Text Classification, a General Approach

Chun Hei Fok, Julio Aguilar Jimenez, David Guest, Jo Houghton, and Sofie Debloudts

***Abstract*—A wide range of classifiers are used on three different data sets ('spam', 'toxic' and 'movie'). Results are reported, using both preprocessed and raw text, with the use of three different word encoders (Tf-Idf, word2vec, GloVe) on the datasets. The relative performance of different encoders, processing, and balancing is discussed. A hybrid CNN-RNN model is implemented and and tested alongside more traditional approaches (eg. SVM, NB, etc.) The evidence points to simpler approaches being more generally applicable and performing better, as although the hybrid model has the best performance in two of the data sets, it underperforms in the third.**

I. Introduction

Text classification is ubiquitous in real-world problems. Spam is a continuing problem besetting email servers, and while a number of different solutions have been posited, most rely on a good spam filter. Toxic language such as hostility, insults and threats present a barrier that discourages, limit and shut down user comments, presenting a big problem for platforms that seek to improve user communication experiences, and automated analysis of user sentiment is important for a range of problems, from analysing product quality, to predicting the stock market. This paper aims to offer a comparison of a range of algorithms on several text classification problems.

II. Text Classification Tasks

A. Toxic

The toxic comment dataset was released with the goal of identifying toxicity of online comments. The data consists of 159,571 comments with binary labels indicating toxic and non-toxic [4]. The data was used with multi-label classification in a Kaggle competition, but for this report only binary toxic/not toxic predictions have been sought.

B. Spam

The spam email dataset contains 33,713 emails from the Enron Corporation. It was obtained by the Federal Energy Regulatory Commission during its investigation of Enron's collapse [6]. The set was used in a Kaggle competition as well as in this work, for binary classification (spam/ham).

C. Movie

This dataset consists of 156,060 excerpts from movie reviews and their respective sentiment labels, ranging from 0 (negative) to 4 (positive). The data was provided by Rotten Tomatoes, a movie review website, and collated for a past Kaggle competition. It notable that this dataset has repeated phrases from the same review, with different sentiment labels given by hand [5].

III. Classifiers

A. Logistic Regression

Logistic regression applies maximum likelihood estimation after transforming the dependent variable into a logit variable, which is the log odds of dependent variable occurrence, with respect to independent variables. In this way, logistic regression estimates the probability of a certain event occurring.

$$log(odds) = log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 * X_1 ... \beta_n * X_n \quad (1)$$

Logistic regression does not assume linear dependence, normal distribution or homogeneity in the variables.

B. LDA and QDA

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are a generalisation of *Fisher's* linear discriminant, they find a combination of features that characterise or separate two or more classes of objects. The resulting combination may be used as a linear classifier or for dimensionality reduction before later classification.

LDA and QDA assume that the measurements have a normal distribution, the predictions and the probability density are calculated by Equations (2), (3).

$$P(y = k|X) = \frac{P(X|y = k)P(y = k)}{\sum_l P(X|y = l) \cdot P(y = l)} \quad (2)$$

$$p(X|y = k) = \frac{1}{(2\pi)^n |\sum_k|^{1/2}} exp\left(-\frac{1}{2}\sigma^t \sum_k^{-1} \sigma\right) \quad (3)$$

Where $\sigma = (X - \mu_k)$. For LDA, the Gaussians for each class are assumed to share the same covariance matrix, and have different covariance matrices for QDA.

Limitations of logistic regression include:

- *Well separable classes*. Logistic regression can become unstable when the classes are well separated.
- *Few examples*. Logistic regression can be unstable when there are few examples from which to estimate the parameters.
- *Two class problem*. Although it can be expanded, logistic regression is focused on a binary classification.

C. Random Forest

Random Forest (RF) is an ensemble of decision trees, it focuses on sampling both observations and variables of training data to develop independent decision trees and take majority voting for classification and averaging for regression problems respectively. In random forest, a few observations

and columns are selected to create uncorrelated individual trees.

Decisions of trees have low bias but very high variance error due to overfitting, however RF random sampling allows to reduce this error preserving a good trade-off with bias, the following considerations were used in the construction of the model [1].

- 2/3 of observations for each individual tree.
- Number of columns =$\sqrt{p},\ p = total\,columns$.
- Number of trees was chosen based on the results obtained.

D. Multinomial Naive Bayes

The Multinomial Naive Bayes (MNNB) classifier is a Bayes-theorem based classifier. It is commonly used in text classifications problems such as email-spam detection, document categorisation, explicit content detection, etc. Despite being outperformed by other learning models such as random forest, boosted trees, SVM, etc. MNNB outperforms others in using less computational resources such as CPU and memory as well as only requiring small amount of training data. Due to these advantages it also requires less training time. MNNB is also suitable when multiple occurrences of words matter in a problem. The classifier can be defined as:

$$c_{map} = arg\ max(P(c|d)) = arg\ max(P(c) \prod_{1 \leqslant k \leqslant n_d} (t_k|c))$$

Where t_k are the tokens of the text document, C is the set of classes.

E. SVM

Support Vector Machine (SVM) classifiers are based on the idea of building a hyperplane which divides the dataset into two or more classes. The SVM identify the hyperplane which maximises the distance between itself and the training data. Kernels as well as soft margins are employed when there is no hyperplane to separate the data points. The cost function of the learning algorithm can exist in the dual form:

$$\min_{\mathbf{w},\mathbf{b}} \frac{|w|^2}{2}\ subject\ to\ y_k(w^T x_k - b) \geq 1 \tag{4}$$

$$and \quad \max_{\alpha} \sum_{k=1}^{P} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{P} \alpha_k \alpha l y_k y_l {x_k}^T x_l \tag{5}$$

$$with \sum_{k=1}^{P} \alpha_k y_k = 0\ and\ \alpha_k \geq 0 \tag{6}$$

Where x_k is an n-dimensional vector, y_k is the respective class identifier, P is the number of vectors and w is the distance margin. Both forms can be solved via quadratic programming algorithms.

Some of the advantages of using an SVM are its accuracy and efficiency in small datasets, particularly those with large numbers of features, such as text data with large vectoriser spaces. However it is less effective on noisier datasets with overlapping classes.

F. ADA Boost

Boosting works in a sequential manner and does not involve bootstrap sampling; instead, each tree is fitted on a modified version of an original dataset and finally added up to create a strong classifier.

$$H(x) = \sum_t \rho_t h_t(x) \tag{7}$$

A classifier is fitted on the data and evaluate overall errors. The error used for calculating weight should be given for that classifier in the final additive model (α) evaluation as in Equations (8), (9).

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{8}$$

$$\alpha_m = log\left(\frac{1 - err_m}{err_m}\right) \tag{9}$$

Higher weights will be given to the model with fewer errors. Weights for each observation will be updated.

$$w_i < -w_i * exp[\alpha_m * I(y_i \neq G_m(x_i))],\ i = 1, 2, ..., N \tag{10}$$

Although Ada Boost is a powerful classifier, it can be sensitive to noisy data and outliers. The algorithm also does not currently support null rejection. The *base estimator* used in datasets was a Decision Tree Classifier and the number of *maximum estimators* was set to 10.

G. Extreme Gradient Boost

Extreme Gradient Boost (XGBoost) is an implementation of gradient boosted decision trees designed for speed and performance. The library has several advantages such as parallelisation of tree construction, cache optimisation, supports block structure, automatically handles missing data and has a continuous training to improve already trained models.

XGBoost works under the principle of boosting, which has the following elements:

- *Loss function*, varies depending on the type of problem (mean squared error for regression and logarithmic loss for classification).
- *Weak learner to make predictions*, decision trees are used as weak learners.
- *Additive model*, trees are added one at a time.

Using a weighted average model (higher importance given to better models that predict results with greater accuracy than others) the output is given by Equation (11).

$$Y = \alpha * F(x) + \beta * G(x) + \gamma * H(x) + error \tag{11}$$

The configuration used in this work is as follows:

- Tree method = exact greedy.
- Max depth of tree = 1...4.
- Subsample ratio of training instance = 0.8.
- Subsample ratio of columns = 0.8.

H. CNN-RNN Hybrid

The architecture used for this classifier is as follows:

- *Embedding input layer*. Receives a previously sequenced vector of size 100, keeps the top 10,000 words and outputs a vector length 32.
- *Convolutional layer*. The dimensionality of output space is 32, kernel size 3, uses "relu" activation function and preserves input size.
- *Max pooling layer*. 1D pooling layer with max pooling windows of 2.
- *LSTM layer*. 100 units, tanh activation function, dropout, and recurrent drop out of 0.2.
- *Dense output layer*. Sigmoid activation function.

The embedded layer transforms the positive integers into dense vectors of length 32. [2]

$$int(x) = \left[v_1, v_2, v_3, \ldots, v_{32}\right] \tag{12}$$

The 1-D convolutional layer performs 32 filters on a 3x3 window to be processed later, gradients of the filters are calculated by Equations (13), (14), where W, b, x, y, k, a and δ are the weights, bias, training data, labels, filter number, inputs of actual layer and error term respectively.

$$\nabla_{W_k^{(l)}} J(W, b; x, y) = \sum_{i=1}^{m} (a_i^{(l)}) * rot90(\delta_k^{(l+1)}, 2) \tag{13}$$

$$\nabla_{b_k^{(l)}} J(W, b; x, y) = \sum_{a,b} (\delta_k^{(l+1)})_{a,b} \tag{14}$$

The 1-D max pooling layer reduces the output dimensionality by half and provides input for the long short term memory network (LSTM), which is capable of learning long-term dependencies. The first step is to decide what information is to be thrown away from the cell state using a sigmoid layer called the"forget gate layer. A number is output between 0 and 1 as in Equation (15) using h_{t-1} and x_t.

Next, a sigmoid layer called the"input gate layer decides which values to update and a tanh layer creates a vector of new candidate values $\tilde{C}_t$, that can be added to the state (16).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{15}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + bc) \tag{16}$$

In order to update the old cell state, the old state is multiplied by f_t and new candidate values are added (17). The output will be based on our cell state, a sigmoid layer is run which decides what parts of the cell state will be output (18). For the last step in LSTM, we put the cell state through tanh and multiply it by the output of the sigmoid gate (19).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{17}$$

$$o_t = \sigma(W_o[h_{t-1},\, x_t] + b_o) \tag{18}$$

$$h_t = o_t * tanh(C_t) \tag{19}$$

The final layer is a dense fully connected layer or, in the case of multiclass problems, is composed of n dense layers. This is the most complex classifier used in this work.

IV. Preprocessing and Encoding

A. Pre-Processing

The following techniques were used to pre-process the text:

- *Remove standard punctuation*. Symbols like". , ;, :, &, /, (, etc.", were replaced with empty strings.
- *Tokenise* text into words based on space characters.
- *Lowercase* the words to avoid duplicates.
- *Stop words removal*. Frequent words like "the, on, and, which", were removed.
- *Threshold*. Words with length < 3 were removed.
- *Stemming*. Reduce words to the root to avoid duplicates.
- *Lemmatise*. Tag the words into 4 categories of nouns and 6 verbs, e.g. noun common singular, noun proper plural.

B. Tf-Idf

Term frequency-inverse document frequency is intended to reflect how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus. The parameters used are:

- Ignore terms that have absolute count lower than 2.
- Ngrams boundaries or range = $1 \leq n \leq 2$.
- 1,000 features ordered by term frequency.

The number of features was found to be the most crucial parameter, with the SVM classifier, the spam dataset and features of 20, 100, 1000, 4000 and 10000; the accuracy obtained was 78, 82, 93, 94 and 93 respectively.

C. Word2Vec

Word2vec was developed at Google by Tomas Mikolov, and subsequently analysed by Goldberg and Levy [7]. It involves a one hot encoding of the words used, then a two layer neural net, where a word is predicted using the surrounding words. This can be done using a 'continuous bag of words', which does not preserve word order, but is fast, or a 'skip-gram' model, which places much more emphasis on the words nearest the word in question. The 'skip-gram' model is slow, but generates a more subtle representation of the words, better for rarer words. In this work, 'skip-gram' was used, as it was hoped that the more subtle encoding would preserve the useful information in the word order better. For example, from the raw, unpreprocessed toxic dataset, these words were found to be nearest to 'im' in vector space: 'I'm'; 'i'm'; 'Im'; 'its'; 'am'; 'it's'. This can be used to generate a 't-SNE' plot of the words in a dimensionally reduced vector space (Fig. 1).

To produce an encoding for whole phrases, a paragraph2vec approach was adopted to produce an embedded representation for whole phrases in the training set [13].

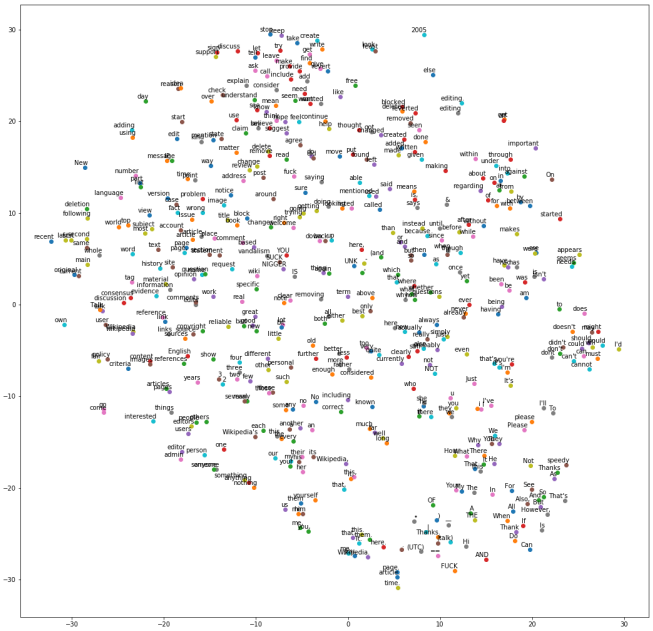

Fig. 1. Plot of word2vec encoding of words, using the unpreprocessed 'toxic' dataset, in 2d (using PCA to reduce the dimensionality)

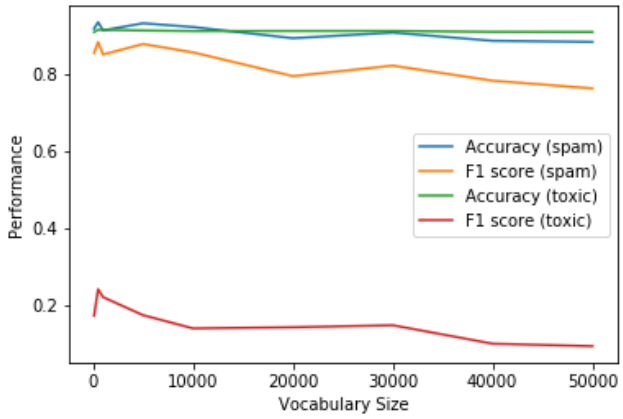

Fig. 2. Graph to show variation of classification accuracy vs vocabulary size.

a) Vocabulary Size: Using word2vec, on unpreprocessed data, the random forest classifier was used to classify the 'spam' dataset. Vocabulary size was varied, from 50 to 50000, and the results plotted below. It appears there is no advantage in increasing vocabulary sizes above 5000, , though there is a cost in terms of decreased f1 score. Interestingly, a vocabulary size of only 100 was able to allow 90% accuracy on the spam dataset, most of those words are punctuation, and stopwords, but also include words such as 'thanks', 'enron', and 'farmer', indicative of the dataset they were taken from. For better generalisation, a larger vocabulary should be necessary, but in the case of email for a particular person, it may be possible that an algorithm could learn quite effective simple rules to provide a personalised supplement to good spam detection.

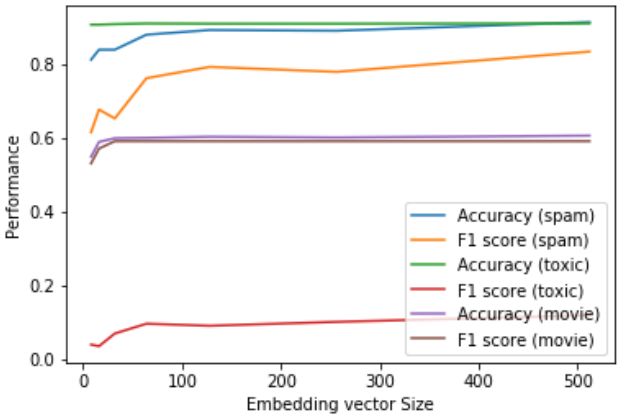

Fig. 3. Graph to show variation of classification performance using accuracy and f1 score vs embedded vector size

b) Embedded Vector Size: In our implementation of word2vec, the size of the feature vector output can be varied to change the number of features available to the classifier. Vector size was varied for each of the three datasets, using the unpreprocessed data and a random forest classifier. A graph of this classification accuracy can be seen in Fig. 3.

D. GloVe

One of our hypotheses was that word embeddings trained within the data sets would lead to superior accuracy than the more generalised pre-trained word embeddings. To represent a word embeddings model trained using an outside corpus, we used the Stanford GloVe embeddings, trained on the Wikipedia 2014 and Gigaword 5 corpus [11]. This corpus contained 6 billion words, outputting embeddings with 100 dimensions.

V. Balancing

Another hypothesis was that the accuracy of classification may be improved by training on more balanced data sets. Therefore for both the toxic and movie datasets, we applied differing methods of balancing the dataset for bias correction.

A. Toxic

During the preprocessing stage, toxic was discovered to be highly unbalanced, with 89.4% of comments being non-toxic. To correct the class bias, a oversampling technique called Synthetic Minority Over-sampling Technique (SMOTE) was used. For example, given a dataset of x samples and k features in the feature space. The idea behind the technique involves creating synthetic data points (from a minority class) by considering the k nearest neighbours (in feature space) of a sample from the dataset. A vector is taken between one of the k nearest neighbours, and multiple it by a random number z which lies between 0 and 1 to create the new synthetic point [10].

B. Movie

A simpler method of balancing the dataset was used for movies. Originally this was a 5-class problem, with over 50% of the data classified as neutral, with the other classes containing between 10% and 15% of the data points. For comparison, we therefore combined classes 0 and 1 into a single class indicating negative sentiment, and classes 3 and 4 into a single class indicating positive sentiment, to reduce this to a 3-class classification problem with a distribution of roughly 25-50-25.

VI. Results

A. Spam

The spam dataset was trialled with a wide range of classifiers, support vector machine (SVM), logistic regression (LR), Naive Bayes (NB), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), a CNN-RNN hybrid network, as described in section III-H (CNN-RNN), XGBoost (XGB), and random forest (RF). The accuracy results are tabulated (Table I) for both raw and preprocessed text, with Tf-Idf, word2vec and GloVe encodings.

B. Toxic

The 'toxic' dataset was trialled with a good range of classifiers, SVM, LR, NB, CNN-RNN, XGB and RF, with both preprocessed and raw data. This data is tabulated for balanced data (Table III) and unbalanced data (Table II).

C. Movie

The 'movie' data set was a very hard task for the algorithms to accomplish. They had to select one of 5 classes, based on a ratings system. Additionally, several different labellers were shown different segments of the same review. This has lead to some inconsistency in the labelling, leading to a very noisy dataset. The results are tabulated (Table IV. To simplify the task somewhat, a second trial was run, collapsing classes 0 and 1 into a negative class, and classes 3 and 4 into a positive class. These results are tabulated (Table V).

Due to the multi-class problem of this dataset, the number of output layers increase to 5, complicating the CNN-RNN model and resulting in extended times for the training stage. A test was performed resulting in 42% accuracy, however optimising parameters was not feasible with our limited computational power, so this approach was discontinued without obtaining a full set of test results for populationg either table.

VII. Analysis

A. Preprocessing

Preprocessing led to a minor increase in improvement when Tf-Idf was used as the encoding, on the 'toxic', 'movie' and 'spam' datasets, for most of the classifiers. However, when GloVe encoding was used, there was a significant decrease in performance in the 'spam' dataset for every classifier apart from random forest. This may well be due to the information lost, that in this smaller data set would be necessary for extracting the information needed to classify the mail text correctly. For the 'toxic' dataset, preprocessing resulted in a small (0-2%) increase in improvement for all vectorisers, whether balanced or not balanced.

B. Balancing

It was strongly the opinion of this group that a balanced dataset would give a better classification, an opinion with a good standing in the literature [8], [9]. However, this was not found to be the case. The 'toxic' data set is highly unbalanced, ca. 89.4% not toxic, but when the dataset was balanced, results do not improve for any classifier, in fact across the board, results are either slightly lower (for XGB, CNN-RNN, RF) or considerably lower (for SVM, LR, NB).

C. Datasets

a) Spam: The spam dataset is a clean dataset, most words are correctly spelled and in English. The classifiers performed well on this data set, largely as it is well balanced, 70.1% not spam, which results in better f1 scores, and classifiers finding better rules for discrimination.

b) Toxic: The toxic dataset is a rather 'messy' dataset, much of the text has words that are in no dictionary, contain many spelling mistakes, and also contain numbers, IP addresses, and other information that may make classification difficult. 89.4% of this data is classified as 'not toxic'. This means most results in the table are little better than a dumb classifier that picks the most common. It has already been noted that balancing did not improve accuracy, and more surprisingly, that preprocessing did not improve accuracy. The best performing algorithm here was the CNN-RNN hybrid network (CNN-RNN), using the Tf-Idf vectoriser, on preprocessed data, but there was less than 2ppts difference between the CNN-RNN and SVM, LR, and RF (94.3-95.7). The random forest algorithm was consistently accurate in this dataset.

c) Movie: The movie dataset was confusing for the algorithms, with multiple parts from different comments repeated, but with different sentiment classifications. These classifications did not appear to be particularly consistent, due to the subjective nature of human sentiment. In the competition the best result was only 76.5 % classification accuracy, though it is not reported which approach was used to achieve this. In our work, random forest was consistently the most accurate classifier, achieving the best results when using preprocessed data. There was little difference between using word embeddings or tf-idf for this classifier, and regardless of the pipeline it scored sufficiently higher than the 51.2% mark of a dumb classifier.

D. Classifiers

a) SVM: The support vector machine here performed comparably to other classifiers, however as its hyperparameter space was only briefly explored, it is possible that it could have been improved with some more personal attention.

TABLE I
TABLE SHOWING ACCURACY RESULTS ON THE SPAM DATASET, FOR RAW AND PREPROCESSED DATA, USING THREE DIFFERENT ENCODINGS

Spam	Tf-Idf		word2vec		GloVe	
Classifier	Preprocessed	Raw	Preprocessed	Raw	Preprocessed	Raw
SVM	94.7	94.3	90.5	82.0	89.1	91.9
LR	94.8	94.3	90.7	83.2	89.2	92.0
NB	94.1	93.0	88.8	71.0	78.9	45.2
LDA	94.1	94.0	89.6	93.4	88.1	91.7
QDA	89.0	89.4	88.3	89.3	83.4	82.8
ADA	90.3	87.2	93.4	77.4	73.8	83.6
CNN-RNN	**95.8**	**94.9**	**95.3**	**94.4**	**96.1**	**96.5**
XGB	73.5	74.3	90.6	89.5	77.9	82.1
RF	94.7	94.3	95.2	93.7	85.8	90.7

TABLE II
TABLE SHOWING ACCURACY RESULTS ON THE UNBALANCED TOXIC DATASET, FOR RAW AND PREPROCESSED DATA, USING THREE DIFFERENT ENCODINGS

Toxic - Unbalanced	Tf-Idf		word2vec		GloVe	
Classifier	Preprocessed	Raw	Preprocessed	Raw	Preprocessed	Raw
SVM	94.3	**94.3**	93.6	92.7	93.1	92.7
LR	94.3	**94.3**	93.6	92.8	93.2	92.8
NB	93.8	94.0	90.5	90.5	90.5	90.5
CNN-RNN	**95.7**	93.7	90.5	85.1	90.5	90.7
XGB	92.3	91.0	92.0	92.0	93.0	92.3
RF	94.2	93.5	**94.4**	**93.5**	**93.4**	**92.9**

TABLE III
TABLE SHOWING ACCURACY RESULTS ON THE BALANCED 'TOXIC' DATASET, FOR RAW AND PREPROCESSED DATA, USING THREE DIFFERENT ENCODINGS

Toxic - Balanced	Tf-Idf		word2vec		GloVe	
Classifier	Preprocessed	Raw	Preprocessed	Raw	Preprocessed	Raw
SVM	84.9	84.0	87.6	86.0	86.9	86.6
LR	85.4	84.5	87.4	85.9	86.4	86.2
NB	88.4	88.1	89.3	88.1	84.1	88.1
CNN-RNN	**93.9**	89.4	**92.8**	**92.8**	91.7	91.7
XGB	92.3	**93.3**	90.3	90.0	87.3	90.0
RF	92.3	91.8	92.6	92.0	**92.7**	**92.1**

b) LDA and QDA: Linear discriminant analysis and the related quadratic discriminant analysis were only used with the spam data set, as the smallest and simplest of the datasets. LDA was very similar in accuracy to other algorithms (SVM, XGB,LR, NB), but QDA was moderately weaker.

c) LR: Logistic regression was quite successful on the unbalanced toxic data set and the movie dataset when vectorised by Tf-Idf, with the same accuracy whether preprocessed or not. It was also successful on the spam dataset, with all types of vectoriser.

d) NB: Multinomial naive bayes was notably a poor performer, when GloVe and Word2vec were used, on the spam and movie datasets.

e) XGB: Extreme gradient boost would not have been a good algorithm to bet on in this set of tests, as it had the worst performance in the spam and movie data sets. However, it performed well on the toxic dataset, on both unbalanced and balanced data.

f) RF: A good all-round solid performer, quick and effective. It was the best performer by far in the movie data set.

VIII. CONCLUSION

In our exploration of different methods of vectorisation, balancing and preprocessing the clear outcome has been that simplicity is key. This should not be surprising, but it does seem that the great lengths that were gone to to clean, balance and process the text either had a negligible effect, or was actually detrimental. Tf-Idf, the simplest, was the best overall vectoriser, preprocessing had little effect, and balancing was detrimental. The method that had the highest accuracy across the tasks explored was the CNN-RNN hybrid network,

TABLE IV
TABLE SHOWING ACCURACY RESULTS ON THE 'MOVIE' DATASET, FOR RAW AND PREPROCESSED DATA, USING THREE DIFFERENT ENCODINGS

Movie (5 class)	Tf-Idf		word2vec		GloVe	
Classifier	Preprocessed	Raw	Preprocessed	Raw	Preprocessed	Raw
SVM	58.8	57.4	52.1	52.2	52.2	53.8
LR	58.8	57.6	52.1	52.1	52.2	53.9
NB	55.7	55.6	51.2	51.2	51.2	51.2
XGB	51.3	51.5	51.7	51.4	51.2	52.6
RF	**61.1**	**59.9**	**60.1**	**53.9**	**59.2**	**57.3**

TABLE V
TABLE SHOWING ACCURACY RESULTS ON THE 'MOVIE' DATASET, FOR RAW AND PREPROCESSED DATA, USING THREE DIFFERENT ENCODINGS

Movie (3 class)	Tf-Idf		word2vec		GloVe	
Classifier	Preprocessed	Raw	Preprocessed	Raw	Preprocessed	Raw
SVM	64.8	63.3	54.6	55.1	54.1	57.9
LR	64.9	63.7	54.7	55.2	54.4	58.0
NB	61.0	60.5	51.2	51.2	51.2	51.2
XGB	53.4	53.1	54.1	54.8	54.4	59.4
RF	**68.4**	**66.9**	**68.2**	**59.6**	**66.9**	**65.0**

combining the LSTM approach, known to be successful with text and series based data, with the CNN approach, better known for images. Support vector machines can be an immensely powerful technique, and though they did not perform particularly brilliantly here, with more care and attention to the parameters used, better performances could be elicited. Random forest was a consistent good performer in these tests, it coped with all the tasks reasonably well, required very little tinkering with the parameters, and ran quickly. A combination of random forest and CNN-RNN hybrid network approaches, using an adaptive weighting algorithm such as MWUA [12] may be a fruitful avenue to pursue in getting closer to winning competitions, and with real data. In real world applications, where simplicity and time are important, random forest in combination with other simpler methods would be a good avenue to explore, for example in spam detection and filtering, with the ability to adapt and personalise for each email account being an efficient way to keep performance levels high.

IX. FURTHER WORK

We have tested a wide range of approaches with a good range of datasets. The best performing algorithms combine a range of these methods, depending on each method combined getting different things right in different ways. A good extension to this work would be finding a good way to compare the performance on a selected number of datapoints, to see which methods are more successful on which type, and combining the different methods accordingly, giving a stronger basis for prediction. Exploring different methods of balancing data sets to find one less deleterious to good performance would also be a good extension to this work, testing it on different data sets to see what type of data performs better with this different treatment.

REFERENCES

[1] P. Dangeti, *Statistics for Machine Learning: Techniques for Exploring Supervised, Unsupervised, and Reinforcement Learning Models with Python and R*. Packt Publishing, 2017.

[2] Y. Gal and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," *ArXiv e-prints*, Dec. 2015.

[3] T. Hastie, R. Tibshirani, and J. Friedman, New York, NY, USA.

[4] G. Jigsaw. (2018) Toxic Comment Classification Challenge identify and classify toxic online comments. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[5] Kaggle. (2015) Sentiment Analysis on Movie Reviews classify the sentiment of sentences from the rotten tomatoes dataset. [Online]. Available: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

[6] I. A. V. Metsis and G. Paliouras. Spam filtering. [Online]. Available: http://csmining.org/index.php/enron-spam-datasets.html

[7] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding, method," *CoRR*, vol. abs/1402.3722, 2014.

[8] A. H. S. Solberg and R. Solberg, in *Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International*.

[9] N. Chawla, *Data Mining for Imbalanced Datasets: An Overview*, 01 2010, vol. 5, pp. 875–886.

[10] L. O. H. W. P. K. Nitesh V. Chawla, Kevin W. Bowyer. (2002) SMOTE synthetic minority over-sampling technique.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[12] E. Chastain, A. Livnat, C. Papadimitriou, and U. Vazirani, "Algorithms, games, and evolution," vol. 111, no. 29, pp. 10 620–10 623, 2014.

[13] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053