Classifying Climate Change Attitudes with Twitter

Felicie Bizeul, Mathieu Grasland, Iason Tsardanidis, Eleftheria Tetoula Tsonga and Anil Keshwani (Group 7)

Abstract

We sought to build a classifier to distinguish tweets in a binary manner according to their underlying attitudinal belief regarding climate change: climate change denier vs. believer. To do this, we collect a relatively balanced dataset of c. 18,000 tweets from people (accounts) with a known attitude and label the tweets accordingly. We perform standard NLP cleaning to transform our tweets into token lists and embed these token lists in vector spaces with a number of approaches. We implement penalised logistic regression, support vector machine classifiers, random forests, a variant of boosting and naïve Bayes, and finally aggregate these classifiers into an ensemble classifier via soft-voting. We achieve classification accuracy of 0.92 and identify salient features used by our trained classifiers.

Introduction

We undertook a natural language processing task wherein we sought to classify tweets on the basis of the belief the account holder had with respect to whether climate change is genuinely occurring.

Broadly, this report follows our project's progression which consisted sequentially of

1. **Data collection** via web scraping to circumvent Twitter's API limitation
2. **Data cleaning** to normalise the linguistic data
3. **Embedding tweets** in a vector space trialling multiple methods
4. **Training of statistical classifiers** on an 80% subsample test dataset including hyperparameter optimisation via cross-validation in a grid or random search
5. **Validation of Predictions** on a test dataset (the remaining 20% of our collected tweets)
6. **Assembly of an ensemble classifier** to attempt to capitalise on any variation in the probabilistic predictions output by individual classifiers to return a better overall classifier

Data Collection

In order to collect a large sample of tweets to train statistical classification machinery, we identified a set of accounts known to belong to individuals or organisations with a clear, public position on climate change from which to pull tweets; the specific accounts (in the form of Twitter handles) are detailed in Appendix A. We pulled tweets containing the string "climate" published between 1st January 2012 and 1st January 2020 from these accounts and labelled tweets according to the position held by the account holder.

To pull tweets, we initially wanted to use Twitter's API (e.g. via the Python interface Tweepy) but quickly discovered its prohibitive limitations; for example request rate limits and search returning only tweets from a sample published in the past 7 days. We instead scraped tweets directly using Twitter-Get-Old-Tweets-Scraper run in the shell. An example command is shown.

```
python main.py --username "GretaThunberg" --query "climate change" --since 2018-12-01
--until 2020-07-28 --max-tweets 1000
```

We retrieved a dataset consisting of 18,009 tweets with 7,438 tweets from climate change sceptics or deniers and 10,571 from putative believers.

Our choice of collection and labelling method was expedient given that no perfect labeler was available and manual labelling of individual tweets was infeasible for the size of dataset we collected but the logic behind it sets up some important assumptions which we comment on in the discussion.

Data Cleaning

We performed standard data cleaning of our tweets as is required for NLP tasks.

Specifically, we implemented the following as methods within a scikit-learn estimator class:

- Removal of URLs, extraneous whitespace, punctuation and English possessives: "[noun]'s"
- Removal of stopwords - Common, 'contentless' function words e.g. has, was, a, or, into
- Cleaning of Twitter mentions to usernames
- Stemming (via Porter's Algorithm) - Consolidation of inflected word forms into stems
- Expansion of Twitter abbreviations and slang into regular speech, e.g. "AFAIK" is transformed into "As Far As I Know" via a lookup
- Expansion of emoji into textual emoji shortcodes, i.e. single word representations for individual emoji

As can be seen in the `CleanText` class, we chained and applied all of these to give a linguistically normalized set of tokens for each tweet.

Embedding Tweets

Having acquired a set of normalized tokens for each tweet in our dataset, we represented the tweets in vector space via four embedding methods.

1. Bag of Words

All unique tokens across the whole *corpus* (collection) of tweets are put in a set and a vector of length equal to the cardinality of this unique token set is used to encode each tweet. Each element in this vector corresponds to a specific token in the corpus. An individual tweet is represented with a binary encoding of elements in the vector such that the element has a 1 in the token's position if it appears in the tweet and 0 otherwise, i.e. a tweet, $t_i \in R^V$, has elements $t_{ij} \in \{0, 1\}$.

This encoding was produced both using unigrams (described above) and bigrams, the same as the above but where vector elements correspond to length-2 sequences of tokens instead of individual tokens.

2. Frequency Bag of Words

An additional encoding identical to the above with the elaboration that the specific number of times a token occurs in a tweet is recorded in the vector[1], i.e. $t_{ij} \in \mathbb{N}^+$, instead of simply its presence of absence being flagged with (effectively) an indicator variable.

3. Term Frequency-Inverse Document Frequency (TF-IDF)

An elaboration of the previous vector representation of tweets wherein the token *frequencies* across tweets are additionally weighted according to the log of the tokens' inverse frequencies across tweets:

$$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \cdot \log \frac{N}{n_t}$$

where the left term denotes a term's frequency across the corpus and the right term is the logarithm of the number of tweets expressed over the number of tweets in which a specific token occurs.

In our context, for *term*, read *token* and for *document*, read *tweet*. This tweet representation implies implies $t_{ij} \in \mathbb{R}^+$.

[1]This is not a true frequency; it is a count. We retained the wording for consistency originally with external sources.

4. Pretrained Word2Vec Vectors

We used pretrained embeddings from Word2Vec.

Summarily, Word2Vec word embeddings themselves are derived from training of a shallow neural network on the dummy task of predicting the colocation of other words in a *window* of words using a large corpus of news article data. For example, given a window of length four (in bold) in the phrase "**the quick brown fox** jumped over the lazy dog", the neural network was trained to predict which words occur nearby, if given the input "quick". This dummy task was not of inherent interest, but rather the weights of a hidden layer estimated via training were - to use as embeddings for individual words.

The number of neurons in the hidden layer in the architecture described above can be arbitrarily set, and in the case of the publicly published embeddings, was set to 300. The approach brings the significant advantage that whilst bag of words embeddings produce vector representations of length 2207 in the case of unigrams and 3201 in the case of bigrams, the vector embeddings we retrieve via Word2Vec are constant at 300.

In order to produce representations of tweets, we straightforwardly average the available individual word vectors of the words constituting a given tweet. Words for which no embedding was published were removed before (or during) this process.

Notably, this final embedding method does not require words to be stemmed or otherwise preprocessed to yield a vector. As such, we pass the raw tweets through a function in this latter case.

Training Classifiers and Prediction Validation

The central component of our analysis is the training of individual classification estimators. We provide a short summary of each method. For each classifier, we tune hyperparameters via either a grid or random search (the latter for classifiers which were highly expensive to fit) using cross-validation within our training dataset. We comment briefly on this as well.

A table of accuracies is provided in the results section following.

Logistic Regression

An archetypal GLM approach modelling the logit (log of the odds) our binary outcome: being a "denier". This highly efficient method but can overfit high dimensional datasets. We prevent overfitting by using regularization. It seeks to reduce coefficient sizes using L1 or L2 penalties; ElasticNet uses a combination of both. If the regression coefficients are to high, they are removed (L1; LASSO) or shrunk (L2; Ridge).

For the hyper-parameters, we tweak:

- the penalty which is the norm used for the regularization
- the solver used for optimizing the problem (mostly for computational efficiency given the penalization)
- the L1-L2 penalty ratio when using ElasticNet
- The maximum number of iteration for the algorithm to converge, which is very useful to know whether the optimization was solved
- C a constant, the smaller it is, the stronger is the regularization

Support Vector Machine

Support Vector Machine classifier (SVC) is a method that seek to separate group of individuals using the variables by a maximal margin. The algorithm searches for the hyperplane that separates observations, according to the given features. It doesn't require large dataset to provide accurate results. It can solve linear and non-linear classification problems. In high dimensional space, it is very efficient when the number of variables is higher than the number of samples.

For the hyper-parameters, we tweak:

- C a constant, the smaller it is, the stronger is the regularization.
- the kernel used to transform the data, in order to find non-linear separation.
- gamma value, when Radial Basis Function (RBF) kernel is used

Random Forest

Random Forest classifier is based on the bagging algorithm and decision trees. It creates many trees on bootstrap samples of the data. Each (likely overfit; high variance) tree is fit on a random sample of the features; therefore trees are likely to be different. The algorithm averages across these high variance trees; bootstrap aggregation or *bagging*. The bagging step reduces the variance of the method. Whilst a decision tree tends to overfit data, this methods avoids this. The Random forest method is efficient because it can automatically handle outliers and missing values. The predictive performance can compete with the best supervised learning algorithm.

For the hyper-parameters, we tweak:

- The number of trees in the forest
- The maximum depth of the trees
- The minimal number of observation required to split a internal node
- The minimal number of observation required to be leaf node
- The number of features to search when looking for the best split

Cat Boost

This recently published method creates decision trees using gradient boosting, and is purposed specifically for categorical features which adapts it well to our setting. This algorithm is an implementation of ordered boosting. It uses gradient boosting by using permutation of subset of dataset. On small dataset, gradient boosting methods usally overfit small dataset, but using permutations corrects this problem. It can handle automatically missing values.

For the hyper-parameters, we tweak:

- The number of splits for numerical features
- Depth of the tree
- The coefficient at the L2 regularization term of the cost function
- The learning rate
- The maximum number of trees that can be built

Naïve Bayes

The methods is based on the Bayes rules assuming all features are independent conditional on the label (denier vs believer). It combines decision rules without being sensitive to variation in the attitude label explained by *covariance* of predictors (due to its conditional independence assumption). In practice, each feature will add a contribution to the probability of an event. It is combined with kernel density estimation when using Gaussian Naive Bayes. It computes the mean and variance of each feature for each classes with normal kernel, to generatively model probabilities.

For the hyper-parameters, we tweak:

- The variance smoothing, that is a value added to variance calculation

Results

The classification accuracies are reported (along with additional figures from alternative boosting models we ran; we opted for Cat Boosting as a single boosting model to include in our ensemble classifier).

	Cat Boosting (CB)	Gradient Boosting (GB)	Light Gradient Boosting (LGB)	Logistic Regression	Gaussian Naive Bayes (NB)	Random Forests (RF)	Support Vector Machines (SVM)
Bag-of-Words	0.913	0.900	0.905	0.910	0.873	0.903	0.904
Frequency Bag-of-Words	0.913	0.901	0.904	0.912	0.850	0.902	0.903
Bag-of-Words (Bigrams)	0.913	0.906	0.905	0.920	0.898	0.902	0.911
TF-IDF	0.906	0.891	0.900	0.915	0.849	0.899	0.909

Ensemble Classifier

After tuning the best hyperparameters of the classifiers, we use a soft-voting method in order to compile them into an ensemble classifier. We excluded the Naive Bayes classifier because of its relatively poor performance. The soft-voting method simply takes the average probabilities of the 4 other algorithms. After finding the best cut-off threshold - 0.55 - with this method we achieve an accuracy over the test set of 0.922, which is a modest improvement over any of our individual classifiers.

Discussion

Having shown that it is possible to predict attitudes with respect to climate change with over 90% accuracy without explicit language (e.g. grammar) modelling, we can make a few summative remarks and highlight some caveats in our analysis.

Our classification methods largely used similar features for separation of attitude classes, with frequently used word or hashtag features including (the cleaned and tokenised forms of) "#sharethis"; "climate crisis"; "sun"; "global warming"; and "alarmist", to name a notable few. The robustness of these features' importance across methods identifies them as distinguishing shibboleths[2] for climate change deniers and believers

It is unclear *why* bigrams provided the best classification accuracy. One explanation is that bigrams provide vector representations with the greatest number of features. Whilst we can confidently assert that this greater number of features did not simply provide better classification accuracy due to overfitting - we as performed cross-validation and, in any case, report test set errors in the results section - the greater number of features may well be capturing additional variation for a small number of the ~3,600 test set observations (tweets). Whilst increasing n to yield longer n-grams may yield better results still, this would be more computationally intensive as vector representations would obviously grow.

Whilst Naive Bayes and support vector machine classifiers should, according to our reading and research, perform best, we found this was not the case, but the accuracies between classifiers were of course very similar so this finding may not be very robust.

Whilst Word2Vec performs well on benchmarking tasks related to semantic modelling, it performed poorly when used with logistic regression and random forests in our trials, yielding typical accuracies around 85%. Furthermore, when populating length-300 vectors with random Gaussian noise, we were also able to achieve 75% classification accuracy on the test set; i.e. this 85% should not be compared to a 50% baseline, but rather a 75% accuracy one. It may be unsurprising that this was the case as our task was not one of language or semantic modelling. The fact that Word2Vec outputs vectors embeddings of length 300 and that we summarized tweets simply by summing these word vectors may have compounded any noise found per word vector. More simply still, a length 300 vector is likely to contain less information than a length 3201 one.

[2]A shibboleth is any custom or tradition, usually a choice of phrasing or even a single word, that distinguishes one group of people from another.

Caveats and Possible Improvements

We conclude by mentioning a couple of caveats, with suggestions for how one could improve on our end classifier.

Foremost, as mentioned initially, our method of acquiring labeled data limited us to collection and labeling of tweets account-wise for practical motives. This has two significant downsides.

First, our classifier may perform poorly on accounts outside the dataset used for this project; even our test data came from accounts listed in Appendix A that made up the initial data pull. Systematic bias may exist in our classification with respect to features, for example, the specific lexicon used by the accounts from which we gathered tweets may be well classified by our model, but other individuals' tweets may be less so.

Secondly, we fail to take account of the multi-level structure of our data, consisting of tweets nested within individuals, and tweets from a particular individual cannot truly be said to be independent samples. Explicitly modelling this multilevel structure may yield generalizability of our model and better estimation of parameters. In the simplest case, one could implement a scheme in the context of regression via random and fixed effects.

Additionally, we of course do not implement complex machinery in the form of transformers or other modern neural network architecture (such as Bert) which was outside the scope of this project. The lack of explicit language models certainly mean that recoverable information is lost, but conversely the over-90% accuracy in the absence of such complexity may equally be a surprising and positive result.

Core References

Mikolov et al. (2013) Distributed Representations of Words and Phrases and their Compositionality *NeurIPS*
Prokhorenkova et al. (2018) CatBoost: unbiased boosting with categorical features *NeurIPS* Manning, Raghavan and Schütze (2008) *Introduction to Information Retrieval*
Dan Jurafsky and James H. Martin (2019) *Speech and Language Processing*

Appendix A: Accounts by Attitudinal Label

Deniers	Believers
BjornLomborg	jrockstrom
CatoInstitute	elonmusk
DineshDSouza	MichaelEMann
realDonaldTrump	bruneski
GrrrGraphics	xiyebastida
RepJoeBarton	vict_barrett
KurtSchlichter	johnpauljos
LtGovTimGriffin	Luisamneubauer
mattwridley	israhirsi
MaximeBernier	UNFCCC
MicheleBachmann	WWF
SeibtNaomi	climateWWF
PrisonPlanet	ayanaeliza
JimInhofe	EricPooley
JunkScience	frannyarmstrong
HeartlandInst	GreenpeaceUK
ToddRokitaIN	severnsuzuki
TomFitton	iraghislain
RonPaul	alexandriav2005
	kehkashanbasu

Deniers	Believers
	felixfinkbeiner
	jeromefosterii
	laurajoparker
	jamie_margolin
	fisher_danar
	xiuhtezcatl
	twcrowther