Task1: Splitting

For the sake of similar class ratio in both the train set and hold-out set, I shuffle the stances in the training dataset and randomly select 90% of the stances to the training set. The remaining stances are left for the hold-out set. In the end, the train set and the hold-out set has a similar class distribution as below:

	Train	Hold-out
Agree	0.072	0.079
Disagree	0.017	0.017
Discuss	0.176	0.187
Unrelate	0.735	0.717

Task2: Vector representation

In my final model, I chose to represent each word in the headline and body using its TF-IDF value.

TF-IDF is often used in computing text similarity. It is the multiplication of word term frequency(TF) and its inverse document frequency(IDF). The TF is the number of occurrence of term t occurs in its context. In our case, the context is the headline or article body. Formally, the IDF can be expressed as below:

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

where N is the total number of articles in the train set. The denominator means the total number of article body contains the term t. In such way, although stop words can be removed by penalized small IDF, I removed all the stop words to save my computation power.

In the end, each headline/body can be seen as a vector where each dimension is a unique word and the value is the word's TF-IDF value. Because of the larger vocabulary size in article body than in the headline, I choose to store TF-IDF values in a dictionary.

Task3: Calculate vector cosine, KL

The cosine similarity of two vectors A, B can be calculated as :

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

In our case, A is the headline vector = ($TFIDF_{h_1}$, $TFIDF_{h_2}$, ….) where B is the article body vector = ($TFIDF_{a_1}$, $TFIDF_{a_2}$, …). Words that missing on one side can be padded with zero. Then the dimension of the headline and body vector is the total number of unique words that occur in headline and body.

As we observe the similarity equation, we can tell that the multiplication of TFIDF is zero for words that only appear in the headline or body. Thus we can sum over the multiplication of all common words' TFIDF values as the numerator. The denominator is the multiplication of two norms.

The KL can be computed as: $D_{\mathrm{KL}}(P\|Q) = -\sum_{i} P(i)\,\log\frac{Q(i)}{P(i)}$

where P is our headline vector and the Q is the body vector. Similarly, only common words are multiplied together.

Task4: Propose other features

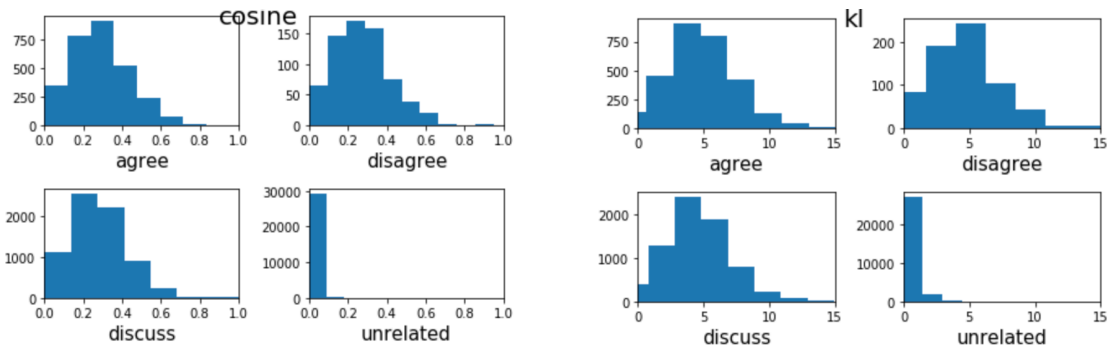
For unrelated headline and articles, I come up with Euclidean distance. It is a way of measuring vector distance in the vector space.

For discussed articles, their headlines are expected to contain uncertain words, such as maybe, might. So I add a binary value to indicate uncertain words exist in the headline.

For agree and disagree cases, keywords are expected to occur often in the article body. So I count the number of overlapping words that occur both in the article and the headline, namely 'overlap count' feature. Sometimes the value might be higher for longer articles. So I also invent the 'overlap ratio' feature, which describes the proportion of overlapping words counts in the whole counts of headline and body.

Task5: Plot feature distribution

The cosine similarity and KL similarity are expected to be important. As they measure the difference between headline and body and should be significant to separate unrelated cases. This can be proved as we draw the feature distribution for the four classes.



As we can see from the two charts, the left one is the cosine distribution and the right one is the KL distribution for four class. We can see that the the majority of unrelated case is close to zero while other classes is mainly away from the zeros. Thus, these two features are indicative for unrelated case.

Task6: Linear regression, logistic regression, gradient descent

Both linear and logistic regression applies to binary classification. So I trained four classifiers and combined the result by selecting the stance with the highest probability. I choose to use the mean squared error as the loss function. According to the gradient descent, we can easily get the update formula for weight θ as below, where X is the features and Y is prediction.

$$\theta = \theta - \alpha \mathbf{X}^T(\mathbf{X}\theta - \mathbf{Y})$$

The only difference in linear and logistic regression is there prediction equation. The linear prediction is: $Y = X\theta$ where the logistic regression is $Y = (1+ \exp(-X\theta))^{(-1)}$.

The alpha α is learning rate. It's critical for finding the optimal value. Thus, the learning rate, the total iteration and the starting guess of weight are all tuned with the hold-out data.

Task7: Analyze prediction performance

The dataset is significantly class imbalanced. We can easily get 71% of accuracy by just guessing 'unrelated' for all the data. Thus, we should penalize unrelated accuracy and boost accuracy of finding 'related' classes. I follow the official way of scoring, that is giving 0.25 point if correctly distinguish 'unrelate' and 'related' classes. Give 0.75 more if correctly classified what kind of 'related' class it is. Then we calculate the proportion of point award out of the total point. Further, I have calculated the pure accuracy for 'related' and 'unrelated' classes.

From the result, the best performance of linear logistic gives 76.08 while the logistic gives 73.37 in the test set. The linear regression is better at predicting related classes, who has 0.85 and 0.61 accuracies for unrelated and related cases. The logistic regression is better at predicting unrelated classes, who has 0.95 and 0.58 accuracies for unrelated and related cases. As we penalize the unrelated accuracy, the linear regression model performs better overall.

By hyper-parameter tuned, I found the best learning rate for the linear and logistic regression is 0.01, and 0.02. If the learning rate is too small, it's more likely to reach the local optimal not the global optimal depends on the initial guess. When the learning rate is too large, it's likely to miss the optimal even the loss converges.

Task8: Analyze feature importance

I follow the way that the 'sklearn' wrapper choose features. We begin with all features we have. It is assumed that the larger the absolute weight value is, the more the feature contributes. So in each round, we discard the least contributing feature and compute the new score until no feature left. Checking the scoring record, we can calculate feature importance based on how much they improve the total score.

Logistic	KL	overlap_ratio	cos_similarity	distance	overlap_count	uncertain
score change	10	−4	71	−0. 03	−0. 68	−8

Linear	KL	overlap_ratio	cos_similarity	distance	overlap_count	uncertain
score change	44	33	1. 6	0. 34	−0. 2	−0. 28

We can see that the ‘KL’ both contributes a lot to two models. The ‘cosine similarity’ is much helpful in the logistic model. The ‘overlap count’ and ‘uncertain’ drag down the performance. In the end, I choose the ‘KL’, ‘overlap ratio’, ‘cos similarity’ and the ‘distance’ in the linear model. And use ‘KL’, ‘distance’ and ‘overlap count’ in the logistic model.

Task9: Literature Review

Il have checked the top three resolutions in the competition. Their features are similar to my work, that used TF-IDF representation of headline and body. Their better performance is due to their more sophisticated techniques. The second and third-ranking team used a multiplayer perceptron and BoW features.

Interestingly, methods that apply to other NLP tasks are not fit for this problem, such as the word embeddings and neural networks. Instead, bag-of-words, TF-IDF, n-grams and topic modeling work well. This is mainly due to the small size of article bodies. The developed methods have different statistics compared to the FNC-1 corpus.

We can see that even for the top ranking teams, their solution are still poor in ‘Disagree’ and ‘Agree’ class. There’s much room for further research.

Task10: Other machine learning models

I tried the XGBoost model. It is especially good at finding unrelated cases, which achieves 79.36 total score, 0.99 unrelated accuracy and 0.58 other accuracy. From the result, this bagging learning algorithm is especially good at finding unrelated cases.

I also tried training with sampled data. As I experimented, all the classifiers are good at detecting unrelated cases while can’t tell much difference between ‘agree’, ‘disagree’ and ‘discuss’. So I come up with the idea of training only with related data. That is, if we train 'agree' classifier, I discard data with unrelated cases. From the result, this improves the linear regression model but not the logistic regression model.