News Stance Detection

Information Retrieval and Data Mining

Weisi Han - 7 April 2018

Introduction

With the development of internet and media platforms, information can be shared in many convenient and efficient ways. However, this also accelerates the spread of fake news. This coursework aims to detect the fake news by comparing the stance of the news headline and the news body. The whole process can be divided into the following 10 subtasks.

Task 1. Train - Test Split

Based on the data exploration, we found that the number of the four training set classes is highly imbalanced. Overall, the combination of news headline and body pairs are more biased towards unrelated group, which accounts for 73% in the whole training set. Following is ***discuss*** class (18%) and agree class (7%), whereas class ***disagree*** is underrepresented, taking up only 2%. We then split the whole training set into a training subset and validation subset with ratio 9:1. In each subset, the four-class has been allocated according to a similar ratio of the original training set. The detailed split result is shown in Figure 1 and Table 1. (Related code can be found in Task1.ipynb)

	Agree	Disagree	Discuss	Unrelated
Training Set	0.0736	0.0168	0.178	0.731
Validation Set	0.0736	0.0168	0.178	0.731

Table 1. Ratio of Four Classes in Training subset and Validation Subset

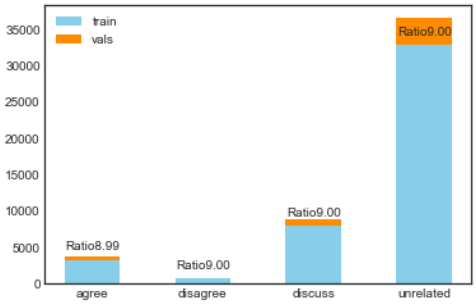


Figure1. Ratio of data numbers in Training Subset and Validation Subset

Task 2. Vector Representation of Headline and Body

With an aim of categorising each pair of headline and article body into four classes, we need to transform this raw text information with variable length into fix-sized numerical feature vectors so that they could be applied into different classifiers. The first step is to do data processing, which involves removing punctuations, numbers, stop words, and leaving only the root words in lowercase format.

During the further text vectorisation process, I adopt two common approaches: Word2Vec and TF-IDF, to extract features and conduct conversion. The first approach requires us to create our own Word2Vec word embeddings. I apply the *word2vec* method in ***gensim*** packet directly. As it expects a sequence of sentences as input, each headline and body paragraph has been tokenised and made as a sequence of sentences to train a Word2Vec model. After this step, it will return an 100-dimension vector representation of each single word within the training test. However, what we need to figure out is the vector representation of the whole headline and main body and thus, we average the word embeddings for word in the headline sentence or body paragraph. For example, in the following headline, "police find mass graf least body near mexico", $v_{hdl} = v('police') + v('find') + \ldots + v('mexico')$, and the final vector representation is the normalised format: $v_{hdl} = \frac{v_{hdl}}{|v_{hdl}|}$.

Another method to obtain the vector representation is through TF-IDF, which indicates Term Frequency – Inverse Document Frequency. It is computed by multiplying the TF and IDF. TF measures how frequently a term occurs in each headline or body, whereas IDF can be calculated as taking the log of the documents (headlines and bodies) numbers divided by the the number of documents which contains this specific word. It measures how important a term is. We first compute the TF-IDF value for each word in every headline or news body. Then, initiate a vector dictionary with length 19984 (number of all unique word in the dataset) so that each of the headlines or news bodies will be represented by vectors with equal length.

In summary, each headline and main body in our training and testing dataset will be finally represented by using two vectors, one is Word2Vec (with length 100) and another is TF-IDF (with length, 19984) . The cosine similarity will be calculated for every headline and main body pair by using the following equation The final result of each Head Body pair is within range (-1, 1).

$$Cos\_Sim(Head, Body) = \frac{\sum_{k=1}^{n} Head_k Body_k}{\sqrt{\sum_{k=1}^{n} Head_k^2}\sqrt{\sum_{k=1}^{n} Body_k^2}}$$

Task 3. Language Model Establishment and of KL-Divergence Calculation

In this part, I would like to build a simple Unigram Query-Likelihood Model to present a probability distribution over words in all datasets. Here we assume each word in the headline or body is associated with a probability and the previous word has no impact on the next word. The summed probability of every word in one document is 1 and the document is a pair of news headline and body. As each term in the document is sampled independently from each other, the probability of the query (one headline and one article body) can be calculated as the production of the probability of each term.

However, as some words appear in the the headline may not exist in the article body as well, our model will assign zero probability to that event. To avoid such problem, I tried two types of smoothing techniques, Laplace and Lindstone, and finally adopt the latter one to smooth the word probability since Laplace smoothing gives too much weight to the unseen terms. The Lindstone smoothing formula is shown below:

$$P(w|D) = \frac{tf_{w,D} + \epsilon}{|D| + \epsilon|V|}$$

where **ϵ** represents a small number adding to all count, aiming to renormalise the whole probability distribution. In fact, I tried different values of **ϵ**, ranging from 0.0001 to 1, and finalised value as 0.1, since it provides the best probability distribution.

The next step is to calculate the KL divergence based on the term probability distributions of the headline and body pairs. KL divergence is a measure of how the the term probability distribution in the headline diverges from the bodies. As zero-probability does not exist after we conduct smoothing, we can directly apply the following formula to calculate the KL divergence.

$$D_{KL}(P_{Head}||P_{Body}) = \sum_i P_{Head}(i) log \frac{P_{Head}(i)}{P_{Body}(i)}$$

where P is the probability distributions for headline and article body.

Task 4. Implement Alternative Features and Distances.

In order to enhance the overall performance of our classifier, I created 4 more features apart from the above cosine similarity and KL divergence.

1. Pearson Correlation Coefficient

The Pearson Correlation Coefficient investigates the strength of the association between pairs of quantitative variables. In our stance detection case, it describes the relationship between the headline - body pair. An coefficient of 1 indicates a strong positive relationship between variables, whereas 0 indicates no linear relationship. This feature can be calculated by the following left equation.

$$Pearson_r = \frac{n(\sum xy) - (\sum x(\sum y))}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \qquad Euc\_dist = \sqrt{\sum_{k=1}^{n}(x_{1k} - x_{2k})^2}$$

2. Euclidean distance

Euclidean distance can be seen as the distance comparison feature. The Euclidean distance between two points in n-dimensional space measure the length of a segment connecting the two points. As the headline and body have been represented by vectors, the distance between these documents can be represented. A large distance means a lower similarity between two pairs.

3. Fraction of Word Overlaps between Headline and Body

Inspired by the baseline model, I also created a word overlap feature, which is calculated by using the common words appeared in both headline and body divided by the the union word set of each headline-body pair. Intuitively, the higher number of overlapped words, the higher possibility the headline and body are related.

4. Word Polarity

The polarity of the headline and body is determined by counting the number words with negative sentiments. These words include: fake, fraud, not, etc. If the headline and body have the same odd or even number of negative words, they tend to be associated with the same sentiment.

All code related to Task 2-4 can be found in Task2-4.ipynb

Task 5. Features and Distances Plot

In this part, I choose to analyse the distribution of ***word overlap*** and the ***Pearson correlation coefficient***. These values are computed by using the TF-IDF vectorisation representation. The bigger picture plots the distributions of four classes all together, where each colour stands for a separate stance. As the overall distributions for the four features tend to have a heavy-tailed characteristic, we will control the value range of the x-axis to focus on its main body. The four small subplots are the feature distribution for each stance. It is obvious that for both of the two features, the unrelated class distribution is noticeable different from other classes. Focusing on feature ***word overlap***, more than 20000 unrelated data have a value less than 0.01 and even for the rests, they are still less than 0.05. However, for related classes: agree, disagree, and discuss, their word overlap distributes across x-axis from 0 to 0.3. The situation for ***Pearson correlation coefficient*** is similar. All of the values for unrelated class is concentrated around 0 to 0.1 whereas the values for related classes spread across the x-axis (from 0 to 1). By exploring the feature distance distribution, we can find there is a significant difference between the related and unrelated stances, therefore, these two features might be helpful when applied into the models.

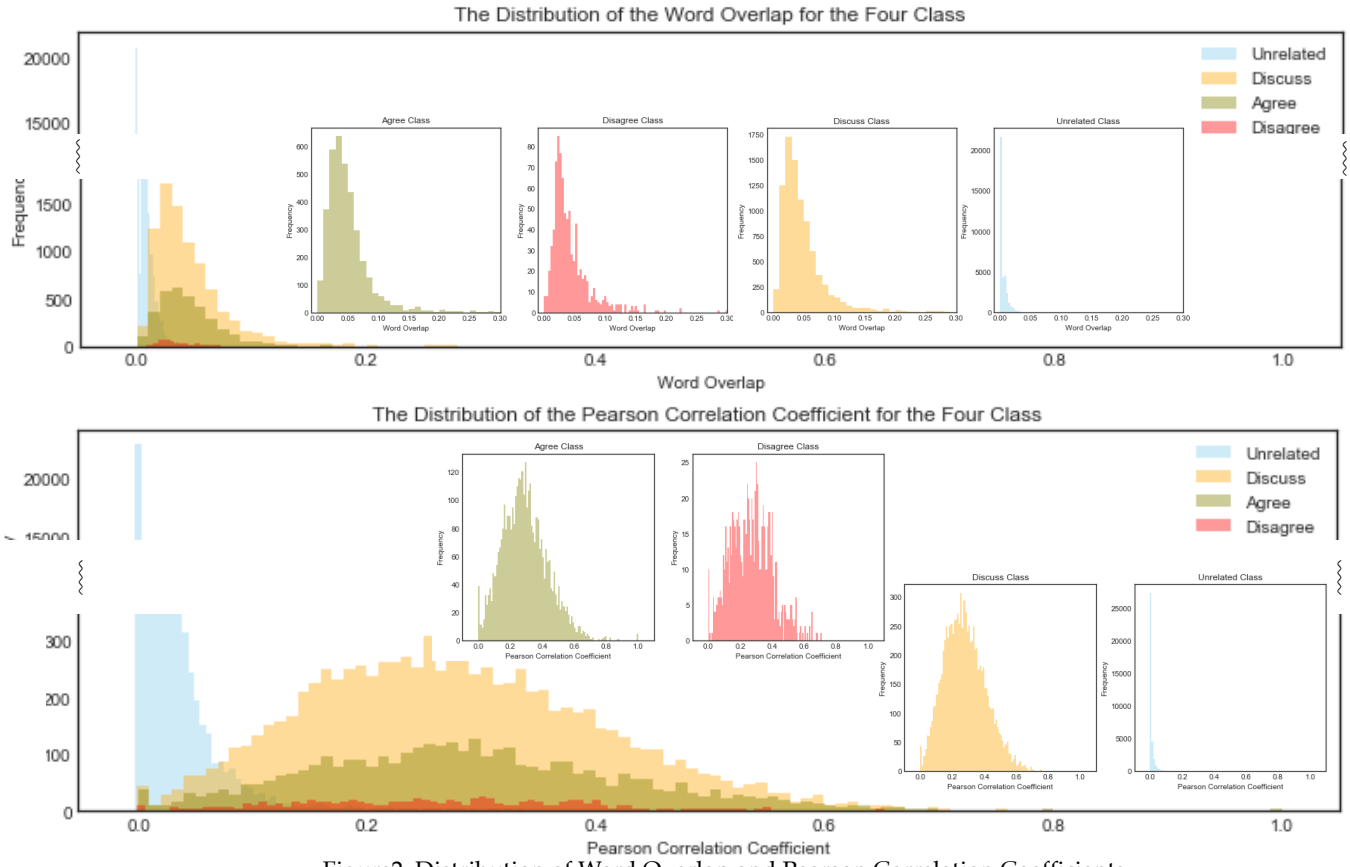


Figure2. Distribution of Word Overlap and Pearson Correlation Coefficients

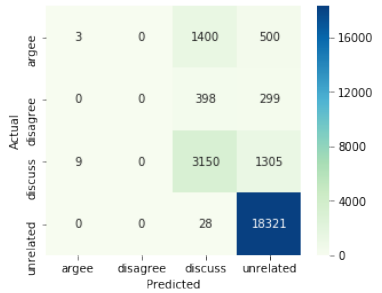
Task 6. Implementation of Linear Regression and a Logistic Regression

Because of the page limit, The detailed model construction process, feature description and selection have been put in the Appendix. Our stance detection problem can be treated as a multi-class classification problem with four stances. However, rather than constructing a complicated multi-class classifier, the code I implemented (both linear regression and logistic regression) only works for simple binary classification. Therefore, I decide to use the one-vs-all method. The idea is to split this problem into four separate binary classification problems, and each of these binary models for the individual classes is assessed against its complement classes. (For each stance class, eg. Agree, I would apply my binary classifier to predict whether this headline-body pair falls into this category (the result is the probability)) Prediction is then performed by

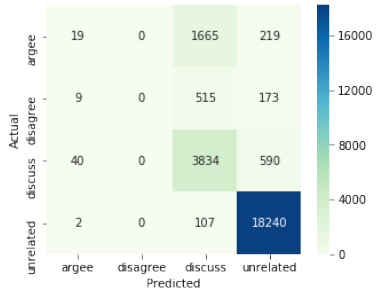
running all these binary classifiers and selecting the predictions with the highest prediction score. In our case, we first apply our binary classifier to predict each individual stance and the combined result would be our final output.

Task 7. Analyse the Performance of the models

There are many ways to assess a predictive model. Although the percentage error (accuracy) is an important indicator to measure most classification models, it does not hold well against our highly imbalanced dataset. Therefore, I compute the confusion matrix for the true positive, true negative, false positive and false negative class for four types of predictions, which is listed as heat map bellow. Furthermore, the precision, recall, F1 score will be calculated based on that.



Class	Precision	Recall	F1	Support
Agree	0.25	0.00	0.00	1903
Disagree	0.00	0.00	0.00	697
Discuss	0.63	0.71	0.67	4464
Unrelated	0.90	1.00	0.95	18349
Average	0.78	0.80	0.80	25413



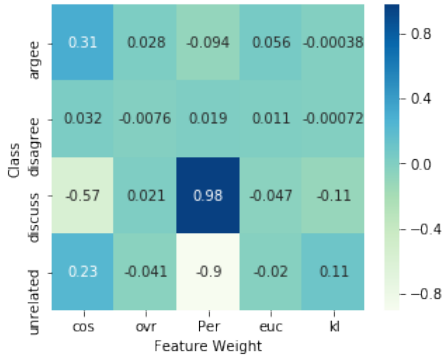
Class	Precision	Recall	F1	Support
Agree	0.27	0.01	0.02	1903
Disagree	0.00	0.00	0.00	697
Discuss	0.63	0.86	0.72	4464
Unrelated	0.95	0.99	0.97	18349
Average	0.82	0.87	0.83	25413

Figure3, 4. Table 2,3. Performance of the Linear Regression Model and Logistic Regression Model

Overall, the Logistic regression model outperforms Linear Regression Model. The accuracy score for the multiple linear regression model is 0.845. Logistic regression is higher than that (0.869). Both models have a high precision, Recall and F1 score for identifying discuss and unrelated class, and the logistic model is better than Linear when correctly distinguish agree class. Unfortunately, both models fail to identify the disagree class (will be improved in task 10). Looking at the confusion heat map, the vast majority of the predictions end up on the diagonal (predicted label = actual label). However, there are a number of misclassifications. Both models tend to miscategorise label agree to discuss. This can be explained as these two stances are both in related group, and they indeed discuss the same claim. The only position take difference is hard to be detected by the model.

Task 8. Feature Importance Exploration

Recall that we have adjusted all five predictors to a common scale before feeding them into the models. We can directly figure out the feature importance by observing the weight for each term in the linear or logistic function. The value of coefficients have been plotted as heat map in Figure 5 and 6, where either the darker blue (large positive) or the lighter yellow (large negative) can represent a higher weight. For agree and



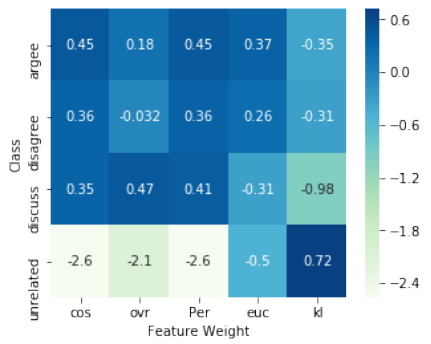


Figure 5, 6. Heat map of feature weight in Linear Regression (left) and Logistic Regression (right)

disagree classes, cosine similarity and Pearson Correlation Coefficients are both important in Logistic regression, where as in Linear Regression, only Cosine similarity is of high weight. For class discuss, Pearson in very significant in Linear regression whereas in Logistic, the most promising feature is KL divergence. As for unrelated class, cosine, overlap and Pearson are of extremely high weight in Logistic regression, while Pearson is the only one important feature in Linear Regression. In a nutshell, the cosine similarity and Pearson Correlation Coefficient can be seen as important features for our stance detect project.

Task 9. Literature Review

"Fake news detection is defined as a task of categorising news along a continuum of veracity, with an associated measure of certainty." (Conroy, 2016). Stance section is is the initial and crucial step of identifying fake news, which involves estimating the relative stance of two pieces of text (headline and article body) related to a particular topic. Specifically, whether the news article body agrees with, disagrees with, discusses the headline topic or these two parts are completely unrelated. To achieve this goal, two main process have to be solved in this challenge.

One is identifying features that might be helpful for the stance detection task, and another is to build several different machine learning models to conduct stance classification process. Based on my paper research, the popular features for stance may include : 1) Headline and body length. 2) Refuting words and Polarity words from FNC1, which intends to detect the sentiment information 3) Word overlap and Word N-grams features: (n = 2, 3, 4) which are the overlapping grams extracted from the headline and body pair. 4) Vector features: these include the Word2Vec, TF-IDF or Bag-of-Words representation of the headline and body pair. 5) Distance Features: like Euclidean distance, Word Mover's distance and Cosine Distance. Furthermore, by combining some of the features mentioned above, the relationship between the body and headline pair can be better represented.

These features will be later applied to some ML models for classification. Popular models used in semantic analysis and stance detection include Support Vector Machines (SVM) (Zhang et al., 2012), Naive Bayesian models (Oraby et al., 2015) and Random Forest (Papanastasiou, 2017). In terms of the Neural Network approaches, Convolutional Neural Network (CNN) has been widely used to learning from the processed headline and main body vectors. Additionally, other models are employed in some ensemble methods, like Gradient-Boosted Decision Tree (GBDT) because of the robustness to deal with different scales of feature vectors (Largent, 2018).

Task 10. Ways for Improvements

One of the biggest shortcoming of my models is the weakness to predict disagree class. Therefore, I would like to use two approaches to deal with this problem.

1. Downsampling

As the conventional models are often biased towards the majority class, under-sampling the majority class with unrelated class label or oversampling the minority class would be a better approach to balance the stance proportion. Exemplified by oversampling, we could duplicate the minority disagree entries by two or three times so that the distribution of the four class in the training set can be changed from 7:2:18:73 to 7:4:18:71 or even 7:6:17:70. By doing so, the learning performance of the model would be improved.

2. Train new models.

I would like to apply the following five models: XGBoost, Gaussian Naive Bayes (GNB), K-Nearest Neighbours (KNN), Linear Discriminant Analysis (LDA) and Decision Tree. In sum, XGB and GNB perform better than the other three classifiers, with XGB having a slight advantage with an accuracy of around 87%. However, for Decision Tree and KNN, even though their accuracy is 3 percent lower than XGB, they successfully predicts 52 and 18 disagree class separately.

All the code related to the new created models and their performance evaluations can be found in Task 10.ipynb

Reference

Largent, W. (2018). Talos Targets Disinformation with Fake News Challenge Victory. [online] Blog.talosintelligence.com. Available at: https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html [Accessed 9 Apr. 2018].

Oraby, S., Reed, L., Compton, R., Riloff, E., Walker, M. & Whittaker, S. (2015). And That's A Fact: Distinguishing Factual and Emotional Argumentation in Online Dialogue

Papanastasiou, Y. (2017). Fake News Propagation and Detection: A Sequential Model. SSRN Electronic Journal.

Zhang, H., Fan, Z., Zeng, J. & Liu, Q. (2012). An Improving Deception Detection Method in Computer−Mediated Communication. Journal of Networks, 7 (11).

Appendix

In this part, I would like to illustrate how to build my Multiple Linear Regression Model step by step.
We have variable X and output Y. The input variable has n features. (Later decided… five selected features (Cosine Similarity, Pearson Correlation Coefficient, Overlap, Euclidean distance, and KL distance))
Therefore, our model can be represented as:
$Y = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_n x_n$, where $x_0 = 1$

By converting this equation to the matrix form, we can get
$Y = \beta^T X$, where $\beta = [\beta_0, \beta_1, \beta_2, \ldots, \beta_n]^T$ and $X = [x_0, x_1, x_2, \ldots, x_n]^T$

Our hypothesis function can be defined as $h_\beta(x) = \beta^T x$

And the Cost function can be represented as $J(\beta) = \frac{1}{2m} \sum_{i=1}^{m} (h_b(x^{(i)}) - y^i)^2$

In order to find the optimal weight β, we would like to use Gradient Descent to minimise the cost function. The gradient descent updating function can be expressed as:
$\beta_j := \beta_j - \alpha(h_\beta(x) - y)x_j$, where α is the learning rate.

After iteration several times, it will converges and we can obtain the most appropriateβ to construct the linear regression model.

The implementation of logistic regression is similar as the linear one. Both of the detailed code can be find in Task6xxxx.ipynb