across all models. The second way involves training all models in every possible order and taking each model from the same state, i.e the end of the chain of 17 models.

References

Barry, J. (2017). Sentiment analysis of online reviews using bag-of-words and lstm approaches.

Chilimbi, T. M., Suzue, Y., Apacible, J., & Kalyanaraman, K. (2014). Project adam: Building an efficient and scalable deep learning training system. In *Osdi* (Vol. 14, pp. 571–582).

Clark, A. (2003). Pre-processing very noisy text. In *Proc. of workshop on shallow processing of large corpora* (pp. 12–22).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *CoRR*, *abs/1103.0398*. Retrieved from `http://arxiv.org/abs/1103.0398`

Denil, M., Demiraj, A., Kalchbrenner, N., Blunsom, P., & de Freitas, N. (2014). Modelling, visualising and summarising documents with a single convolutional neural network. *CoRR*, *abs/1406.3830*. Retrieved from `http://arxiv.org/abs/1406.3830`

Ganu, G., Elhadad, N., & Marian, A. (2009). Beyond the stars: Improving rating predictions using review text content. In *Webdb* (Vol. 9, pp. 1–6).

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, *17*, 26–32.

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. Retrieved from `http://dx.doi.org/10.1162/neco.1997.9.8.1735` doi: 10.1162/neco.1997.9.8.1735

Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, *abs/1408.5882*. Retrieved from `http://arxiv.org/abs/1408.5882`

Komninos, A., & Manandhar, S. (2016, 01). *Dependency based embeddings for sentence classification tasks.*

Li, J., Jurafsky, D., & Hovy, E. H. (2015). When are tree structures necessary for deep learning of representations? *CoRR*, *abs/1503.00185*. Retrieved from `http://arxiv.org/abs/1503.00185`

Lund, K., & Burgess, C. (1996, Jun 01). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, *28*(2), 203–208. Retrieved from `https://doi.org/10.3758/BF03204766` doi: 10.3758/BF03204766

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Retrieved from `http://arxiv.org/abs/1301.3781`

Mikolov, T., & Zweig, G. (2012, Dec). Context dependent recurrent neural network language model. In *2012 ieee spoken language technology workshop (slt)* (p. 234-239). doi: 10.1109/SLT.2012.6424228

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the acl-02 conference on empirical methods in natural language processing - volume 10* (pp. 79–86). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from `https://doi.org/10.3115/1118693.1118704` doi: 10.3115/1118693.1118704

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Reimers, N., & Gurevych, I. (2017). Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *CoRR*, *abs/1707.06799*. Retrieved from `http://arxiv.org/abs/1707.06799`

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, *abs/1503.00075*. Retrieved from `http://arxiv.org/abs/1503.00075`

Tang, D., Qin, B., Feng, X., & Liu, T. (2015). Target-dependent sentiment classification with long short term memory. *CoRR*, *abs/1512.01100*. Retrieved from `http://arxiv.org/abs/1512.01100`

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *CoRR*, *abs/1409.2329*. Retrieved from `http://arxiv.org/abs/1409.2329`

Zhang, Y., & Wallace, B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, *abs/1510.03820*. Retrieved from `http://arxiv.org/abs/1510.03820`

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2015). A C-LSTM neural network for text classification. *CoRR*, *abs/1511.08630*. Retrieved from `http://arxiv.org/abs/1511.08630`