Oracle DeepTweets

Mirto Randellini [PGX- DS-T2468]

January 2023

1 Introduction

This is a report on Oracle's DeepTweet task for the Data Science internship selection process. The task is a binary classification for tweets into the two classes Politics and Sports using the tweet text as input. The training dataset contains 6525 labelled tweets, while the test dataset contains 2610 unlabelled tweets. Visualizing the label distribution in the training set shows that it is balanced. Inspecting the vocables in the training and test datasets shows that many words in the test set are not in common with the training set. This means that only using word frequencies as features will produce predictions with limited accuracy.

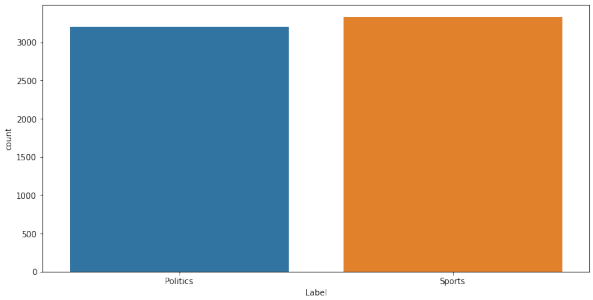


Figure 1: Label distribution in the tweets of the training set.

2 Preprocessing

The text in the dataset contains punctuation marks and several non-alphanumeric character that can interfere with proper text tokenization. The first step of preprocessing is removing these special characters. After cleanup, the second step is to split the text into individual words or tokens. This can be done using a variety of techniques, such as splitting the text on whitespace or using regular expressions to identify word boundaries.

	TweetId	Label	TweetText
0	304271250237304833	Politics	'#SecKerry: The value of the @StateDept and @U...
1	304834304222064640	Politics	'@rraina1481 I fear so'
2	303568995880144898	Sports	'Watch video highlights of the #wwc13 final be...
3	304366580664528896	Sports	'RT @chelscanlan: At Nitro Circus at #AlbertPa...
4	296770931098009601	Sports	'@cricketfox Always a good thing. Thanks for t...

Figure 2: The tweet texts before being cleaned.

After tokenization, it is often necessary to normalize the words in order to keep a meaningful feature space. This can involve lowercasing all the words, stemming the words (reducing them to their base form), or removing stop words (common words that do not convey much meaning, such as "the" or "a"). By calculating the frequency of appearance of words per message it is evident that the overall most common words are stop-words, so they are filtered out.

	Common_words	count
0	the	4324
1	to	2527
2	in	1929
3	of	1921
4	a	1515
5	for	1362
6	and	1259
7	on	1196
8	rt	891
9	is	858
10	at	783
11	with	711
12	amp	599
13	from	512
14	by	495
15	you	491
16	we	472
17	will	467
18	this	407
19	has	404

Figure 3: The most common words are filler words that are not useful for classification.

After these operations, the most common words in the two topics are different, this will be helpful for the downstream classification.

	Common_words	count
1	indvaus	347
2	test	314
3	1st	220
4	bcci	201
5	runs	196
6	bbl02	194
7	atp	181
8	amp	177
9	bigfinals	165
10	cfc	162
11	tennis	146
12	win	142
13	final	137
14	first	136
15	aus	133
16	india	123
17	1	118
18	team	114
19	game	114

	Common_words	count
1	rt	368
2	president	329
3	obama	236
4	pm	188
5	conference	159
6	video	155
7	new	152
8	press	140
9	medvedev	138
10	nelsonmandela	136
11	live	134
12	cabinet	130
13	chief	129
14	us	129
15	secretary	128
16	meeting	121
17	prime	115
18	minister	114
19	people	108

Figure 4: After removing stop-words the difference in lexicon is apparent between categories.

3 Feature extraction

Different feature extraction methods and models were evaluated for the task. Feature extraction is the process of extracting useful features from raw data. These features can be used to train machine learning models, or to understand the underlying structure of the data.

Feature extraction is an important step in the process of preparing data for machine learning, as it helps improve the performance of the model by reducing the number of dimensions in the data, and by focusing on the most relevant features.

3.1 Word Frequency

The first feature extraction method used is word frequency. Word frequency is a feature extraction method used in natural language processing (NLP) and information retrieval (IR). It involves counting the frequency of each word in a document or a corpus of documents, and using the resulting counts as features for machine learning algorithms or for information retrieval tasks such as document search. In the context of NLP, word frequency is often used to build features for text classification tasks, such as spam detection or sentiment

analysis.

For this project, the word frequencies are computed on the whole training set after preprocessing. As shown in figure 3 and 4, the preprocessing of the text is a necessary step. After the word frequencies have been computed, the next step is to convert the word counts into a numerical feature vector that can be used as input to a machine learning algorithm. This can be done using a variety of techniques, such as one-hot encoding or term frequency-inverse document frequency (TF-IDF) encoding. In our case, we first implement a one-hot encoding of each tweet, which produces a matrix containing the frequency of each word for each tweet. Then, we experiment with a TF-IDF encoding.

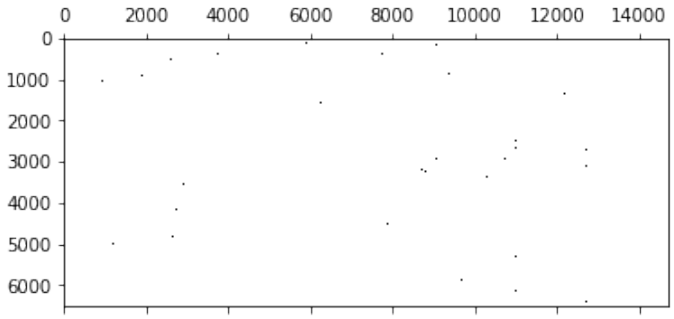


Figure 5: The table resulting from one-hot encoding is extremely sparse (black dots represent non-zero values), so dimensionality reduction is necessary.

3.1.1 Frequency Threshold

The first approach for dimensionality reduction is applying a frequency threshold. The words are sorted by the overall frequency in the training set. Then, the least frequent words are discarded.

Hyper-parameter Tuning The word list is cut off at different frequency thresholds to have multiple feature selection strategies. The optimal threshold is selected by using the cross-validation score of a model trained with the filtered word-frequency features. Figure 6 shows the distribution of the word frequencies in the training dataset.

3.1.2 Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that is used to identify patterns in data and to reduce the dimensionality of feature spaces. In the context of feature extraction on word frequencies, PCA can be used to reduce the dimensionality of a document-term matrix, which represents the frequency of different words in a collection of documents.

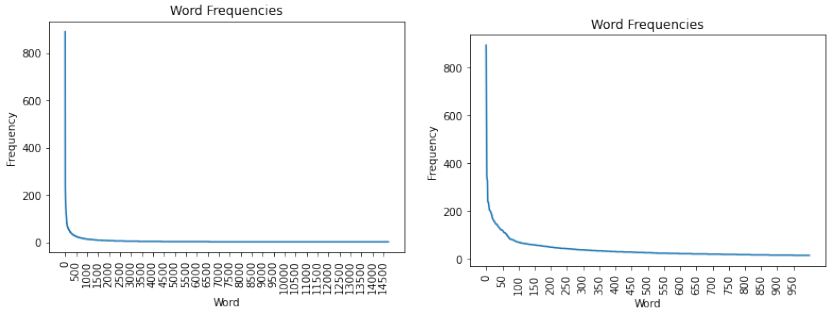


Figure 6: The dataset is cut according to the optimal frequency threshold.

PCA works by finding the directions (called "principal components") in the data that capture the most variance, and then projecting the data onto these directions. This can be useful for identifying the most important words in a set of documents, or for grouping similar documents together based on the words they contain.

In our case, we have a document-term matrix that represents the collection of the tweets, where the columns represent all the possible words that can appear in a tweet. Using PCA, the dimensionality of this matrix is reduced from 14727 to a smaller number, such as 1000 or even 100, while still retaining much of the information about the frequency of different words in the documents. This can make it easier to analyze and visualize the relationships between the documents and the words they contain. In figure 7, it can be seen that the majority of the *sports* tweets and the *politics* tweets can be separated with just 2 principal components.

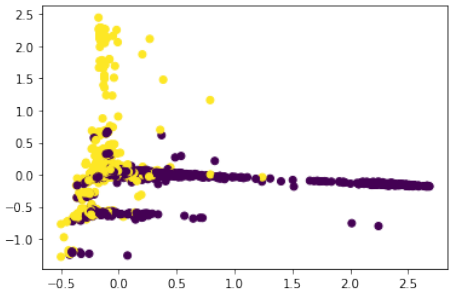


Figure 7: Scatter plot of the tweets using the first 2 principal components. Yellow dots represent *sports* tweets and purple dots represent *politics* tweets.

Hyper-parameter Tuning The optimal number of principal components is selected by using the cross-validation score of a model trained with the low-dimensional document-term matrix.

3.1.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) is a statistical measure that is used to evaluate the importance of a word in a document or a collection of documents. It is often used in information retrieval and text mining to measure the relevance of a document to a particular query or to a set of documents.

TF-IDF is based on two factors: term frequency (TF) and inverse document frequency (IDF). Term frequency is the number of times a word appears in a document, normalized by the total number of words in the document. Inverse document frequency is the logarithm of the total number of documents in a collection, divided by the number of documents that contain the word.

TF-IDF is calculated by multiplying the term frequency by the inverse document frequency for each word in a document. The resulting value is higher for words that are more important to the meaning of the document, and lower for words that are less important or more common.

To extract more informative features from the text, in addition to the frequency of the single words we also extract the frequencies of groups of words, also known as n-grams. In particular, we count the frequencies of single words, 2-grams, 3-grams and 4-grams. With this technique, the amount of features is large (more than 200000). Therefore, we apply a truncated singular value decomposition (SVD) for dimensionality reduction, which extracts 500 features. The combination of TF-IDF and truncated SVD is commonly known as latent semantic analysis (LSA).

3.2 Full text

3.2.1 E5 Text Embedding

Text embedding is a technique used to map words or phrases in a text to a numerical vector space. The goal of text embedding is to capture the meaning of the words and the relationships between them in a way that can be used by machine learning models.

There are several different methods for creating text embeddings, including word2vec, GloVe, and fastText. These methods all involve training a model on a large dataset of text, such as a corpus of news articles or a collection of Wikipedia pages. The model learns to predict the likelihood of a word occurring given the context of the surrounding words, and uses this information to create a high-dimensional vector representation of each word.

Text embeddings are useful because they allow machine learning models to take into account the meaning of words in a text, rather than just treating them as individual tokens. This makes it possible to use the embeddings to perform tasks such as text classification, language translation, and text generation.

E5 [Wang et al., 2022] is a family of state-of-the-art text embeddings that transfer well to a wide range of tasks. The model extracts from each tweet an embedding vector with 384 dimensions which is representative of its overall meaning.

4 Model training

Logistic regression is the statistical model used to predict the binary outcomes. It is called "logistic" because it uses a logistic function to model the probability of a particular event occurring.

In logistic regression, the dependent variable is binary, and the independent variables can be continuous or categorical. The goal is to find the best coefficients for the independent variables that will allow the model to accurately predict the binary outcome.

In our case, the dependent variable is the binary variable indicating if a tweet is about *Sports* or *Politics*, while the independent variables are the features extracted from the tweets.

Besides logistic regression, we also experimented with a multi-layer perceptron classifier, which did not perform better than the simpler logistic regression.

5 Evaluation

Having extracted the features using various methods, we now proceed to train our models and evaluate their performance using 5-fold cross-validation. The model used on top of the various feature extraction method is a logistic regression. As mentioned previously, the hyper-parameters for frequency threshold and PCA feature selection are optimized with a grid search on the results of 5-fold cross-validation.

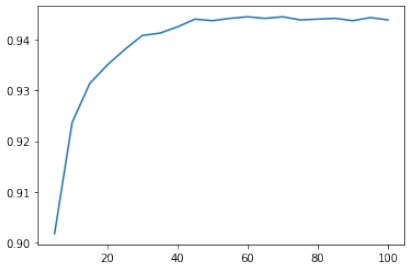


Figure 8: Grid search on frequency thresholds.

The following table contains the results for 5-fold cross-validation for the above specified models and features:

Model	CV score	Public score
Word frequency without feature selection	0.9439	0.8659
Word frequency with frequency threshold	0.9440	0.8672
Word frequency with PCA	0.9464	0.8621
N-gram features + LSA	0.9162	0.8544
E5 embeddings	**0.9505**	**0.8723**

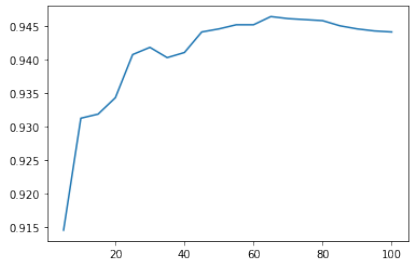
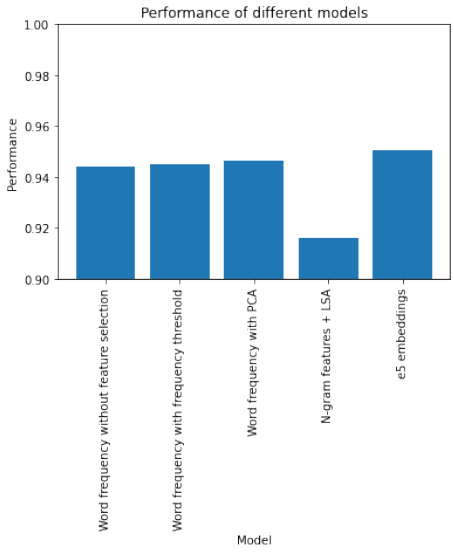


Figure 9: Grid search on number of principal components for PCA.



From the results, it can be seen that the CV scores of every method is consistently higher than its public leaderboard score. This is not surprising since the test set vocabulary is quite different from the training set vocabulary. Nonetheless, the relative rankings of the model performances according to the CV scores and to the leaderboard scores are very similar.

6 Other approaches

To further improve the prediction accuracy of our model, we could try several approaches. One option is to use more advanced classification algorithms such as Support Vector Machines or Random Forests, which have been shown to be more powerful than logistic regression in some cases. Another option is to use domain-specific models for feature extraction, for example embedding models trained specifically on tweets, which can be more effective than general-purpose ones like E5. Additionally, we could try using an ensemble of models,

which can be more accurate than any individual model. We could also consider using additional features such as the length of the tweet, the presence of certain hashtags or keywords, and the number of retweets or likes. These features may provide additional context and signal that can help improve the accuracy of the model. By carefully tuning the hyper-parameters of our models via a more extensive search and using cross-validation to evaluate their performance, we could increase the accuracy of our predictions.

References

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., & Wei, F. (2022). Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.