CA4022 Assignment 3 Final Report

Comparative analysis of Yahoo topic classification using Machine Learning and TF-IDF embeddings

Joseph Oluwasanya
joseph.oluwasanya2@mail.dcu.ie
Student ID: 19329261

Diarmuid Brady
diarmuid.brady35@mail.dcu.ie
Student ID: 19479166

December 17, 2022

1 Introduction

With a suitable representation of our data for a given task, even simple machine learning algorithms can perform on par with more complex approaches. A well-known example of this is Razavian et al. (2014), where Convolutional Neural Networks (CNN) were used to produce embeddings for several pattern recognition tasks, including scene recognition, attribute detection, and image retrieval. A Support Vector Machine using a linear kernel, and trained with these derived embeddings showed superior results at all of these tasks. In this project, we analyse the performance of two machine learning algorithms at identifying topics in the Yahoo Answers Topic Classification dataset (source). Our hypothesis was that given a suitable representation, the SVM with a linear kernel will perform at least on par with more complex machine learning algorithms. The benchmark model for our analysis was Random Forest. We transformed the text input data into numerical data using TF-IDF, We carried out this analysis using PySpark, leveraging the SparkSQL capabilities for data processing and initial data size reduction, and SparkML for setting up modelling pipelines, evaluating models, and performing principal component analysis. We ran Spark on Hadoop, with our local machines acting as standalone single-node clusters.

2 Data Source

We used the Yahoo Answers Topic classification data for this analysis. The dataset was constructed by Xiang Zhang (xiang.zhang@nyu.edu), and is used as a text classification benchmark in Zhang et al. (2016). It has four variables: *Class Index, Question Title, Question Content, and Best Answer*. The class index refers to a number label, which has a corresponding topic name. If a Question could belong to several topics, the topic that best matches the question is selected. The Class Index is the target variable for our predictive models, and corresponds to the topics as indicated in Table 1.

3 Data Preprocessing & Cleaning

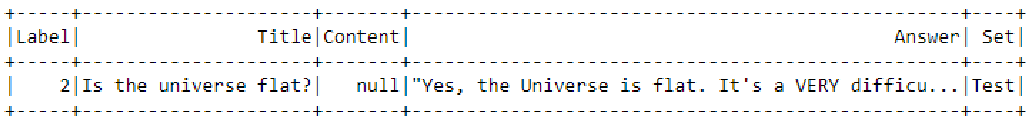
Firstly, we reduced the data size. There are 1.4m in the training set and 60k in the test set of the original data. We used stratified sampling to reduce this, but maintain the same proportions of the different classes as are present in the original data. Although, note that the data is balanced, so a random sample should maintain this balance roughly. Each topic makes up 10% of the dataset. Our resulting train and test sets were reduced to a quarter of the original size, so we now had 350k training and 15k test examples. We also added the Set Column which labels each example as "Train" or "Test"

After a brief inspection of the data, we saw that there were empty fields in the Content and Answer fields. We assumed that the empty Answer fields occur for questions that didn't receive

Label	Topic
1	Society & Culture
2	Science & Mathematics
3	Health
4	Education & Reference
5	Computers & Internet
6	Sports
7	Business & Finance
8	Entertainment & Music
9	Family & Relationships
10	Politics & Government

Table 1: Classes in Yahoo dataset

any answers. As for empty Content, this was often down to the question being outlined in the Title section already. The question title of the example below is ”Is the Universe flat?”. With no added context in the Content field. We replaced these null values with empty strings such that the next preprocessing step works properly.



```
+-----+---------------------+-------+--------------------------------------------------+----+
|Label|                Title|Content|                                            Answer| Set|
+-----+---------------------+-------+--------------------------------------------------+----+
|    2|Is the universe flat?|   null|"Yes, the Universe is flat. It's a VERY difficu...|Test|
+-----+---------------------+-------+--------------------------------------------------+----+
```

Figure 1: Example with missing Content field

We wanted the Title, Content, and Answer fields to be treated as a single document when we apply the TF-IDF transformation. To do this, we concatenated all the text fields into a Document variable. The remaining preprocessing steps were chained together using a Spark ML Pipeline. Next, we tokenized the documents such that each token is a word, which was done using *RegexTokenizer*. The search pattern used was “\W”, so the tokenizer splits on any non-word character in the document. The output was passed to a stop word remover, which by default uses an English stop word list to filter the tokens. Finally, *StringIndexer* was used to map the labels to an ML label indices column, such that the Spark ML models can recognise the target.

4 Experiment Design & Results

As mentioned in Section 1, we compared the performance of Linear SVMs and Random Forest at classifying the Yahoo Answer topics. Initially, the TF-IDF vector had about $260,000$ dimensions so we had to limit the vector sizes to run our experiments. We tried several vector sizes which were recorded in our experimental setup. The choice of the vector size range was based on RAM constraints. Also, note that an SVM is a binary classifier, so for this multiclass classification task we had to use the One-Vs-Rest strategy. That is, a separate SVM is trained for each class, and the results are combined to form a single classifier. And the only performance measure we used here is accuracy as the classes are balanced. Our experiment design was as follows:

For **X** in [1,000, 2,000, ..., 6,000]

1. Apply the TF-IDF transformation to each document, limiting the vector size to **X**.
2. Fit One-Versus-Rest Linear SVM classifier on the TF-IDF-derived sparse vectors.
3. Predict the topics of examples in the test set using the model.

4. Record the accuracy and size of the TF-IDF vector used.

This same experiment was repeated using Random Forest and SVM, both with default hyperparameters. The results were as follows:

	Vector Length (thousands)					
Algorithm	1	2	3	4	5	6
SVM	0.509	0.572	0.601	0.617	0.627	**0.634**
RF	0.324	0.357	0.374	0.366	0.362	0.388

Table 2: Model accuracy on Test examples

As can be seen here, the SVM algorithm consistently outperformed Random Forest at classifying yahoo questions into their respective topics when we use TF-IDF document vector representations as the training data. The accuracies here don't appear very high, but note that there are 10 balanced classes in the dataset so an Accuracy score of 0.10 suggests that the model is exhibiting random guessing behaviour.

One of the advantages of SVM is its superior performance when working with high-dimensional sparse feature vectors. For a feature vector in $\mathbb{R}^n$, SVM finds a hyperplane in $n-1$ dimensions to try and maximise the margin between the two classes. As the dimensions increase, the classes generally become more easily separable using this strategy. It has been shown that SVM performance is not impacted by the curse of dimensionality Salimi et al. (2018). Whereas the performance of most other machine learning algorithms, including Random Forest, will suffer as the dimensions increase over certain thresholds.

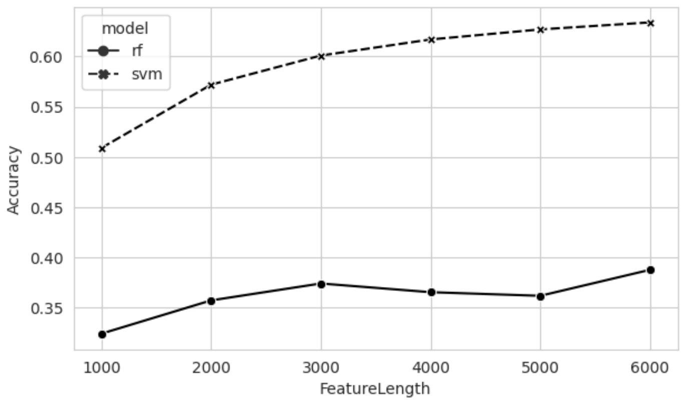


Figure 2: Line plot of Random Forest and SVM accuracy

Following this, we ran a stress test on the linear SVM increasing the TF-IDF vector size even further and found that accuracy stopped increasing at around 64% and Vector length 10,000. (See Figure 3)

5 Related Work

Linear SVMs have many applications in pattern recognition, including image classification and fraud detection, and even medical diagnosis. Related works include Razavian et al. (2014) and Salimi et al. (2018), which motivate the use of the methodology we proposed here.

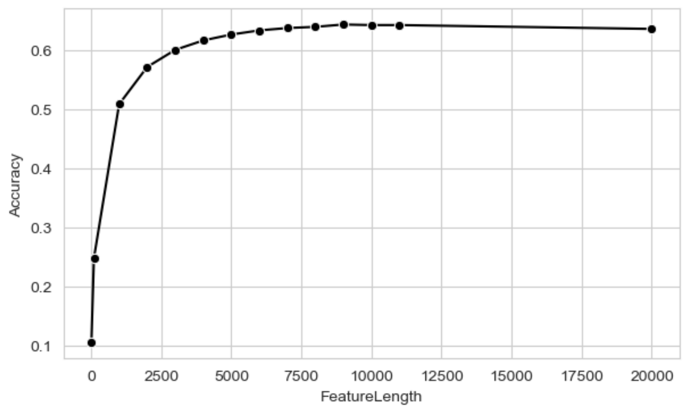


Figure 3: Line plot of SVM Stress Test Accuracy

6 Challenges & Lessons Learned

We faced several challenges throughout the course of this project. Initially, we planned on using BERT embeddings to compare with the TF-IDF performance but didn't get this working correctly. The Bert Embedding model was available through Spark NLP, which is separate from Spark ML and Mlib, so separate jar files were required. Once this was working, the notebooks using Spark NLP would get stuck after creating the BERT embeddings. There was no error message, and we didn't find a workaround in sufficient time, so we decided to redefine the goal of our project. We believe this worked out well as it allowed us to form our new hypothesis. Rather than comparing TF-IDF and BERT embeddings, we used only the TF-IDF embeddings and looked into the suitability of this document representation for training a simple linear classifier.

We also planned on using Hive for data reduction but ran into delimiter issues when moving the data into spark. Using Spark SQL on the PySpark client we were able to do the data reduction more easily, so we went with this approach in the end. Building the models sequentially in the experiment setup caused java.lang.OutOfMemoryError: Java heap space Errors. This was resolved by changing the memory allocation configuration of Spark. From doing this project we gained practical experience with technologies which facilitate big data processing and analytics; Namely Hadoop and Spark.

7 Responsibility Statement

Text Preprocessing, Modelling Pipeline, and SVM Experiment Setup by Diarmuid Brady.
Data size reduction, Random Forest Experiment Setup, And Report Writeup by Joseph Oluwasanya.

8 Response to peer feedback

Overall, we believe the feedback was constructive, practical and precise. The feedback has illuminated areas of potential improvement and recognized the project's strengths. Because of the feedback, we were more careful to make our steps clear and unambiguous. Although we respect the concerns of using SVM for multiclass classification, we maintain our position that the One-Versus-Rest Linear SVM approach was a worthwhile method to try out, independent of the results we obtained.

Links

- Data source **here**

- Source code hosted on GitHub **here**. The recording is not comprehensive as runtimes are too long, but the code is all hosted there.

References

Razavian, A. S., Azizpour, H., Sullivan, J. & Carlsson, S. (2014), CNN Features Off-the-Shelf: An Astounding Baseline for Recognition, *in* '2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops', pp. 512–519. ISSN: 2160-7516.

Salimi, A., Ziaii, M., Amiri, A., Hosseinjani Zadeh, M., Karimpouli, S. & Moradkhani, M. (2018), 'Using a Feature Subset Selection method and Support Vector Machine to address curse of dimensionality and redundancy in Hyperion hyperspectral data classification', *The Egyptian Journal of Remote Sensing and Space Science* **21**(1), 27–36.
URL: *https://www.sciencedirect.com/science/article/pii/S1110982317300571*

Zhang, X., Zhao, J. & LeCun, Y. (2016), 'Character-level Convolutional Networks for Text Classification'. arXiv:1509.01626 [cs].
URL: *http://arxiv.org/abs/1509.01626*