Machine Learning Engineer Nanodegree

Capstone Project

Sharon Zollner
September 21, 2017

Project overview

New Zealand is rapidly approaching what has become an unexpectedly close general election. Using machine learning I classified the texts of the press statements from the start of June this year from the six parties most likely to be represented in Parliament following the election based on current polling (National, Labour, NZ First, the Greens, the Maori Party, and ACT).

The aim of this analysis was firstly, to see if the data could be successfully classified by authoring party, and secondly, if so, examination of whether the distinguishing features (words and phrases) were interesting in and of themselves, and could reveal something about the prevailing preoccupations of each party in the run-up to the election.

Problem statement

The problem I am trying to solve with this project is twofold. First, can an algorithm could work out with reasonable accuracy which party had written an unseen document based on its analysis of the training set? Secondly – difficult to quantify but more interesting, in my view, since press statements are always clearly labelled, after all – to identify the key distinguishing features of each party and see if they match up intuitively with my understanding of the different parties' focuses and politics. For example, is talking about the environment a giveaway that the author is the Green party, as one might intuitively expect?

Machine learning for text analysis is a well-established field, and techniques are available in the Python package Scikit-learn. In broad-brush terms, the strategy is:

- Collect the data (scrape the press releases off the websites using BeautifulSoup).
- Analyse the broad structure of the dataset (eg balanced? Of a reasonable size?) Consider what particular challenges this dataset may pose and what techniques may address them.
- Preprocess the data to make it ready for machine learning algorithms (see Preprocessing section below).
- Select an initial algorithm. Multinomial Naïve Bayes is the most common choice for text analysis, and the obvious choice here, but others exist.
- Calculate a performance benchmark. Examine performance using carefully selected metrics appropriate to the problem and dataset.
- Refine and optimise the algorithm using a mix of informal (trial and error) and formal (grid search) techniques. Consider alternative approaches, eg including two- or three-word phrases (n-grams), varying the number of features considered through vectoriser parameters or percentile selection.
- Consider a different algorithm, eg a Support Vector Machine (SVM).

- Finally, the ultimate test – a second test of the optimised algorithm on data that is completely unseen, that has not been used for model selection either: the press statements that have been released since I collected the data for this project.
- Examine the most useful features for the model and consider qualitatively whether they are intuitive, surprising, or simply uninformative from a thematic point of view.

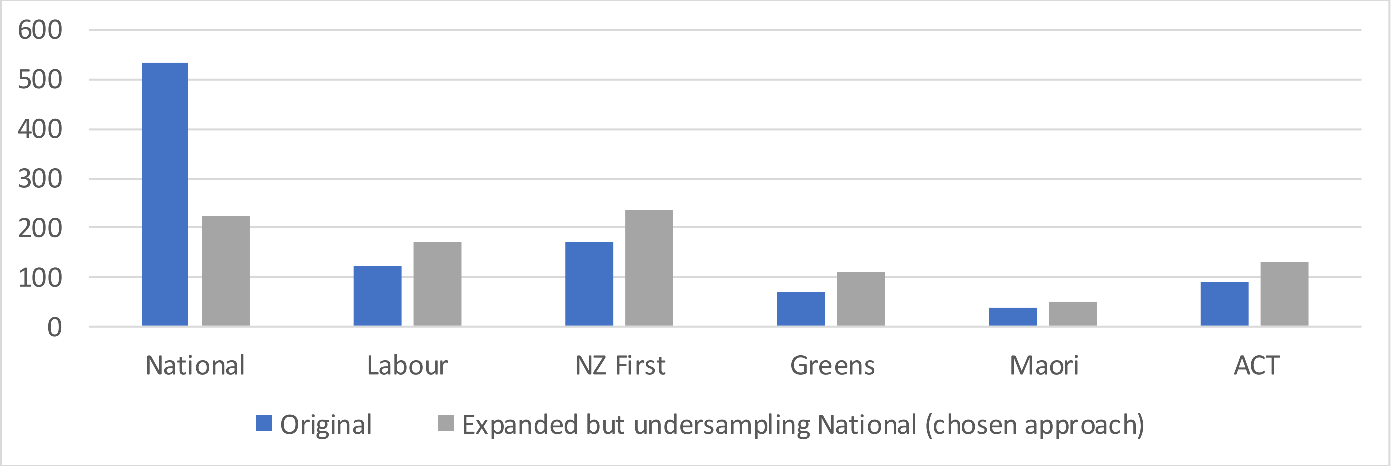
Metrics

The simplest metric for a classification model's success is accuracy: what proportion of documents in the test set were accurately classified? However, this is just a starting point. Particularly in the case of an unbalanced dataset (such as mine proved to be, see below), accuracy may give quite misleading results (eg a credit card fraud model that simply labelled every transaction as valid would have a very high accuracy score but be completely useless).

I therefore used not only accuracy, but also precision[1], recall[2], the F1 statistic (an equally weighted average of precision and recall since there was no reason to weight one kind of error more highly than another), and the confusion matrix (a matrix showing the number of releases of each party that were classified as which party, which the accurate classifications along the downward-sloping diagonal) to see which parties' releases were more difficult to distinguish.

Data exploration

Initially (in early September) I decided to look at two months' worth of press releases, which was just over 1000 documents: National: 535, Labour: 123, NZ First: 169, Greens: 70, Maori: 36, ACT: 91. However, this is a very unbalanced dataset (the National Party, currently in Government, writes far more press releases). I researched techniques for dealing with this problem[i], and concluded that more data was the key. I wanted to keep the datasets for each party covering the same time period, so I took them all back to 1 June, which resulted in 1592 documents. The balance was even slightly worse – National was 900 of these – but it meant I could experiment with undersampling National. After trial and error I decided to remove full ¾ of the National Party documents. The fit of my model improved markedly, and proved robust to different selections (see further discussion below).



[1] Precision: what proportion of the releases predicted to be (party) were in fact authored by that party?
[2] Recall: what proportion of releases authored by (party) were in fact correctly identified as such?

Algorithms and techniques

The text classification problem is commonly addressed with Multinomial Naïve Bayes. For situations with relatively little data, such as mine, this is often the preferred approach, as it has a high bias.[ii] Naïve Bayes is based on a conditional probability model. It assumes the value of a particular feature is independent of the value of any other feature, given the class variable, which seldom holds true in practice, but the algorithm nonetheless often works well regardless. Essentially, the classifier asks, "Given the words in this (test set) ddocument, and how often the different parties used them in the training documents, who is the most likely author of this document?" The Naïve Bayes classifier combines the associated probability model (using Bayes' theorem to simplify the maths) with a decision rule – typically choosing the most likely hypothesis.

The 'features' in Naïve Bayes are the counts of words/phrases in the text (in my case adapted with TF-IDF conversion, as described below) with some degree of Laplace smoothing used to avoid the issue of zero-counts of words (ie unseen words in a test set document) causing multiplied probabilities to go to zero. The Multinomial Naïve Bayes classifier is appropriate for multi-class settings such as this problem, and can be interpreted as a linear classifier when expressed in log-space, which is helpful for interpreting the coefficients as measures of the "importance" of the various identifying features, which is relevant for the second part of my obejctives.

I decided to use TF-IDF conversion[iii], which led to better results in a study[iv] that is fairly analogous to my problem and makes results more robust to different lengths of document. Instead of simple word counts, these counts are up- or down-weighted according to the frequency of the words in the overall corpus (more common words are downweighted). This vectorisation essentially gives the algorithm a head-start by encouraging it to pay less attention to words that are unlikely to prove to be distinguishing features. It is also commonly used for removing words that appear in a large proportion of the documents (max_df, kind of a super-stopwords) but it can also be used to rule out words that are extremely rare in the corpus. Since I was interested in pulling out thematic-style features rather than one-off quirks, I was interested in using this feature (min_df) too.

Not only optimising one algorithm, but also experimenting with alternatives, can provide something of a robustness check on the solvability of the problem at hand. Accordingly, I also modelled a linear Support Vector Machine Classifier, as this can be quickly and easily implemented as it requires the same preprocessing and requires only a few lines of code. A SVM Classifier (SVC) is a non-probabilistic classifier that works by representing the examples as points in space, mapped so that the examples of the separate categories are divided via a hyperplane by a clear gap ('margin') that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall (See Wikipedia: Support Vector Machine).

In the extension to multiple classes, as in this instance, the problem is reduced to a series of binary classification problems: one class versus 'the rest' or else between every pair of classes. Scikit-Learn uses the former as the default – this performed well and was the approach used in this analysis.

Convolutional neural networks have been successfully used for textual analysis, but are considered an unusual approach and are beyond the scope of this project.

Benchmark

The simplest benchmark for the model’s performance is how a random choice model would perform. A very low benchmark is to simply say the random model chooses each class with an equal probability. Then the expected accuracy is simply 1/6 = 0.167. A fairer comparison is to allow the random model to know the proportions of classes in the data (noting that the test data is stratified to match this split). In this case, for each class, the chance that it is drawn out as the actual label and the chance that it is drawn out as the predicted label are both equal to the proportion of the data represented by that class. These are independent probabilities, so the expected accuracy for each class is thus this proportion squared, and the total expected accuracy is the sum of these, as shown below. This lifts the random benchmark to almost 0.21:

National	0.091457
Labour	0.029024
NZ First	0.056119
Greens	0.012296
Maori	0.002341
ACT	0.017174
TOTAL	0.208411

The dataset is new and so there is no clear benchmark for how well a classifier “should” perform. However, this thesis, Word-based classification of political text based on speaker and topic labels, is a fairly similar kind of analysis to what I intend to do, albeit with a more varied and much larger dataset (nearly 5500 documents). Their Naïve Bayes models achieved accuracy of around 65-70% in identifying the speaker, depending on the specification.

I would consider this project as described a success if I were able to achieve accuracy comparable to this on a held-out test set, and more qualitatively, if some of the high-ranking words that characterised each party’s releases had some intuitive meaning that could be interpreted in an interesting way.

Data preprocessing

The following preprocessing steps were undertaken. In practice, it was an iterative process.

- Removing punctuation, case, digits, and common words, or words that intuition suggested were ‘giveaways’ (eg names, party-specific formatting quirks such as “ENDS”, or words such as “spokesperson’ which give away that the party is not currently in Government). These words would assist the model fit but are ‘cheating’, and more practically, would prevent the analysis from capturing the ‘thematic’ information I am actually most interested in;
- Stemming words (eg ‘recognise’ and ‘recognition’ become the same word for the purposes of analysis) – particularly useful when one has a relatively small amount of data;
- Converting the text data to arrays of wordcounts ready for the chosen algorithm, using a TF-IDF vectorizer that allows exploitation of information regarding the relative rarity and commonality of features;
- Applying a SelectPercentile selector to keep only the top x% (60% in the final model) that have the most explanatory power when considered individually (helps to alleviate overfitting, which can easily happen with text analysis, given the high number of features);

- Separating out a test set of data that is not used for estimation, to assess fit (and in particular, check for overfitting).
- Convert to numpy arrays for easy integration with scikit-learn.

Implementation and refinement

The modelling was implemented using the Python library Scikit-learn and its dependencies such as numpy, as well as Snowball Stemmer from the NLTK package.

I started with a simple Naïve Bayes estimation (with TF-IDF vectorisation) on the data just for July and August. This achieved accuracy of around 0.60. Adding the June releases increased accuracy to 0.61. I then (belatedly) stratified the test sample to match the distribution of the data set as a whole; accuracy improved to 0.62.

Looking through the words that were coming up as distinguishing features, I removed those that I considered 'cheating', such as names. This reduced the accuracy to under 0.60. I then experimented with addressing the dataset imbalance by removing some of the National press releases (and from this point on examined more detailed performance metrics including precision and recall for each class). Removing a great proportion of the National releases had a small negative impact on the overall accuracy (as one would expect, given the problem has become harder), but improved both precision and recall for the smaller parties (though not the smallest, particularly). I also explored a few different options for the proportion of 'best' features included via the Selectpercentile option.

The party distinguishing words were a mix of intuitive and a fair degree of uninteresting words at this point (in an iterative process I kept checking the most important features and removing more names and other 'cheats', which made progress something of a "two steps forward, one step back" scenario). At this point precision was excellent for the parties with the most data but very poor for the smaller parties.

At this point I added a very small min_df to the vectorizer to rule out words that were too rare, given my interest in discovering the distinguishing words themselves. I also increased the test size from 0.1 to 0.2 to give a more robust read on the performance for the smaller classes. The combination of these changes increased the model's performance. With min_df of 0.005 (too large a number decimated the number of available features) and a test size of 0.2, I was achieving average precision of around 0.6, and recall of around 0.5, with a slightly better performance on the smaller classes than I had before.

Given the improvement in the smaller classes' fit, I decided to persevere with them (I had been considering dropping them), with the next step being the introduction of n-grams, in my case 2-word and 3-word phrases, as these are much loved by politicians.

This had a marked positive impact on the measured performance of the model, and also on the intuitiveness of the words and phrases that were selected as key features. Accuracy lifted over 0.60, with the following metrics:

```
Score: 0.614130434783
             precision    recall  f1-score   support

        act       0.27      1.00      0.42         7
      green       0.05      1.00      0.09         1
     labour       0.53      0.67      0.59        27
      maori       0.00      0.00      0.00         0
   national       0.96      0.63      0.76        68
    nzfirst       0.94      0.54      0.69        81

avg / total       0.85      0.61      0.69       184
```

Key words and phrases now made more intuitive sense, eg ACT (far right) had “tax” in its top 10, while the Green party had “climate” and “pollution”, the left-of-centre Labour party had “housing” and “health” in their top 10, and NZ First, who campaign against immigration, had this word in their top ten.

The confusion matrix for the above model showed that the model was very biased towards predicting the two parties with the most documents in the corpus: National (even though I had removed ¾ of their releases) and NZ First (the last two columns).

```
Confusion Matrix
[[ 6  0  2  0  3 15]
 [ 0  2  3  0  7 10]
 [ 0  0 13  0 10 11]
 [ 0  0  1  0  6  3]
 [ 0  0  0  0 44  1]
 [ 0  0  0  0  1 46]]
Test set counts: {'act': 26, 'green': 22, 'labour': 34, 'maori': 10, 'national': 60, 'nzfirst': 
47}
```

At this point I decided to move on from exploratory trial and error to a more formal grid-search methodology to choose my final model, using the process outlined in the Scikit-learn documentation.[v] I decided to search over max_df in the Tfidf Vectorizer (0.3, 0.5, 0.7, 0.9), min_df in the Tfidf Vectorizer (0.005, 0.01), percentile in SelectPercentile (40, 60, 80), and alpha (the Laplace smoothing parameter) in the Multinomial Naïve Bayes classifier (initially 0.0001, 0.001, 0.01). I decided to not include a min_df option of zero as I have other objectives that cannot be captured by the Gridsearch scorer, namely pulling out “interesting” words. I therefore only wanted to consider words and phrases that appear in at least ½% of the documents, even at the cost of not maximising the fit.

K was set equal to 6 folds, trading off a relatively small set of data with the need to have all classes represented adequately in the validation set. The scorer was a weighted F1.

I did a broad-brush optimisation and then honed in further on the alpha smoothing parameter, offering it further choices in a range 0.005 to 0.2, but it stuck with its original choice. The optimised parameters were as follows:

max_df = 0.9, min_df = 0.005, percentile = 60, and alpha = 0.01

On the validation fold this model achieved a weighted-average F1 score of 0.802. On the held-out test set (20% of the original data) this model achieved the following performance metrics, a considerable improvement on my trial-and-error interim model:

```
              precision    recall  f1-score   support

        act       0.81      0.78      0.79        27
      green       0.45      0.59      0.51        17
     labour       0.68      0.82      0.74        28
      maori       0.90      0.90      0.90        10
   national       0.91      0.82      0.86        50
    nzfirst       0.87      0.79      0.83        52

avg / total       0.81      0.79      0.79       184
```

It was particularly satisfying to see the improved performance for the parties with less data. The Confusion Matrix confirmed that the model now had much less bias towards predicting NZ First or National (the last two parties).

```
Confusion Matrix
[[21  0  0  0  0  5]
 [ 1 12  3  1  1  4]
 [ 1  0 26  0  3  4]
 [ 1  1  0  8  0  0]
 [ 1  0  0  0 44  0]
 [ 1  1  5  0  5 35]]
```

I chose this as my final Naïve Bayes model, subject to robustness checks (discussed further below).

Model evaluation and validation

A number of robustness checks were performed at this point, above and beyond its performance on the held-out test set detailed above.

a) Are the results robust to choosing a different quarter of the National Party dataset for analysis?

As I had done earlier with my interim model satisfactorily, I tested the optimised parameter model on a slightly different dataset, namely an entirely different set of examples for the largest class, National. With the three alternative datasets the model achieved the following results:

Alternative National A

```
              precision    recall  f1-score   support

        act       0.81      0.81      0.81        26
      green       0.55      0.86      0.67        14
     labour       0.76      0.76      0.76        34
      maori       0.80      0.89      0.84         9
   national       0.98      0.83      0.90        53
    nzfirst       0.74      0.73      0.74        48

avg / total       0.81      0.79      0.80       184
```

Alternative National B

```
             precision    recall  f1-score   support

        act       0.88      0.96      0.92        24
      green       0.55      0.86      0.67        14
     labour       0.62      0.88      0.72        24
      maori       0.80      1.00      0.89         8
   national       0.96      0.85      0.90        52
    nzfirst       1.00      0.75      0.85        63

avg / total       0.88      0.84      0.85       185
```

Alternative National C

```
             precision    recall  f1-score   support

        act       0.81      0.88      0.84        24
      green       0.55      0.75      0.63        16
     labour       0.74      0.78      0.76        32
      maori       0.80      1.00      0.89         8
   national       0.96      0.83      0.89        54
    nzfirst       0.87      0.79      0.83        52

avg / total       0.83      0.82      0.82       186
```

These results are all broadly comparable, confirming the results are not dependent on exact nature of the undersampling.

b) Can another model type achieve a similar performance on the same data-set?

To evaluate whether the Naïve Bayes model is doing a good job, I also implemetned a linear SVC (also recommended for text classification, particularly with relatively small datasets) through a gridsearch process. I took the feature reduction parameters from the optimised Naïve Bayes (Tfidf vectorizer: max_df = 0.9, min_df = 0.005 ; Selectpercentile: percentile = 60) and gridsearched over the LinearSVC hyperparameter C in the range (0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000) (class_weight = 'balanced' to account for the unbalanced dataset). C determines the influence of the misclassification on the objective function.

The model with the best weighted F1 statistic (C=1) achieved the following performance (on the original National-undersampled dataset):

```
Best score
0.865740847982
svc_clf__C: 1

             precision    recall  f1-score   support

        act       0.88      0.96      0.92        24
      green       0.55      0.86      0.67        14
     labour       0.62      0.88      0.72        24
      maori       0.80      1.00      0.89         8
   national       0.96      0.85      0.90        52
    nzfirst       1.00      0.75      0.85        63

avg / total       0.88      0.84      0.85       185
```

The SVC classifier soundly beat the Naïve Bayes for both precision and recall.

c) Do the top features make intuitive sense for each class?

I now had two satisfactory models, and decided to use this fact to achieve a robust analysis of what the key features are distinguishing each class. Analysis of the top 200 features for each party under the two models revealed crossover of between 36% and 59%, which seemed reasonable.

The top few words/phrases for each class that made it into the top 200 of both optimised model sare listed in order of importance below (transformed from the stem back into a representative word/phrase). The parties are listed in the order of the political spectrum from left (redistributive, large government) to right (free-market, pro small government). The ones that make particular sense in my opinion are bolded.

Green	**Labour**	Maori	**NZ First**	**National**	**ACT**
pollution	**fresh approach**	**Māori**	year	provide	**property**
percent	**afford**	**whānau**	prime	**project**	left
low	opposite	lead	**Auckland**	include	cut
review	**fail**	**te**	**immigration**	**improve**	**housing market**
risk	**state housing**	**ora**	**country**	inform	**choose**
agreement	**pressure**	**reo**	want	significant	cause
poverty	allow	**whānau ora**	know	**connect**	**simplify**
doesn't	simply	**iwi**	parliament	available	**save**
foreign trust	**deserve**	seat	**economic**	**contribution**	already
left	**fair**	candid	**economy**	**skill**	far
lead	**teacher**	tai	question	**initiative**	kids
landcorp	**fix**	vote	**job**	pacific	**shortage**
isn't	already	**tamariki**	did	social	share
rivers	**quality**	abuse	office	**deliver**	**red tape**
government needs	**time for a fresh**	party	**export**	**focus**	tape
select committee	**river**	royal commission of enquiry	**industry**	**innovation**	red
cut	**Kiwibuild**	**indigenous**	nothing	**success**	actual
son	**healthy**	general	promise	**trust**	

d) Where the classifier does make mistakes, are they forgivable?

While it is difficult to quantify, a classifier that mixes up say Labour and the Green party is probably a better model than one that mixes up Labour and ACT, who come from opposite sides of the spectrum. For the optimised Naïve Bayes classifier, the following confusion matrix resulted:

```
Confusion Matrix
[[20  1  1  0  1  3]
 [ 0 14  3  0  2  3]
 [ 1  0 29  1  0  3]
 [ 0  1  0  8  1  0]
 [ 0  1  3  0 40  1]
 [ 1  1  4  0  1 40]]
```

The order of parties is ACT, Green, Labour party, Maori party, National, and NZ First. The matrix shows that ACT is most commonly mixed with NZ First, Greens with Labour, Labour with NZ First, Maori Party with Green or National (currently in Government with the latter), National with the Labour Party (but only 3/40), and NZ First with the Labour party. These mistakes are forgivable, in my view.

e) The toughest test – classifying press releases since I downloaded my data

I downloaded and processed 184 new press releases that had been published since I downloaded my original dataset. The Naïve Bayes model achieved the following results:

```
                precision    recall  f1-score   support

        act       0.71      0.75      0.73        16
      green       0.45      0.86      0.59        21
     labour       0.62      0.53      0.57        34
      maori       0.89      0.73      0.80        11
   national       0.71      0.75      0.73        64
    nzfirst       0.90      0.50      0.64        38

avg / total       0.71      0.67      0.67       184

Confusion Matrix
[[12  0  0  0  3  2]
 [ 1 18  7  3  7  4]
 [ 0  2 18  0  5  4]
 [ 0  0  0  8  0  1]
 [ 2  1  9  0 48  8]
 [ 1  0  0  0  1 19]]
```

The optimised SVC model achieved the following results:

```
              precision    recall  f1-score   support

        act       0.71      0.75      0.73        16
      green       0.70      0.85      0.77        33
     labour       0.69      0.74      0.71        27
      maori       0.89      0.73      0.80        11
   national       0.74      0.82      0.78        61
    nzfirst       0.95      0.56      0.70        36

avg / total       0.77      0.75      0.75       184

Confusion Matrix
[[12  0  0  0  4  1]
 [ 0 28  3  2  4  3]
 [ 0  3 20  1  2  3]
 [ 0  0  0  8  0  1]
 [ 4  2  4  0 50  8]
 [ 0  0  0  0  1 20]]
```

These results on completely unseen data are comparable to the testing on the held-out training data, confirming that the model is not overfitted.

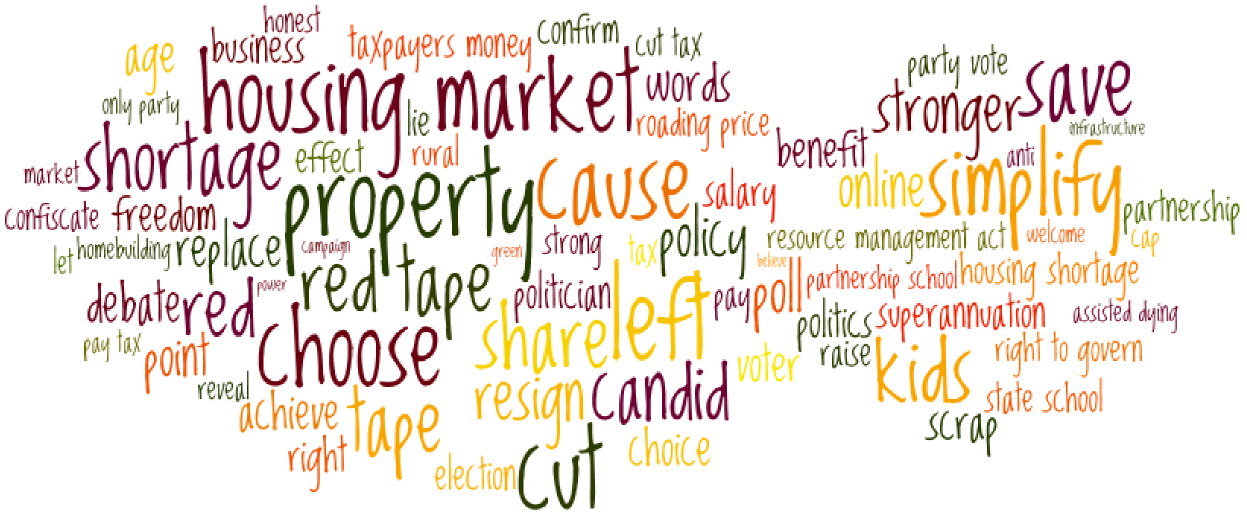
Justification

Despite the removal of helpful but ultimately uninteresting ‘cheat’ features such as names, both the optimised Naïve Bayes and the linear Support Vector classification models soundly beat a random choice (performance at best around 0.20) at identifying the party who authored a given press release, and outperformed the similar political classification study (about 0.70). I am satisfied with the accuracy of my classifier across the various parties, with a high degree of both precision and recall. The results were robust to a range of tests such as using a different subset of National party releases, and testing on newly released completely unseen data. The distinguishing words and phrases make intuitive sense, confirming (along with the test performances) that overfitting is not a problem.

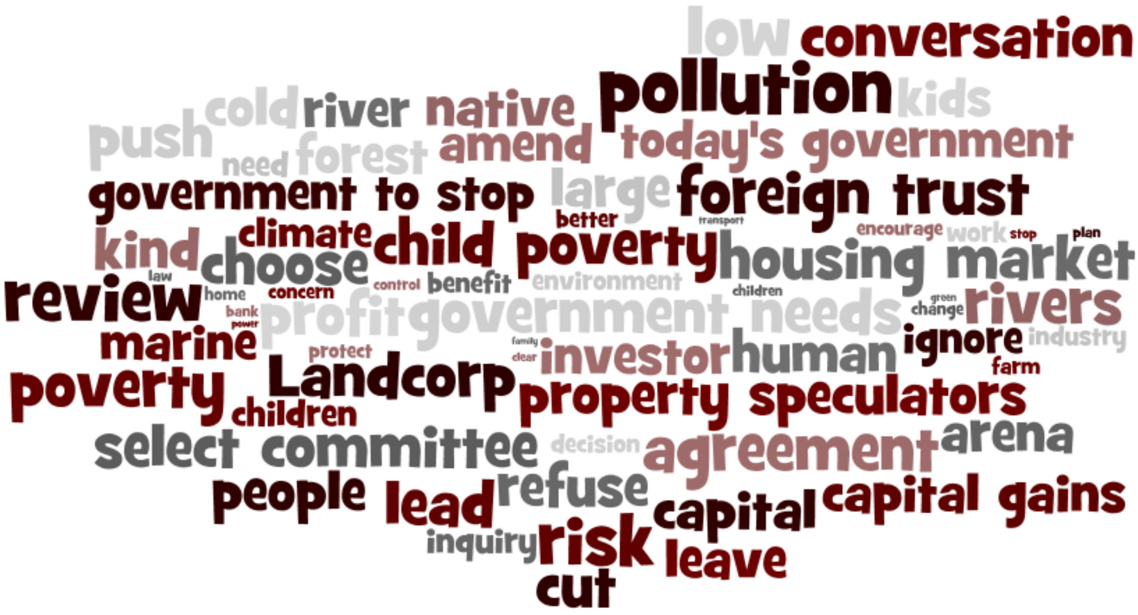
Visualisations

To provide an effective visualisation of the key themes characterising each party the following word clouds were generated on wordle.net using the transformed coefficients from the Naïve Bayes model as weights, removing a few very common words and double-ups. The results are visually appealing and informative.

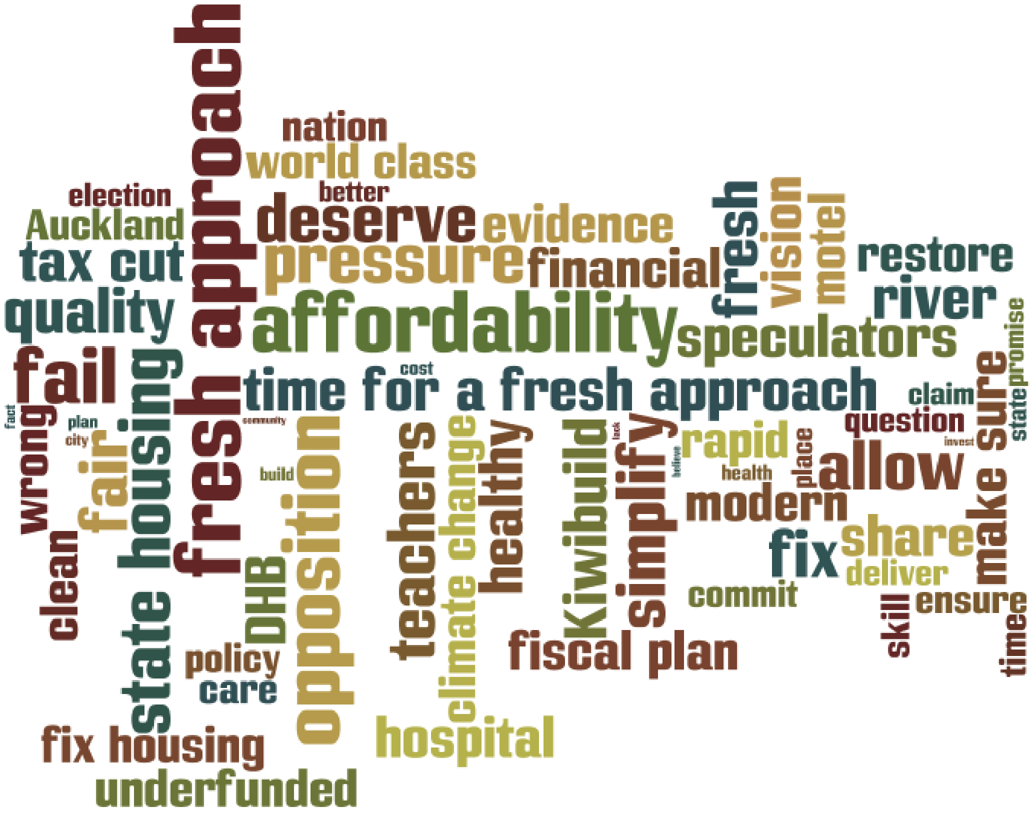
ACT party (pro free-market, small-government, supports current Government):



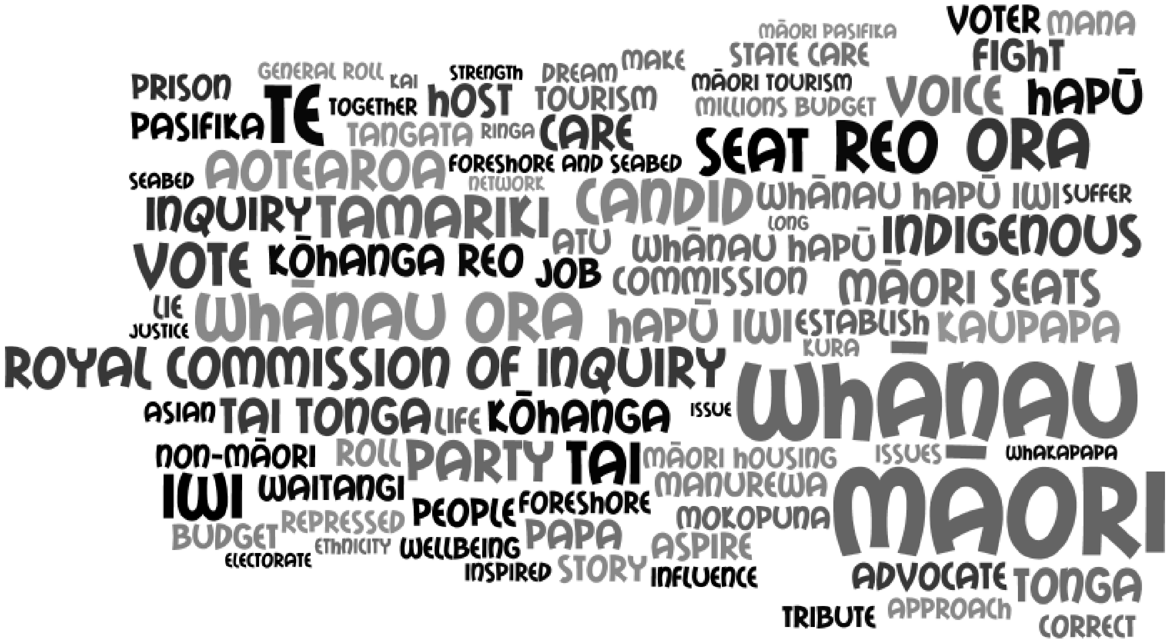
Green party (environment, social justice):



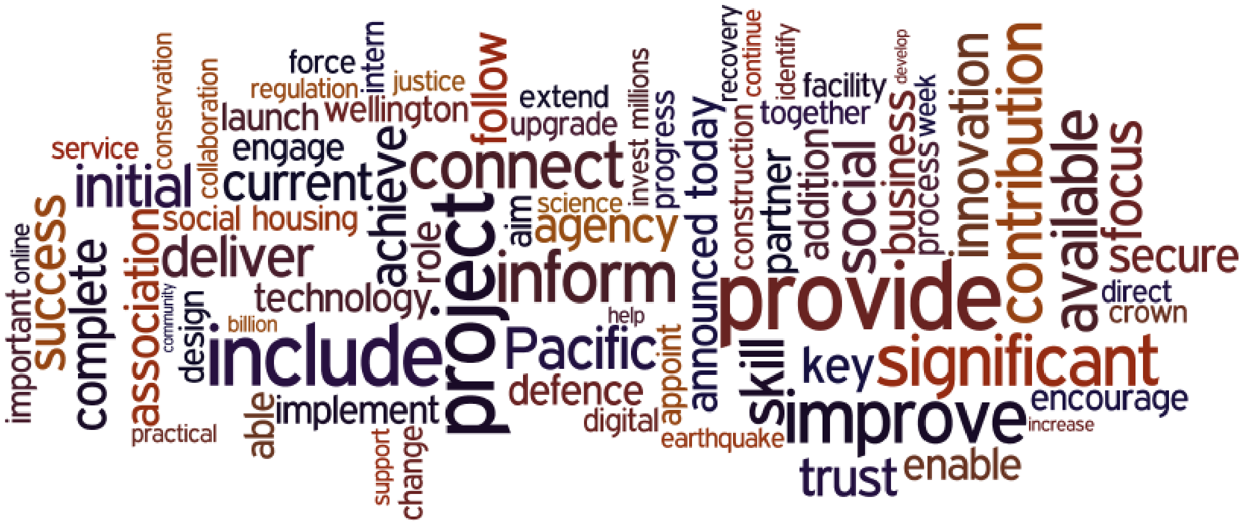
Labour party (left of centre, main opposition):



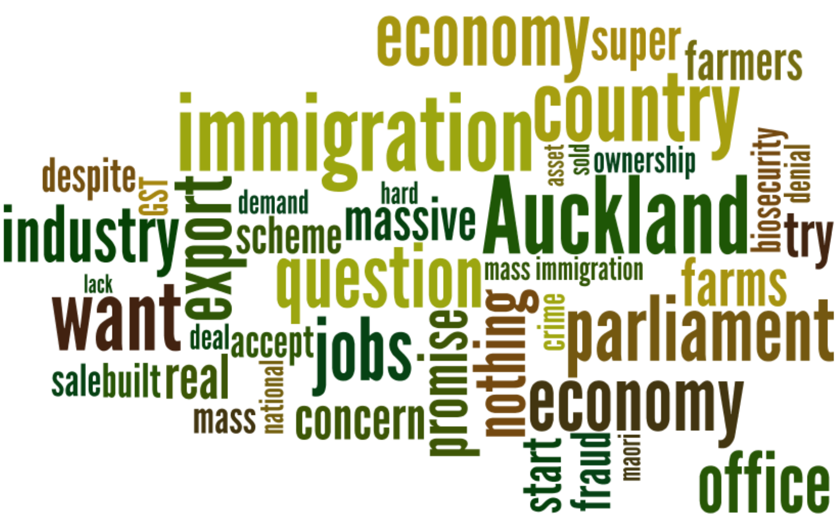
Maori party (Maori rights, currently in Government with National):



National party (right of centre, leading current Government):



NZ First party (anti-immigration, centrist):



Reflection

Recapping the process undertaken in this project, I undertook the following steps:

- Downloading press releases. Vectorize (using TF-IDF), pre-select ingmost useful features.
- Running a Naïve Bayes classifier. Identify key features. Removing any I considered “cheats” in an iterative fashion. Made decisions about the dataset; decided to undersample the National Party as this was heavily over-represented in the dataset.
- Trial and error exploration to understand the implications of different feature selection and model choices such as removing words that appeared either very frequently or very rarely in the corpus, and the hyperparameters of the Naïve Bayes estimator itself.
- Gridsearching an optimised Naïve Bayes model and checking that the features it identified as important made a degree of intuitive sense.
- Subjecting this model to a range of robustness checks beyond its performance on a test set, including checking its robustness to a different subsample of National Party data and also estimating a different estimator – a linear SVC which proved to have an even better performance.
- Examining the performance of both the optimised Naïve Bayes and SVC models to a completely unseen set of data – the press releases that had come out since I downloaded my dataset in early September.

This modelling project was considerably more successful than I expected, given the relatively small dataset, and my dual quantitative/qualitative objectives. There were various surprises along the way, such as the Support Vector Classifier soundly beating the Naïve Bayes model, and various interesting learnings, such as the powerful impact of the min_df parameter in the vectorizer, how to scrape websites, and how to interpret coefficients in a text-classification model. Bug-squashing in my code took quite a lot of time; getting out lists of key features and their coefficients was probably the most significant single problem in terms of time taken to solve.

Improvement

Arbitrary lines had to be drawn in order to keep this project of a manageable size. In particular, there are many specialist techniques available for addressing an unbalanced dataset that are considerably more elegant than my “delete three quarters of one class” approach. While I did implement some techniques such as ensuring I considered more advanced metrics than just accuracy, and weighting my test set appropriately, I could also have used other approaches such as looking at the area under the ROC curve, or kappa. Optimising using these more sophisticated approaches may enable me to improve the classification of the National party class without sacrificing the performance for the smaller classes.

I could have removed the Maori language words in order to make the classification job for the Maori party harder but also more worthwhile in terms of extracting key themes. However, many of the Maori words are in fact interesting themes, eg ‘tamariki’ (children) and whanau (family).

In terms of models, it would have been interesting to consider Bernoulli Naïve Bayes (which takes into account not only the words that are included but also the words that are *excluded*. Or I could have explored combining my two successful models using Adaboost or Community methods to achieve a greater overall performance.

I could also have made more use of functions in my code to reduce repetition.

Resources

The press releases for the parties are available on their websites. See footnotes for the main theoretical references. Coding help was often sourced from Stack Overflow – usually small snippets but more significant excerpts in the case of extracting key features:

https://stackoverflow.com/questions/35353150/sklearn-multinomialnb-how-to-find-most-distinguish-word-in-class

https://stackoverflow.com/questions/11116697/how-to-get-most-informative-features-for-scikit-learn-classifiers

[i] Practical guide to deal with imbalanced classification problems in R by Analytics Vidhya, An overview of classification algorithms for unbalanced datasets by Vaishali Ganganwar and Addressing the Problem of Unbalanced Data Sets in Sentiment Analysis by Asmaa Mountassir, Houda Benbrahim and Ilham Berrada.

[ii] https://nlp.stanford.edu/IR-book/html/htmledition/choosing-what-kind-of-classifier-to-use-1.html

[iii] For further detail see Multinomial Naïve Bayes for text categorisation revisited.

[iv] Word-based classification of political texts based on speaker and topic labels

[v] Working with text data – a Scikit-learn tutorial