Text classification using fastText neural networks

A comparison against state-of-the-art text classification techniques

IFN701	Research Project
Student:	Joshua Garvey 08334871
Supervisors:	Lance De Vine & Shlomo Geva

Contents

Abstract

This paper explores text classification utilizing fastText models with the objective of providing evidence of fastText's capabilities in comparison to conventional text classification methodologies. The SIGIR 2018 Rakuten Data Challenge dataset is used to draw comparisons due to the easily accessible results and methods of each team's models published by SIGIR as well as the importance of short-text classification (Lin, 2018).

This dataset is an example of short-text classification which is a relatively new sub-section of text classification with growing significance due to the increased importance of relevant datasets such as Twitter and instant messaging among other online services and applications. Sources relevant to this and other matters covered by the research report have been identified during a literature review prior to model implementation.

Data pre-processing techniques such as stemming, punctuation removal, ten-fold cross validation and ensemble modelling were utilized experimentally in attempts to improve the effectiveness of models. Of those tested, the most effective pre-processing technique for single model improvement was replacing digits with 0s such that product title strings retain the same length while normalising products of similar category such as differently sized rulers (30cm & 40cm both become 00cm). Stemming, sub-labelling and the removal of punctuation all proved to be detrimental to model performance, possibly due to the nature of short text classification and by extension the importance of details contained in the one input variable; the name of the product.

Model accuracy, represented using the F1 Score of a model over the dataset, was found to be comparable to those reported in the data challenge results paper (Lin, 2016) and in many cases out performed other techniques in this selection criteria (see Figure 3). Over 50 major models have been trained and tested over the course of this research project with varying results, the most effective techniques demonstrated were a combination of ensemble modelling between a "base" model which used no data pre-processing and a pre-processed model which almost outperformed the competing fastText model from the competition. It is recommended that, in future projects, further experimentation with HPC Lyra be performed in order to improve model training and test times in addition to working with different datasets.

1. Introduction

1.1 Background & Aim

Modern text classification tasks can range from information retrieval to natural language processing to data mining. The purpose of this research project is to investigate the use of a simple neural network approach to text classification and evaluate its effectiveness based on several benchmarks (precision, recall and F1 score). The focus of this project will be developing models using fastText (Joulin, 2018); a text classification and word representation library developed by Facebook's AI Research laboratory (FAIR). The aim of this project is to use Joulin's research in combination with our own practical modelling and research to compare the performance of fastText against more complex methodologies.

The primary dataset used in this research project is a subset of the SIGIR 2018 Data Challenge Rakuten Dataset; the data consists of a list of products from a Japanese website (Rakuten Global Market) along with an associated "label" for each given item. For the purposes of this research project, the objective was to determine the correct label for a given product. Accuracy was measured using Precision, Recall and F1 score which will be explain in the following 3 sub-sections. Analysis of this dataset is a form of short text classification, this terminology will be explained in sections 1.5 and 5.

1.1.1 Precision

The precision of a model is a measure of how many relevant results were returned as a percentage of all returned results (as described by the equation below):

$$Precision = \frac{TP}{(TP + FP)}$$
$$where\ TP = sum\ of\ all\ true\ positives$$
$$FP = sum\ of\ all\ false\ positives$$

1.1.2 Recall

The recall of a model is a measure of how many relevant results were returned as a percentage of all relevant results (as described by the equation below):

$$Recall = \frac{TP}{(TP + FN)}$$
$$where\ FN = sum\ of\ all\ false\ negatives$$

1.1.3 F1 Score

Precision and Recall can be summarised using their harmonic mean, known as the F1 Score which will be the main selection criteria used throughout this report:

$$F1\ Score = 2\left(\frac{P \times R}{P + R}\right)$$
$$where\ P = precision$$
$$R = recall$$

1.2 Research Questions, Propositions & Hypotheses

It was hypothesised that fastText would yield similar accuracy to conventional methods as described in the literature (Joulin, 2016). This forms the basis of the research question this paper seeks to answer i.e. Comparatively speaking, "How well does fastText perform relative to conventional state-of-the-art text classification methods?". As discussed in section 1.1, the selection criteria used for this comparison will be the F1 score of each model (see 1.1.1 – 1.1.3). The reasoning behind choosing this selection criteria was not only that it encompasses the precision and recall of a given model, but also because it makes comparison against other models implemented as part of the Rakuten Data Challenge very straightforward as they have also published these results (Lin, 2018).

1.3 Method Summary

The methodologies employed during this research project are as follows (points are ordered by appearance in this report, note that each point is further detailed in their respective sections; predominantly section 2 – Literature Review, section 3 – Project Methodology & section 4 – Results):

- A literature review has been performed in order to substantiate the current knowledge base surrounding fastText, the Rakuten dataset and the current state of research regarding text classification (section 2)
- Models were trialled on the dataset without pre-processing the data to establish a baseline model and tweak initial parameters (section 3 & 4)
- Various pre-processing methods were implemented with varying results (see section 3 for more details on each pre-processing step)
    - punctuation removal
    - lowercasing all characters
    - n-grams
    - replacing numbers e.g. replacing all digits with 0
    - sub-labelling
    - stemming
- Ensemble models were derived from complementary models implemented (section 3)
- Model results were compared against the 2018 Rakuten Data Challenge leader board (section 4)
- Additional pre-processing methodologies were implemented based on those used by other teams during the challenge itself (section 3)
- Findings were summarised into the final report and presentation

1.4 Scope & Deliverables

The scope of this research project includes the development of a pre-processing pipeline implemented predominantly using bash scripting as well as the review of relevant literature associated with the Rakuten data challenge, fastText and text classification methodologies. In addition, it includes the implementation and testing of the various data cleaning operations outlined in section 1.3 above and section 3 below.

While originally it was my intention to perform an analysis of fastText on several text classification problems, however testing the desired parameters and cleaning methods proved more time-intensive than previously estimated. As such, alternative text classification tasks such as sentiment analysis will be assigned to "future-scope" for additional investigation in further research projects. This report also contains guidelines for using fastText for text classification (section 3 below) along with the standard deliverables; the meeting minutes, final presentation and this report.

1.5 Anticipated Significance

Short text classification is an area growing in relevance due to the growth of services such as Twitter and instant messages, as such research in this area is needed to better understand the best processes available to analyse such datasets. This underlines the importance of this research project as the classification of the Rakuten dataset is a form of short text classification.

Challenges involved in short-text classification arise from the unique features of short text datasets; these datasets are generally sparse, large-scale and lack a form of standardisation found in other data such as the structure found in HTML files or journal articles. According to Song's paper "Short Text Classification: A Survey", the main objectives of a short text classification process are to reduce feature dimensions and extract features using semantic relationships (Song, 2014). It also recommends the use of ensemble classification to deal with "large scale short text [datasets]". As such this will be implemented once a basic understanding of the effects of model parameters on model performance is established.

2. Literature Review of Previous Work

2.1 Justification of Motivation

The reasoning behind this project stems not only from the implications of a linear classifier performing as well if not better than conventional text classification methods, but also from the significance of short text classification as an area of research. While both of these topics have been covered in section 1 – Introduction, they will be briefly discussed in the section below.

2.1.1 fastText and the power of linear classifiers

As a tool for text classification and word representation, fastText is a linear classifier which is proposed to "train on a billion words within ten minutes, while achieving performance on par with the state-of-the-art" (Joulin, 2016). Linear classifiers can often train and test on larger scale datasets faster than conventional state of the art techniques due to the simplicity of their models. The implications of this possibility are that model complexity can be lowered while maintaining similar performance in terms of accuracy; meaning that models will give similar results with the added benefit of training and testing faster than some conventional text classification models.

2.1.2 The significance of Short Text Classification

The ever-growing popularity of Twitter, Facebook, instant messaging platforms and similar services has given rise to the introduction of a new genre of text classification. As described in the article "Using deep learning for short text understanding", short text classification has a unique set of problems such as a lack of contextual information due to the shear lack of characters in each "document" e.g. tweets, messages and so on (Zhan, 2017).

2.2 Approach

2.2.1 Methodology

Relevant articles were identified using several search engines including the QUT Library Database search functions and Google Scholar; it was often the case that an article was found using Google's robust search features but inaccessible due to subscription requirements, this is where the library database is useful as the university's various subscriptions often allows access to these articles. QUT Quick Find also features a peer-review filter which was used to great effect during the article selection process. Referencing is performed at the time of paper acquisition such that it is easy to keep track of articles when references need to be made in the future.

2.2.2 Organization

The results of this literature review are structured in order to address several key criteria (detailed in section 2.2 below), the intention being to summarise the overall findings of the review and allow discussion about multiple documents within the same topic. One of the benefits of this structure is that when multiple articles serve a similar purpose or complement each other in their findings/value to the research project, they can be discussed within the same section.

Sections have been adapted from the IFN600 (Understanding Research) subject's literature analysis assessment and as such there may be a similarity in the titles used throughout this section of the report. Using this as the template allowed for a more thorough criteria to be outlined for the literature review itself (Geva, 2018).

2.2.3 Scope (topics & types of literature)

The majority of sources used in this report are journal articles due to the ease with which validity can be determined for these resources; as mentioned in 2.1.1, QUT Quick Find and Google Scholar make it very easy to filter out non-peer reviewed content in order to quickly establish the quality of a given document. Topics used in search terminology include but were not limited to text classification, fastText, data pre-processing and natural language processing.

While this report will speak briefly about other text classification tasks, due to time restrictions it will not go into detail about other datasets for the purposes of sentiment analysis, spam detection, product recommendation and so on. It is my recommendation that these be kept as "future scope" should this project be continued at a later date. As such these topics will only be reviewed in literature for the purposes of establishing knowledge gaps, inconsistencies and the like during the literature review.

2.3 Literature Analysis

2.3.1 Influential Studies

One of the most influential studies used during this research project was the Overview of the SIGIR 2018 eCom Rakuten Data Challenge (Lin 2018), which is a summary article of the various models and pre-processing methods used during the data challenge by each team. The "RITB-Baseline" method described on page 4 utilises fastText for model generation and as such was a major focus for model optimisation as well as data pre-processing strategies.

Bag of tricks for Efficient Text Classification is a study on the performance of fastText written by its developers explaining how it *"is often on par with deep learning classifiers in terms of accuracy"* (Joulin, 2016). This article was of particular significance as it explains the basic parameters of fastText in reasonable detail, such as the advantages of hierarchical softmax on datasets with a large number of classes (in the case of the Rakuten Dataset, each unique label is a separate class) and how n-grams are used to partially account for word order.

Ge Song's paper was of particular interest due to its focus on short text classification. It discusses the difficulties, existing technology and prospects of short text classification. While the full details of methods detailed in this paper will not be elaborated on here, Song explains several methods used for short text classification such as using "Latent Semantic Analysis (LSA)" and "Latent Dirichlet Allocation (LDA)" (Song, 2014).

2.3.2 Knowledge Gaps

This project's methodology is a form of *short text classification* due to the nature of the dataset, similar tasks include the classification of Twitter posts where each "document" is relatively short. A paper on short text classification by Song establishes that short text classification is a reasonably "new genre" and comes with a plethora of new challenges not well suited to traditional methods due to the "sparseness, large-scale, immediacy [and] non-standardization" of short-text datasets. This type of dataset is especially difficult to represent as a feature set due to the lack of information on relationships between terms and documents compared to traditional text-classification in which entire articles are used instead of nouns or phrases (Song, 2014).

Another major issue present in text classification is the extraction of concepts and deeper meaning from documents due to the ambiguity of natural language, other issues include Polysemy and Synonymy i.e. words with multiple meanings and words with similar meanings respectively (Thangaraj, 2018). While this report doesn't discuss a direct solution to this issue, it is mentioned in several other sections (for example 2.2.2 – Influential Studies) that N-Grams have been used in an attempt to account for word order thereby softening the effects of natural language ambiguity.

2.3.3 Inconsistencies

One of the more obvious inconsistencies in the literature is the methods by which text classification is achieved. Teams from the Rakuten Data Challenge applied a wide range of methods from K Nearest Neighbour (KNN) and Best Match 25 (BM25) to Convolutional Neural Networks (Lin, 2018); each having a unique approach to data pre-processing necessitated by the differences in the models themselves.

Another form of inconsistency is identified in the results published by Le and Mikolov in their 2014 paper "Distributed Representations of Sentence and Documents" (Le, 2014). Briefly searching for discussions on this paper reveal arguments as to whether or not the results published are repeatable. Some sources stating "*It's true that Quoc Le's results on the dmpv version of doc2vec have been hard to reproduce. However [evidence shows] that it can be reproduced by not shuffling the data. It's likely that this was an oversight*" (YCombinator, 2017).

Mikolov, one of the co-authors of the aforementioned paper, published a paper later with footnotes directed at these accusations stating *"In our experiments, to match the results from (Le & Mikolov, 2014), we followed the suggestion by Quoc Le to use hierarchical softmax instead of negative sampling. However, this produces the 92.6% accuracy result only when the training and test data are not shuffled. Thus, we consider this result to be invalid."* Indicating the lack of reproducibility of Quoc Le's original results without shuffling (rearranging the order of data) (Mesnil, 2014). While document to vector (doc2vec) is not implemented as part of this research project, discrepancies such as this highlight the current state of text classification in industry.

2.3.4 Major Concerns

Literature sources which discuss optimisation of fast text models often involve highly intensive parameters such as using 300 dimensions with 3000 epochs and 30 threads. As such one major issue with model optimisation has been the lack of appropriate hardware; running models of this magnitude on my laptop (8 logical processors and thus no more than 8 threads) takes almost 10 hours just using hierarchical softmax. Note that this is without looping through parameters in order to identify plateau points such as those found in dimensions and epochs (discussed further in section 4 – Results).

While attempts have been made to run models using QUT's HPC Lyra in order to speed up model training and testing with higher parameter sets such as those described above, difficulties have arisen in using Lyra due to the queue system and my lack of understanding regarding its operation. Several models have been trained and tested using HPC Lyra, however time limitations prevented these results from being collected prior to the end of semester. In future projects, a more formal understanding of the procedures involved in using Lyra will allow for rapid model prototyping which will allow for more rigorous testing.

3. Project Methodology

3.1 Data Pre-Processing

Various pre-processing techniques were implemented with the intention of improving model performance. While not all techniques proved beneficial, all will be documented in the sub-sections below for the purpose of reporting, for more details regarding the performance of each pre-processing step, see the relevant subsections of section 4 – Results. Note that not all models utilised all pre-processing methods due to the effectiveness of certain methods over others.

3.1.1 Punctuation & Capitalization

Using the translate/delete function "tr" in a bash script allowed for effectively removing both punctuation and capitalization from each line in the dataset. The purpose of removing these properties of the data was to normalise lines such that similar products were given the same or similar labels.

3.1.2 Digit Replacement

Using the pattern replacement functions provided by 'sed', digits present within product descriptions were replaced with '0'. Similarly to 3.1.1, the purpose of this pre-processing step is to normalize the data such that similar items with different numeric characters in their description are labelled similarly e.g. such that a "30cm ruler" is given the same or similar label to a "40cm ruler" as both titles become "00cm ruler" and are thus identical in properties derived during training.

3.1.3 Stemming

Stemming was achieved using an ANSI C implementation of the Porter Stemmer Algorithm. As is the case for pre-processing described in the above two subsections, stemming was implemented with the intention of normalising words within the dataset to increase the similarity of similar products as identified by a given model. In practice, stemming is particularly useful for information retrieval due to the ability to search for one term and have similar terms returned e.g. searching for "destruction" and returning results for destructiveness, destructive and so on (Porter Stemming Algorithm, 2018).

3.1.4 Sub-Labelling

Labels for a given product come in the form "3292>114>1231" where the left-most number is the "home" directory and numbers proceeding it are subdirectories. Sublabels for the example label are "3292>114" and "3292". These labels were added to the model prior to training in an attempt to teach the model the hierarchy of the website's product labelling system i.e. where a replacement battery may have the directory "3292>114>1231" which means it is also part of directories "3292>114" and "3292". In order to appropriately label these products at test time, the top 50 results for each product were produced and the first (most probable) full path (non-sublabel) was taken as the predicted label.

3.2 Model Parameters

3.2.1 Epochs

The epochs parameter is used to describe the number of times the model will be trained on a given dataset i.e. if epochs is set to 6, the model will view the entire dataset 6 times during training. While results from the "RITB-Baseline" described in the Rakuten Data Challenge summary paper (Lin, 2018) reported best performance with 3000 epochs, experimental modelling actually yields similar result (within 0.01% difference in accuracy according to F1 scores) with 10 epochs. It should be noted that the pre-processing steps used in their model were different to ours and as such this may not simply be attributed to a difference in epochs, differences will be detailed further in section 4 – Results.

3.2.2 Word N-Grams

The word n-grams parameter describes the size of n-grams to be used when building a given model. For example; if word n-grams is set to 2 then bigrams are created during training such as "leather shoe" or in the case of 3 (trigrams) "Italian leather shoe". The purpose of using n-grams, as reiterated throughout this report, is to attempt to account for the importance of word order in text classification. This is used because word order is not accounted for using fastText under normal circumstances. (Joulin, 2016).

3.2.3 Dimensions

The dimensions parameter "dim" is used to specify the number of dimensions used to calculate a given word vector, word vectors are used in text classification using neural networks to define the properties of a given term (word) where each dimension holds a numeric value. Dimensions are a property of any fastText model and as such their purpose is tied to the functionality of fastText itself.

3.2.4 Loss Function

The two main Loss functions used during this research project are the Softmax and Hierarchical Softmax. Negative Sampling is not explored as training and test times are similar to Softmax with similar results for our measures of accuracy between the two loss functions (Precision, Recall and F1).

Hierarchical Softmax proved most useful while trialling different sets of model parameters (increasing or decreasing dimensions, epochs, word n-grams and so on) as it is orders of magnitude faster than softmax; experimentally speaking, models which take 1 hour to train/test using softmax will often take 1 to 3 minutes using hierarchical softmax. This is because hierarchical softmax is an approximation of softmax. As with dimensions, the loss function is a mandatory component of any fastText model and as such its purpose is tied to the functionality of fastText itself.

3.2.5 Threads

The threads parameter is used to adjust the number of threads used while training the model, generally speaking the developers recommend that the number of threads used be no larger than the number of logical processors within a machine. The majority of modelling was performed on a laptop with 8 logical processors, however 7 threads proved to be more useful as the CPU usage would "throttle" when 8 were used (possibly due to background processes). As with dimensions and loss function, threads are a compulsory variable which is set to 12 by default and as such its purpose is tied to the functionality of fastText itself.

3.3 Ensemble Modelling

Ensemble modelling was achieved by taking the highest probability predicted label for a number of given models, comparing those and taking the highest probability for each row of the dataset i.e. if model 1 predicts the label "92" with a probability of 99.05% while model 2 predicts the label "92>2>412" with a probability of 99.04%, model 1's result is taken as the accepted value. Currently in the case of ties, the first model's value is taken, future implementations of the ensemble model may provide a secondary method by which to break ties in probability. The purpose of implementing an ensemble model is to take advantage of the positive traits of multiple models, for example; one model may remove punctuation prior to testing and training while another takes the base values.

3.4 Ten-Fold Cross-Validation

Ten-fold cross validation was achieved by splitting the dataset into 10 equal-length sections, each subset was used as a test set while a model was trained on the remaining 9. The precision, recall and f1 scores for each model were then averaged (arithmetic mean). The purpose of this process is to reduce the effects of skewed data. For example; if the last 10% of the data are all the same unseen product which isn't present in the other 90% of the data, then training a model of the 1st 90% would not allow the model to accurately determine the labels of the remaining 10% thus giving a poor and misleading accuracy for the model.

4.Results, Findings and Outcomes

4.1 Results

Several rounds of testing were implemented during model optimisation, as such the following results have been separated to display this.

Table 1. Initial fastText results comparisons

Initial fastText Results Comparison (Hierarchical Softmax)						
	Precision @			Recall @		
Trial (Model)	1	5	10	1	5	10
Raw						
1	0.65	0.174	0.109	0.65	0.728	0.751
2	0.654	0.174	0.109	0.654	0.725	0.749
3	0.654	0.174	0.11	0.654	0.729	0.754
4	0.657	0.175	0.11	0.657	0.731	0.753
5	0.654	0.174	0.11	0.654	0.726	0.748
Mean	0.654	0.174	0.110	0.654	0.728	0.751
Pre-Processed (punctuation and case sensitivity)						
1	0.65	0.178	0.112	0.65	0.733	0.756
2	0.651	0.177	0.112	0.651	0.732	0.757
3	0.652	0.18	0.113	0.652	0.733	0.757
4	0.654	0.179	0.113	0.654	0.734	0.757
5	0.649	0.177	0.112	0.649	0.729	0.753
Mean	0.651	0.178	0.112	0.651	0.732	0.756
Stemming (Porter Stemmer Algorithm)						
1	0.647	0.175	0.111	0.647	0.725	0.752
2	0.649	0.177	0.112	0.649	0.732	0.757
3	0.651	0.177	0.111	0.651	0.731	0.754
4	0.65	0.177	0.112	0.65	0.73	0.752
5	0.653	0.175	0.111	0.653	0.727	0.754
Mean	0.650	0.176	0.111	0.650	0.729	0.754

Initial fastText model performance was evaluated using fastText's inbuilt testing function, which only returned precision and recall. As such these results proved less useful in the later stages of the research project. Precision and Recall "@ 1" is equivalent to the precision and recall returned by the python script used to evaluate later models, however it was deemed easier to use the python script which also returned f1 scores rather than manually calculating this value each time. The colour filters implemented show the most desirable values in green with the least desirable in red, colours used are specific to a given column i.e. green in precision@1 is not equivalent to green in precision@5.

Table 2. fastText model results evaluated using the python evaluation script

Model	Pre-Processing Description	epoch	wordNgrams	buckets	dim	loss	Precision	Recall	F1 Score
Model Group 1 (initial models, evaluated using fastText evaluation thus no F1 score recorded)									
1	removed punctuation, alphabetical characters normalised to lowercase, digits normalised to zero	25.00	2	200000	50.00	hs	0.735	NA	NA
2		25.00		1000000	50.00		0.768	NA	NA
3		25.00		2000000	50.00		0.774	NA	NA
4		25.00	3		50.00		0.777	NA	NA
5		25.00			100.00		0.78	NA	NA
6		20.00			100.00		0.778	NA	NA
Model Group 2 (evaluated using python script to automate P, R & F1 calculation)									
7	as above	20.00	3	2000000	100.00	hs	0.7741	0.7782	0.7719
8		25.00			100.00		0.7729	0.7775	0.7709
9		40.00			100.00		0.7732	0.7785	0.7716
10		100.00			100.00		0.7688	0.7771	0.7686
11		100.00			150.00		0.7694	0.7777	0.7691
Model Group 3 (experimenting with high number of epochs & dimensions)									
12	as above	50.00	3	2000000	300.00	hs	0.7714	0.7707	0.7667
13		100.00			300.00		0.7732	0.7712	0.7681
14		200.00			300.00		0.7751	0.7722	0.7697
15		500.00			300.00		0.776	0.7723	0.7704
16		1000.00			300.00		0.7773	0.7733	0.7715
17		2000.00			300.00		0.7779	0.774	0.7722
Model Group 4 (discovered presence of punctuation improves model performance, accuracy can be improved with lower epochs)									
18	digits normalised to zero only	10.00	3	2000000	120.00	hs	0.7754	0.7787	0.7726
19		10.00			200.00		0.7757	0.7787	0.7729
20		10.00			500.00		0.7757	0.7788	0.7728
21		10.00			200.00	softmax	0.8188	0.8134	0.8117
22		10.00			200.00	softmax	0.8189	0.8141	0.8122
Model Group 5 (Sub-Labeling Comparisons)									
23	as above without sub-labelling	20.00	3	2000000	100.00	softmax	0.8112	0.8092	0.8061
24	as above with sub-labelling	20.00			100.00		0.7925	0.801	0.7894
Model Group 6 (increasing epochs with puntucation present)									
25	digits normalised to zero only	25.00	3	2000000	200.00	hs	0.7793	0.7816	0.7762
22		50.00			200.00		0.7799	0.7825	0.7766
27		100.00			200.00		0.7769	0.7802	0.7742
28		50.00			200.00	softmax	0.8151	0.8154	0.8107

This table summarizes the majority of models created during this project; models are divided into major changes described by model groups. The colour filters are column specific such that values of one column do not interfere with the colouration of another. Models using the Softmax loss function were not included in the colour filters due to the large improvement over hierarchical softmax models. In most circumstances, improvements to a model using hierarchical softmax will cause improvements in the same model using softmax, as such the former was used in most trials due to its time efficiency then finally tested using the latter in order to achieve the best results.

Table 3. Ensemble Model Results

Trial	Model Description	epoch	wordNgrams	dim	loss	Precision	Recall	F1 Score
1	Initial Ensemble Testing using	25	3	100	hs	0.7859	0.7914	0.7835
2	Base & Zeros Models	25			softmax	0.8168	0.8178	0.813
3	Test using less epochs with the same 2 models	10		120	hs	0.7882	0.7926	0.7849
4					softmax	0.8191	0.8175	0.8139
5					softmax	0.8188	0.818	0.814
6					softmax	0.8187	0.8178	0.8139
7	Three Models: Base, Zeros and Stemmer				hs	0.7845	0.7909	0.7823
8					softmax	0.8154	0.8155	0.8113
9	Test 2 models with punctuation	10		200	softmax	0.8192	0.8181	0.8141

Similar to Table 2, the ensemble results are grouped by changes in model parameters as well as models used. The majority of ensemble modelling was performed using softmax in order to produce the best possible results.

Table 4. Ten Fold Cross Validation Trials

Ten Fold Cross Validation Results											
Model Parameters	../FastText.exe supervised -input data/rdcTrain$i -output data/model_rdc$i -lr 1.0 -epoch 10 -wordNgrams 3 -dim 50 -loss hs -thread 7										
Sub-model	1	2	3	4	5	6	7	8	9	10	Mean
precision	0.7726	0.7705	0.7685	0.7709	0.7695	0.7676	0.7691	0.7685	0.7704	0.7699	0.7698
recall	0.773	0.7719	0.7694	0.7714	0.771	0.7694	0.771	0.7697	0.7716	0.7705	0.7709
f1	0.7681	0.7668	0.7646	0.7669	0.7657	0.7637	0.7654	0.7644	0.7666	0.7656	0.7658
Model Parameters	../FastText.exe supervised -input data/rdcTrain$i -output data/model_rdc$i -lr 1.0 -epoch 10 -wordNgrams 3 -dim 50 -loss softmax -thread 7										
Sub-model	1	2	3	4	5	6	7	8	9	10	Mean
precision	0.8147	0.8135	0.8135	0.8154	0.8117	0.8104	0.8121	0.811	0.8139	0.8112	0.8127
recall	0.806	0.8056	0.8053	0.8065	0.8045	0.8027	0.8052	0.8032	0.8057	0.8017	0.8046
f1	0.8063	0.8058	0.8053	0.8071	0.8045	0.8027	0.8048	0.8031	0.8059	0.8023	0.8048

Ten fold cross validation results are found by taking the arithmetic mean of 10 versions of the same model using different subsets of the training and test data (as described in Section 3.4).

Table 5. Model comparison against SIGIR 2018 Rakuten Data Challenge Results

Rank	Team	Precision	Recall	F1 Score
1	mcskinner	0.8697	0.8418	0.8513
2	MKANEMAS	0.8425	0.8427	0.8399
3	tiger	0.8397	0.8428	0.8379
4	Uplab	0.8368	0.8419	0.8366
5	JCWRY	0.8528	0.8172	0.8295
6	neko	0.8267	0.8305	0.8256
7	Ravenclaw	0.8289	0.8114	0.8175
8	Uplab-2	0.8186	0.8243	0.8173
9	ssdragon	0.8226	0.8163	0.8172
10	RITB-Baseline	0.8276	0.8077	0.8142
	IFN701 Project	**0.8192**	**0.8181**	**0.8141**
11	inception	0.8259	0.8077	0.8139
12	Tyche	0.8599	0.7644	0.8004
13	minimono	0.8019	0.8023	0.7994
14	Topsig	0.7921	0.8014	0.7941
15	VanGuard	0.7899	0.7917	0.7884
16	HSJX-ITEC-YU	0.7809	0.7821	0.779
17	Waterloo	0.7802	0.7857	0.7781
18	CorUmBc	0.7745	0.7712	0.769
19	Sam-chan	0.7718	0.7745	0.7666
20	Tyken2018	0.7654	0.7603	0.7509
21	Or	0.7419	0.725	0.7245
22	Coumodo	0.7275	0.714	0.7107
23	Uplab-3	0.6698	0.6588	0.6509
24	the1owl	0.5947	0.6277	0.5682
25	sherlock	0.5855	0.5091	0.5025
26	B4_toku	0.434	0.4751	0.4144
27	Hawk	0.2679	0.0561	0.0642
28	Fractal AIML	0.0148	0.0152	0.015

A comparison of the most accurate model produced during this research project (highlighted in **bold**) against those submitted during the data challenge.

4.2 Findings

Figure 1. Single Model Progression

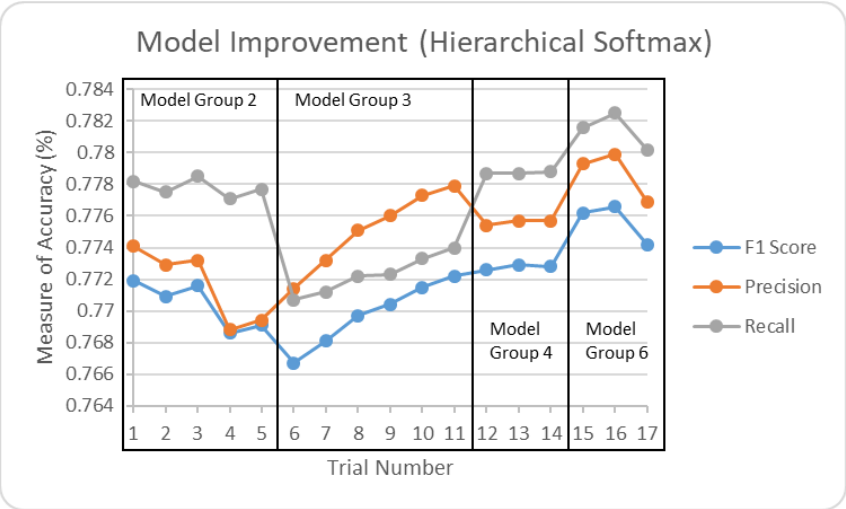

Figure 1 above shows a breakdown of model progression by Model Group as described in Table 2. This graph only includes models using the hierarchical softmax loss function as these were the models used to establish model improvement (as stated previously, it has been observed experimentally that model improvements using hierarchical softmax are almost always present in the same model using softmax). Group 5 was a test group to establish the effectiveness of sub-labelling, as such it is omitted from this and other figures, however results for this group can be seen in Table 2.

Figure 2. Single Model Performance by Epochs

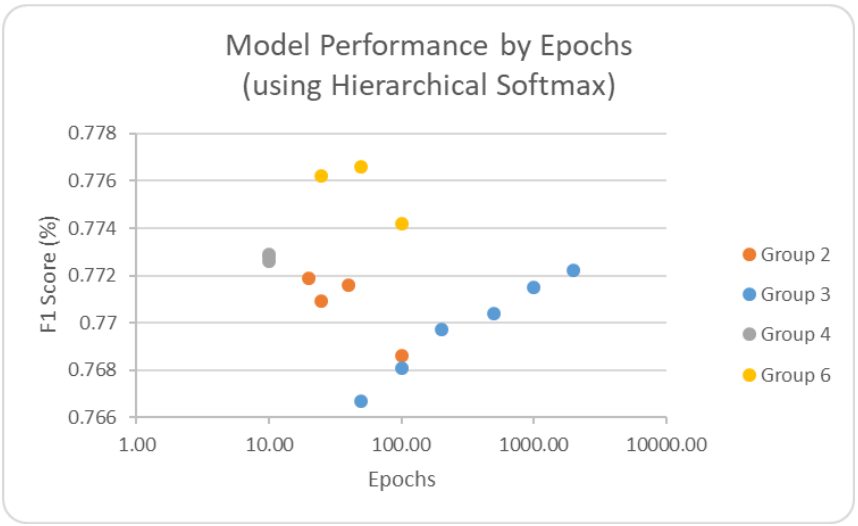

Figure 2 above shows the effects of changing the number of epochs during the 4 main single model trial groups (derived from table 2). Note that the purpose of group 4 was to view the effects in changing dimensions rather than epochs and as such creates a vertical line of points on this plot. There is an easily observable improvement in model performance found in group 3, which is likely why the fastText team involved in the challenge predominantly used a very high number of epochs to improve model performance; citing 3000 epochs in their final model (Lin, 2018).

Figure 3. SIGIR 2018 Rakuten Data Challenge Model Comparison

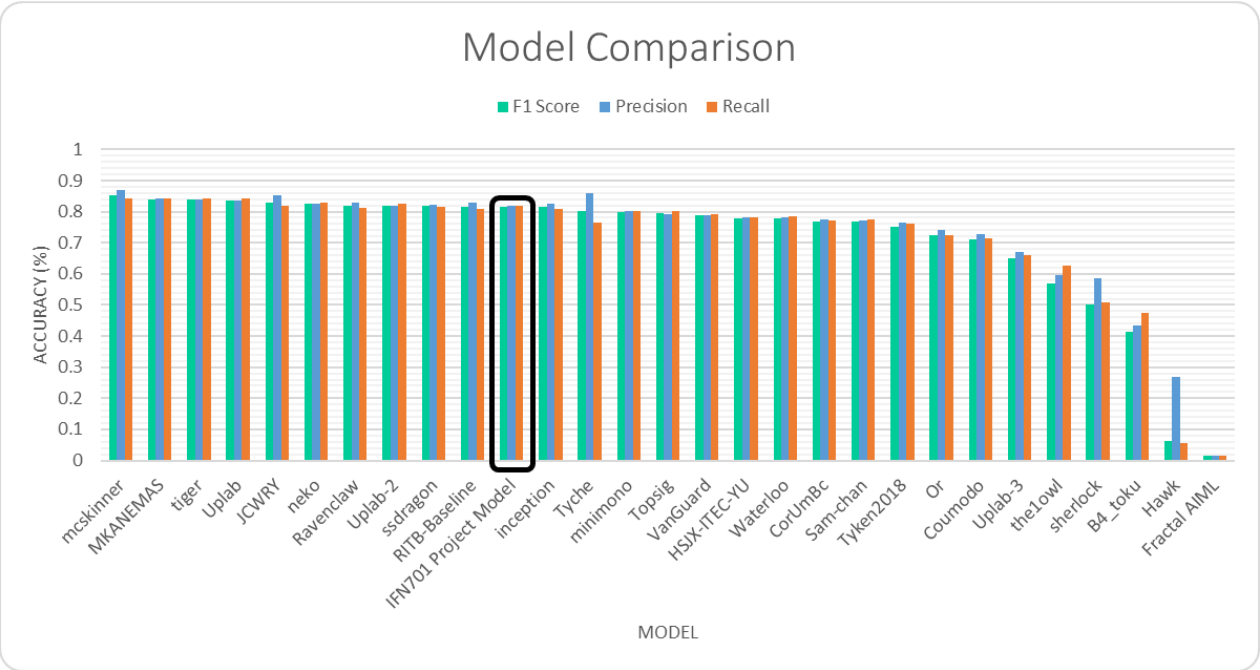

Figure 3 above shows a comparison of accuracy between the best model produced during this project (outlined in black) against the 28 models submitted to the final test of the data challenge. Note that our model did not outperform the RITB-Baseline (competition fastText model), there are several reasons for this which will be explained in more detail in the outcomes section below.

4.3 Outcomes

4.3.1 Data Pre-Processing Pipeline

Various pre-processing techniques were used during this project as outlined in Section 3 – Project Methodology, the majority of data cleaning for a given model or ensemble models was achieved using bash scripting. The reasoning for this is the speed of awk, sed and grep commands relative to implementing a C# executable to handle this process.

The cleaning script is run over the base data set producing a new processed copy which is then subdivided and used for training and testing of a given model. In contrast to this; term stemming, ensemble comparison and sub-labelling are achieved using executables written in C# in order to make the code easier to read (avoiding large single line statements in the bash cleaning script). Appendix 2 shows an example of the ensemble code; this code will rarely need to be changed after implementation and can be quite verbose when written as a bash file, as such it was deemed appropriate to write this file in C#.

The most effective data pre-processing step was the normalisation of digits to 0 i.e. where any digits found in the product title are replaced with 0. Note that labels for each product, which are almost entirely digits, are excluded from this process as this would drastically reduce the number of labels and remove most of the resolution in the results (labels with the same number of digits and format would be considered the same label such as “92>109” and “92>168” where both labels would be processed to “00>000”).

4.3.2 Model Group 1 Description

Being the initial testing phase, the main objective of model group 1 was to establish the effects key parameters such as epochs, buckets, n-grams and dimensions as well as loss functions in order to develop a better understand of each parameters impact on model performance. Minimal pre-processing is implemented at this point; only having removed punctuation and changing characters to lowercase (these steps would prove to be detrimental to model performance, however this effect will be discussed further in later sections) as well as replacing all digits with 0. Models in this group were evaluated using fastText’s inbuilt testing features which returned precision and recall values, however only precision was recorded during this model due to the “rough” nature of initial testing.

4.3.3 Model Group 2 Description

Group 2 is very similar to group 1, the main difference being that the python evaluation script used in the data challenge was now being used to evaluate model performance in order to remove as many differences between the conditions presented to data challenge teams and those of our own models. One new advantage is that this script also calculates F1 Scores, allowing for a better measure of model performance. Group 2 is where some of the less successful modelling techniques were trialled, including using a porter stemmer to normalize terms to their “stems”.

4.3.4 Model Group 3 Description

The purpose of group 3 is to attempt to mimic the parameters used by the RITB-Baseline team during the Rakuten Data Challenge, epochs were increased in each trial in order to see the effects on model performance. The linear relationship observed in Figure 2 (group 3 in blue) demonstrates that model performance may peak higher than even the 3000 epochs used by the competition team.

4.3.5 Model Group 4 Description

By this point, it is realised that punctuation and case sensitivity play an important roll in model performance, as such these pre-processing steps are removed. An additional observation is that model performance is as high, if not higher, than the previous group with only 10 epochs and less dimensions. As such this model group focusses on finding the plateau point for dimensions given 10 epochs.

4.3.6 Model Group 5 Description

This model group is a short experimentation on the effects of the implemented sub-labelling script. Results in table 2 show that sub-labelling was unfortunately an ineffective method of improving model performance for this dataset. The reason for this is stipulated to be that the importance of label order is already established during training through the various other techniques such as the use of term n-grams.

4.3.7 Model Group 6 Description

Using the best performance improvements identified over previous model group trials; group 6 utilized the aforementioned replacement of digits with 0 method along with the inclusion of punctuation. Dimensions are kept 200 as this is the ideal number identified in group 4 then seeks to find the plateau by varying the number of epochs.

4.3.8 Models Summary

Over 50 models were produced over to course of the semester with varying degrees of success, the most successful models were the ensemble models. Utilizing complimentary models as ensemble components proved to yield the highest precision, recall and f1 score (see Table 2 & Figure 1). The most successful ensemble model produced used a "base" model which did not use any pre-processing techniques and a model where punctuation was removed, all letters were changed to lowercase and digits were replaced with 0. In order to better understand why this model worked as well as it did, we will briefly re-explain how labels are chosen by the ensemble model.

In this implementation, ensemble models work by comparing the "certainty" of two or more models for a single product and taking the highest probability. For example, if model A predicts label x to be 92>68 with a probability of 0.9991 and model B predicts the same label to be 68>10>1024 with a probability of 0.9899 then model A's answer will be taken. In the event of a tie between 2 or more models, currently the first model given in the ensemble command line argument will be taken. In future projects, a better method for breaking ties should be implemented such as evaluating the certainty of each models "second-guess" or a similar methodology.

One possible reason for the effectiveness of the aforementioned ensemble model is that the "base" model was more competent at identifying products with a high proportion of non-alphanumeric symbols such as *"Gelco Multi-Flue 5/8"" Mesh Cap with 4"" Overhang – 28"" x 53"" x 10"""*, in which 14 of the 59 characters present would be removed in the "zeros" model (a significant portion when working with short text classification). On the other hand, the "zeros" model was likely more competent at identifying products with an excess of alpha numeric characters such as *Unique Bargains Cabinet Drawer Wooden Round Pull Knob Black 34mm Diameter 15pcs*, which would not be altered at all by the "base" model and may be grouped differently to very similar products such as a cabinet with different dimensions (diameter) or number of pieces (pcs).

4.3.9 Standard Research Project Deliverables

While this is already discussed in Section 1.4 – Scope & Deliverables, the standard deliverables for this research project include the Project Proposal, Proposal Presentation, Meeting Minutes, Final Presentation and this report. In all cases these deliverables have been presented through the relevant channels such as assignment minder, physical presentation or otherwise. Several weeks do not have a corresponding meeting minutes as meetings were cancelled for various reasons, usually to do with up-coming assessment deadlines. These meetings were substituted by regular communication with supervisors via email or phone in order to make sure the content of the project remains within scope.

5. Discussion

5.1 Analysis of Findings

As hypothesised, fastText yielded similar accuracy to conventional state-of-the-art text classification methods, this is exemplified by Figure 3 comparing this project's results to those of other models using the same data. It should also be noted that the data used for this project is only a sub-section of the data used during the Rakuten data challenge i.e. the Rakuten training set has been subdivided into a training and test set due to the absence of labels in the provided test dataset (which was evaluated via online submission during the competition and thus could not be retrieved for this project).

While one of the main points emphasized by Figure 3 is the improvement in accuracy over the course of the project, it is interesting to note that precision and recall "switched" in magnitude in Model Group 3. The major difference between Group 3 and the other groups is an increased number of epochs and dimensions. The purpose of group 3 was to mimic the parameters set by the fastText team during the SIGIR 2018 Rakuten Data Challenge (the RITB Baseline as shown in Table 5). The reversal in magnitude observed is likely due to the change in epochs as seen in table 2, as the majority of other parameters are similar in type and/or magnitude to those in other groups.

Figure 2 describes a near linear relationship between F1 score and epochs for Group 3 (mimicking the RITB Baseline parameters), it is likely that this setup will plateau given enough epochs. The reason a near linear relationship can be seen is most probably due to the exceptionally high number of dimensions, as such there is a need for more epochs to fully take advantage of these 300 dimensions. In contrast to this observation, Group 6 appears to plateau between 50 and 100 epochs, the purpose of this group was to test the improvement in accuracy with punctuation relative to Group 4 which used a static value of 10 epochs. Further testing would be needed to establish this relationship at higher and lower epoch values.

5.2 Strengths & Limitations

One of the key advantages of this project is the significance of the type of data being examined. As described in section 2 – Literature Review of Previous Work, the importance of short text classification has risen in recent times due to the prevalence of short text datasets. From online chat services and Twitter to company review websites (Glassdoor, 2018) and document titles, the importance of short text classification necessitates research such as this to further understand the capabilities of traditional modelling techniques and the requirements of future ones.

Another strength lies in the modelling application itself. The simplicity of fastText models, a linear classifier which utilize only one hidden layer, is best described by their ability to scale to very large corpuses while maintaining "state-of-the-art performance" when compared to traditional modelling techniques (Joulin, 2016). As explained in section 5.1 above, evidence of this is shown in Figure 3. It should be noted that in some circumstances, especially where relationships between variables are non-linear, linear classifiers such as fastText can be outperformed by more complex models. However, in the case of the Rakuten dataset, fastText is similarly capable due to the use of one input variable and one target variable (Product Name and Label respectively).

As mentioned in 2.2.4, two of the most prevalent limitations to the progression of this project were the capabilities of the laptop running the majority of model and experience using QUT's HPC Lyra. Both limitations negatively affected model progression by extending the time between testing, training and modifying model parameters, especially in cases with large values for epochs, dimensions and using the softmax loss function.

One of the more glaring limitations which pertains to model training and testing is the size of the dataset; as previously mentioned, a labelled version of the test dataset could not be obtained for SIGIR prior to the completion of this report, as such the training dataset was subdivided into training and test data. The implication of this limitation is that models produced during this project are working with a smaller dataset than those reported in the Rakuten data challenge. It is highly possible that, given the full dataset, our models may have performed better comparatively. The greatest example of this possibility is that the difference in F1 score between our "best" model and the RITB-Baseline (the competition fastText model) is 0.01% (see Figure 3), meaning it is likely to have placed in the top 10 of those submitted to the final test.

5.3 Recommendations for the Future

Several improvements can be made in future projects to streamline model training and testing as well as implementation of pre-processing and so on. Proper understanding of how to connect to and use HPC Lyra will improve training and testing times dramatically which will allow more model parameters to be trialled in addition to larger parameter values especially with regards to epochs and dimensions.

Reimplementation of some scripts used over the course of this project will likely improve readability and possibly performance; my recommendation would be to reimplement all scripts into scripts of a single language, either bash or python would likely be most appropriate given the nature of the tasks required. This will have the added benefit of making all components more readily compatible with HPC Lyra by removing C# executables and other windows related components, further reducing the effects of this limitation as explained in section 5.2.

To further prove the effectiveness of fastText on text classification tasks, it is necessary to perform other forms of text classification with objectives such as sentiment analysis and spam detection. New, larger datasets will further test the performance of fastText while also introducing the necessity for new pre-processing techniques. As an example; sentiment analysis of Twitter posts may benefit from stemming terms due to the natural language used as oppose to the product names found in the Rakuten dataset. This would be an especially interesting comparison as both exercises work with short-text classification.

6. Conclusion

The aim of this research project was to determine how effectively fastText performed against other text classification methodologies using the Rakuten Data Challenge results for comparison. To this effect, our results have given evidence in favour of fastText; showing that it is as effective if not more so than conventional text classification techniques (see Figure 3).

In summary, the findings discussed in this report provide further evidence of the validity of fastText in terms of its performance characteristics in comparison to conventional state-of-the-art text classification methodologies. As explained in section 5.1 – Analysis of Findings, experimental results acquired from model analysis during this research project have shown that the best model produced during this project ranks 11th out of 28 on the SIGIR 2018 Rakuten Data Challenge leader board when ordered by F1 Scores; indicating similar performance to models submitted during the data challenge. This provides evidence in favour of our original hypothesis; fastText is capable of similar performance to conventional state-of-the-art text classification methods.

The significance of fastText and short-text classification underpin the significance of this report; these results are especially interesting as the utilization of fastText in short-text classification scenarios is a new area of interest which will only grow with the growth of the datasets it uses (Twitter, instant messaging and so on). Further research is necessary to understand not only the significance of these areas, but also the best methods for modelling data in these areas and future implementations of these methods.

With regards to the limitations outlined in section 5.2, these issues are easily manageable in future projects if the recommendations outlined in section 5.3 are adhered to; this is especially true for those limits pertaining to proper use of HPC systems as well as laptop limitations. It may be difficult to acquire a full copy of the Rakuten dataset for testing purposes, however a more obvious solution is to pick a new dataset which will also necessitate writing new, more streamlined scripts to run fastText itself as well as handle pre-processing. This will give the added benefit of an improvement in readability assuming a single scripting or programming language is used and well understood.

The key aspects of this report which are relevant to future studies are the implementation of more uniform, readable code for pre-processing and modelling techniques (as described in section 5 – Discussion), acquiring a better understanding of High Performance Computer systems available and the development of a “plan-of-attack” which should outline the objectives of each set of models and the number of trials for each. It is not always necessary to follow such a document strictly, however it would make reporting on results easier by guaranteeing what results will need to be analysed and allow for planning of diagrams prior to results being obtained.

Appendix

Appendix 1: Reflection

I have thoroughly enjoyed my experience working on this project; its various challenges forced me to adapt quickly in an everchanging work environment, especially when new pre-processing methods were developed.

Initially, the majority of my pre-processing was achieved using bash scripting in order to make use of the powerful tools provided by awk, grep, sed and other similar languages and scripting tools. My previous experience in this area from IFN509 – Data Manipulation has given me the skills required to effectively write bash scripts for most data manipulation tasks, as such I believe I performed best at this aspect of the pre-processing phase.

I believe that mixing several languages together for convenience has made it difficult to debug my work when a component isn't functioning correctly as I end up looking through not only bash files but also C# solutions used to build executables during the debugging process. While this was certainly an interesting learning experience, in hind sight, I would have been far better off using a single language such as python which incorporates the structure of a programming language while still maintaining some of the conveniences of a scripting language.

During the research and development process, starting in week 3, I would always make sure to discuss issues I was having with developing models or understanding methods such as ensemble modelling or 10-fold cross validation with my supervisor. This was my most common method of problem solving beyond the literature review as their background in text classification would always yield a useful explanation or ideas for a solution without being too overt. This methodology allowed me to organically develop my own solutions to project issues without hindering the progress of the project as a whole.

One of the hardest parts of this project has been continually testing new pre-processing steps, especially sub-labelling which I originally implemented incorrectly due not understanding the importance of order in sub-labels. This issue was rectified during discussions with my supervisor who further explained the purpose of including these sub-labels and the possible benefits. Implementation was achieved using C# in order to increase the "readability" of this code as, given my current experience, it would be time consuming to create this file using purely bash scripting. It would also likely be difficult to read in a bash script compared to C#.

In terms of personal development, I believe I need to become a more confident programmer such that implementation is not as daunting to me in future projects. Having a better understanding of python programming would be especially useful given the nature of this project. In future projects I would like to make better use of my time by understanding python well enough to overcome the issues outlined in this report such as debugging and code cleanliness.

One of the most important concepts garnered from my research and practical development during this project is a better understanding of the behaviour of models, which is to say that increasing the resources available to a model will not always improve model performance for various reasons including overfitting. Many examples of this can be found throughout the results section of this report as attempts are made to find “plateau” points for various models when variability is constrained to one parameter such as epochs or dimensions.

This research project taught me the importance of performing a thorough literature review, as the majority of beneficial pre-processing techniques were found in papers written about either fastText or the Rakuten data challenge. Having a good understanding of the current state-of-the-art in this field has helped me progress my models further than I would have been able had I not reviewed the literature. It would have been very difficult if not impossible to substantiate any claims or propose the significance of my research project had I not performed this process.

The field of short-text classification is very interesting to me due to the limitations of the datasets i.e. the small length of fields leaves little room for the derivation of dimensions. As such my ideal next project would continue in this field while perhaps tackling a more complex form of classification such as sentiment analysis or some form of natural language processing (NLP). This would give me a chance to further improve my modelling capabilities, my understanding of HPC Lyra and my programming capabilities.

Appendix 2 – Ensemble Modelling Code

```
List<string> compareModels(List<string[]> pFiles) {
    var ensembleResults = new List<string>();
    //check all files are of the same length
    //(may need to add a "- 1" to the pFiles count otherwise there will be a null reference)
    for (int i = 0; i < pFiles.Count - 1; i++) {
        if (pFiles[i].Length != pFiles[i + 1].Length) {
            Console.WriteLine("Files " + i + " & " + (i+1) + " are of different lengths (" + pFiles[i].Length + " & " + pFiles[i+1].Length + " lines respectively).");
            Console.WriteLine("All files must have an equal number of lines to ensure correct comparisons. Please fix this and rerun the application.");
            return ensembleResults;
        }
    }

    //for each line i
    for (int i = 0; i < pFiles[0].Length; i++) {
        string bestLabel = pFiles[0][i];
        //for each file j
        for (int j = 0; j < pFiles.Count - 1 ; j++) {
            //parse the probability as a double then compare it to the next probability; if the next one is higher, this is the new best label!
            double p1 = Double.Parse(pFiles[j][i].Substring(pFiles[j][i].LastIndexOf(' ') + 1));
            double p2 = Double.Parse(pFiles[j+1][i].Substring(pFiles[j+1][i].LastIndexOf(' ') + 1));
            if (p2 > p1) {
                bestLabel = pFiles[j+1][i];
            }
        }
        //remove the probability section of each line before adding it to the collection
        ensembleResults.Add(bestLabel.Substring(0, bestLabel.IndexOf(' ')));
    }
    return ensembleResults;
}
```

Bibliography

Geva, S. (2018). *IFN600 Understanding Research (Literature Analysis Criteria Sheet)*. Retrieved from https://blackboard.qut.edu.au/webapps/blackboard/content/listContent.jsp?course_id=_135158_1&content_id=_6569054_1&mode=reset

Glassdoor Job Search | Find the job that fits your life. (2018). Retrieved from https://www.glassdoor.com.au/index.htm

Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2018). *fastText*. Available at: https://fasttext.cc/ [Accessed 19 Aug. 2018].

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). *Bag of Tricks for Efficient Text Classification*. Retrieved from https://arxiv.org/pdf/1607.01759.pdf

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).

Lin, Y. (2018). *SIGIR eCom*. [online] Sigir-ecom.github.io. Available at: https://sigir-ecom.github.io/data-task.html [Accessed 12 Aug. 2018].

Lin, Y., Das, P., & Datta, A. (2018). *Overview of the SIGIR 2018 eCom Rakuten Data Challenge*. Retrieved from https://sigir-ecom.github.io/ecom18DCPapers/ecom18DC_paper_13.pdf

Mesnil, G., Mikolov, T., Ranzato, M. A., & Bengio, Y. (2014). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*. Retrieved from https://arxiv.org/abs/1412.5335

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781.

Ng, A. (2018). *Machine Learning Yearning*. deeplearning.ai.

Porter Stemming Algorithm. (2018). Retrieved from https://tartarus.org/martin/PorterStemmer/

Song, G., Ye, Y., Du, X., Huang, X., & Bie, S. (2014). *Short text classification: A survey*. Journal of Multimedia, 9(5), 635.

Thangaraj, M., & Sivakami, M. (2018). *TEXT CLASSIFICATION TECHNIQUES: A LITERATURE REVIEW*. Interdisciplinary Journal of Information, Knowledge & Management, 13.

Twitter. (2018). *Twitter. It's what's happening.* Retrieved from https://Twitter.com/

YCombinator. (2017). Hacker News. Retrieved from https://news.ycombinator.com/item?id=14307062

Zhan, J., & Dahal, B. (2017). Using deep learning for short text understanding. Journal of Big Data, 4(1), 1–15. https://doi.org/10.1186/s40537-017-0095-2