Alexis Illig

# 29342872

Applied Data Analysis: Assessment 2

I. Pre-Processing

My pre-processing in Python followed most of the common data wrangling and natural language processing steps, as well as tailoring my feature selection to maximise the performance during classification.

Data Cleaning Part 1: Tokenization and Lemmatization

After loading the testing and training documents and training labels into dataframes, the data cleaning of both the training and testing documents proceeded as follows:

1. Sentence tokenization using NLTK's punkt English tokenizer
2. Word tokenization of each sentence using NLTK's word_tokenize.
3. Part of speech tagging using NLTK's wordnet
4. Part of speech tag conversion using treebank so the tags can be recognized by the lemmatizer
5. Lemmatization of the sentence tokens using NLTK's WordNetLemmatizer

This process took ~1.5 hours alone.

Data Cleaning Part 2: Stopword Removal and Stemming

The token removal step involved collecting several sets of characters and words to remove.

1. Removal of punctuation
2. Replacement of contractions with their uncontracted tokens. Word_tokenize does separate some contractions but doesn't recognize all of them. So, I made a full list of English contractions, tokenized that list using NLTK's RegexpTokenizer, and replaced all contractions in the training and test tokens with their full counterparts.
3. Removal of common stopwords. For this, I used the list of common English stopwords we were provided in Data Wrangling since it is about four times longer than nltk's default list of English stopwords and includes numbers and single characters.
4. Removal of 'leftovers'. Lemmatization changed some tokens to words not recognizable as stopwords, such as "was" to "wa" and "has" to "ha". These were removed from the tokens.
5. Stemming using NLTK's PorterStemmer since part of speech tagging and lemmatizing did not target all verb conjugations and possessives. Stemming was performed after stopword removal since the PorterStemmer can be aggressive and changes words such as "party" to "parti". For our purposes, it is only important that the stemming be consistent since it the classification relies on vectorization score coefficients only.
6. Removal of hapaxes. A frequency distribution of the tokens remaining after the stopword removal and stemming was created and hapaxes were identified. The number of hapaxes was 67,639, making up 0.6% of the total tokens. Their presence significantly increased the run time of vectorization, and since they would not be included in the final set of selected features anyway, I removed them.

Vectorization and Exploratory Data Analysis

After cleaning and tokenization, the testing and training sets were vectorized using NLTK's TfidfVectorizer, using unigrams only. The training vectors were then subjected to a correlation analysis in order to identify the most important tokens for each of the 23 classes. Using NLTK's feature_selection.chi2, each class's tokens were ranked using the chi-squared distance between the tfidf score of the token and class.

The correlation results clearly showed that the 23 classes represent topics of news articles (Fig 1).

C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
court	rudd	cents	indigenous	wicket	quake	health	election
murder	minister	market	aboriginal	cricket	earthquake	patient	candidate
police	labor	bank	seeker	inning	tsunami	flu	seat
stab	opposition	index	asylum	bowl	plane	swine	labor
charge	government	rate	church	test	magnitude	cancer	party

C9	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**
climate	film	league	seed	bomb	rail	fire
water	entertainment	goal	hewitt	terror	transport	bushfire
emission	actor	striker	tennis	terrorism	airport	burn
enviroment	art	midfield	wimbledon	qaeda	airline	firefighter
irrigation	music	socceroos	fine	al	freight	blaze

C16	**C17**	**C18**	**C19**	**C20**	**C21**	**C22**	**C23**
abc	flood	animal	car	company	nrl	queen	china
news	floodwater	dog	crash	shareholder	rugby	sail	beijing
assange	rain	whale	accident	profit	bronco	guiness	chinese
broadband	river	rspca	driver	share	rooster	shark	olympics
internet	inundate	zoo	road	creditor	bulldog	prince	medal

Figure 1 Top 5 Most Correlated Tokens by Class

Each class has a clear and unique set of top correlated words. Through this we can see that the classes represent the topics of sports (tennis, soccer, rugby, sailing, beijing olympics), finance (markets, shares), Australia government and elections, immigration, natural disasters (bushfires, earthquakes and tsunamis, flooding), health, crime, climate change, entertainment, terrorism, transportation, broadband, animals, and road accidents.

Classes C23 and C6 also give a clue to the time period from which these news articles were collected. The Beijing Olympics were in 2008, and the last time earthquakes and tsunamis were so *continuously* paired in the news was in 2004.

So, it may be reasonable concluded that the corpus we are analysing is a set of Australian news articles from roughly 2004 to 2008, grouped into 23 article topics.

Re-vectorization, Final Feature Selection, and Dimension Reduction

The final features were chosen by selecting a number of the top correlated features by class, removing the rest, and re-vectorizing. Varying the number of the retained top-correlated features allowed the selection of a set of features that would maximise the chosen metrics of accuracy and macro-F1. This tuning also provided significant dimension reduction. PCA was attempted in addition to correlation, but PCA did not perform nearly as well correlation for feature selection and required too much loss of information in the form of class variance. In the end, the 200 top-correlated words per class were selected, giving a total of 3927 features. This final tuning was then applied to the test set as well.

II. Classifier

Model Selection and Comparison

The cleaned training set was randomly partitioned into “subtrain” and validation sets along an 80/20 split. 5-Fold cross-validation was then used to build each classifier model on the subtrain set, and the fit model was then used predict the validation set. K-nearest neighbour classification was the only non-cross-validated model due to its extreme run time and poor results. From the validation results, the classifier producing the highest macro-F1 score was chosen for presentation.

A brief review of the models used is presented for discussion (Table 1).

Model Function	Model Type	Accuracy	Macro-F1
glmnet()	Lasso Logistic Regression	76.6%	75.9%
MASS::lda()	Linear Discriminant Analysis	75.4%	74.8%
LiblineaR()	L2 Logistic Regression (Cost =1)	77.5%	76.6%
LiblineaR()	L2-L2 Support Vector M. (Cost = 0.23)	77.9%	76.8%
class::knn()	K-Nearest Neaighbor	68.2%	67.1%

Table 1: 5-Fold Cross-Validated for Several Models. The metrics shown are for the validation set.

As can be seen from the table of classifiers, a linear support vector machine approach produces the most faithful classification in both metrics. However, it is worth noting that the range of results, excluding KNN, are very narrow. The exception of KNN lends credence to the unsuitability of unsupervised models. The low variance *between the supervised models,* despite their different approaches, is a sign that the pre-processing successfully identified the most important features and most appropriate dimension reduction. Following on that, the tightly consistent results of the supervised models may indicate that the irreducible error boundary is identifiable.

Among the supervised models, LDA performed least well, which is most likely due to it being un-regularized. The logistic regression and SVM models are both regularized, that is, they apply penalties to increasing complexity in order to combat overfitting. In fact, when looking at the training error (not shown) for all models, LDA performed the best. The fact that LDA then trailed the group for predicting the validation set underscores the need for such regularization.

Linear SVM Tuning and Bias-Variance Tradeoff

All the regularized models were tuned, but I report only the tuning of the selected model, linear SVM.

The type of linear SVM model here uses L2-regularization and an L2-hinge loss[1] function. L2-regularization means the sum of the square of the weights is the penalty used to prevent overfitting (as opposed to L1, just the sum of the weights). L2-hinge loss measures the error between target and predicted value using least square errors (L1 would use the least absolute error).

The tuning parameter, cost, is the penalty parameter tempering the hinge loss. In effect, it sets the trade-off between regularization and classification, so it can be thought of as an overfitting parameter. The cost value sets the weights for the samples that fall inside the margins of each support vector, thereby deciding how hard or soft those margins are. Low cost values result in softer support vector margins, and vice versa.

[1] LiblineaR Documentation (https://www.rdocumentation.org/packages/LiblineaR/versions/2.10-8/topics/LiblineaR)

The effect of softening the margins is to reduce variance at the cost of bias, but that in turn is tempered by cross validation. The effect of a greater number of K-folds is to reduce bias, up to a levelling off.

The tuning of the cost parameter for the subtrain set is shown in Figure 2. The best cost was determined to be 0.225, giving a training accuracy of 77.3% , and a validation accuracy of 77.9% and macro-F1 of 76.8%. The reduction in error in the validation set reflects the desire to balance the bias-variance tradeoff in favour of reducing variance since it is the predictions on new data that is most relevant.

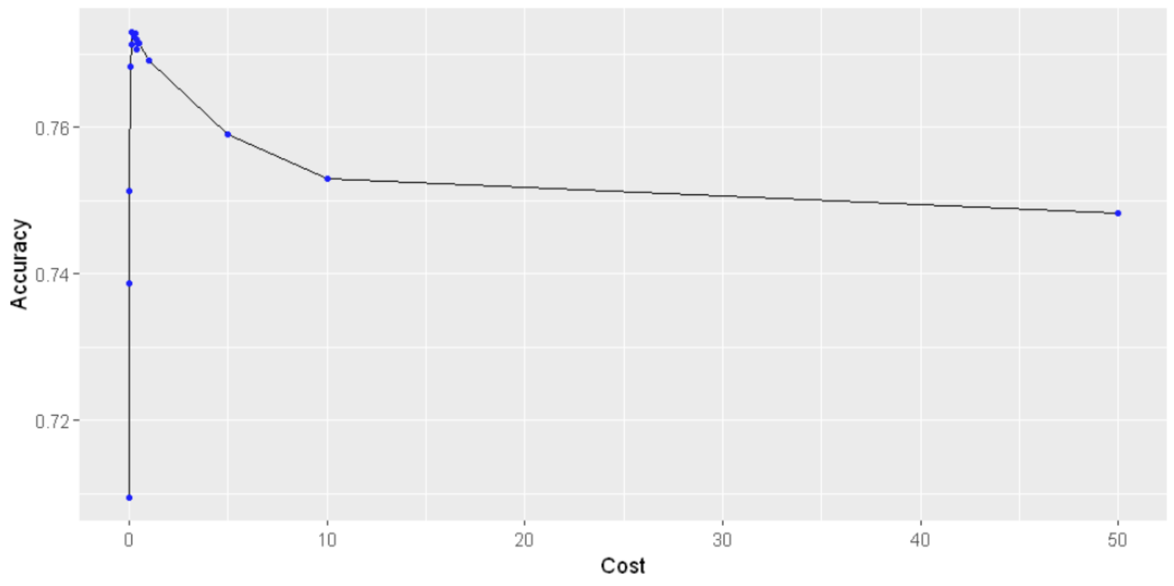

Figure 2 Tuning of the SVM Cost Parameter

A word about using accuracy and not macro-F1 to tune cost; macro-F1 is expected to be the better metric when you have grossly unbalanced classes. This is not the case with our corpus - all classes are roughly the same size. Therefore, we can expect macro-F1 to closely follow accuracy, which it demonstratably does in Table 1.

The other consideration is the kernel to use for SVM. The assumption of the linear kernel is that the classes are linearly dependent on the predictors, which is rarely the case. While a radial kernel can map to higher dimensions to account for non-linear relationships, it is also computationally much more heavy. However for very large datasets with balanced classes, and where the number of samples is much greater than the number of features, the linear kernel can perform nearly on par with the radial kernel[2], as is our corpus.

[2] C. Hsu, C Chang, C Lin, “A Practical Guide to Support Vector Classification”, National Taiwan University, 2016.