Information Nutrition Labelling for better IR – v3

Propaganda detection at article level

Information Systems

Universität Duisburg-Essen

Shoeb Joarder & Seyedemarzie Mirhashemi

Supervisor: Dr. Ahmet Aker

This approach is based on traditional machine learning algorithms. The text is first cleaned and fed in different models such as Linear Regression, Linear SVC and K-NN etc.

1. **Data preparation**

   The dataset contains 35986 rows and 3 columns in a tab-separated format. The first column is the content of the article, second column is the article id and the third column is the gold labels. We have named the columns as the "news_text", "news_number" and "news_type" as shown in Figure 1.

	news_text	news_number	news_type
0	A recent post in The Farmington Mirror — our t...	731714618	non-propaganda
1	President Donald Trump, as he often does while...	731714635	non-propaganda
2	February is Black History Month, and nothing l...	728627182	non-propaganda
3	The snow was so heavy, whipped up by gusting w...	728627443	non-propaganda
4	Four months after the Sandy Hook School shooti...	732126660	non-propaganda

Figure 1. The Dataset

We have analysed the text, we have observed a significant imbalance in gold labels as shown in Figure 2.

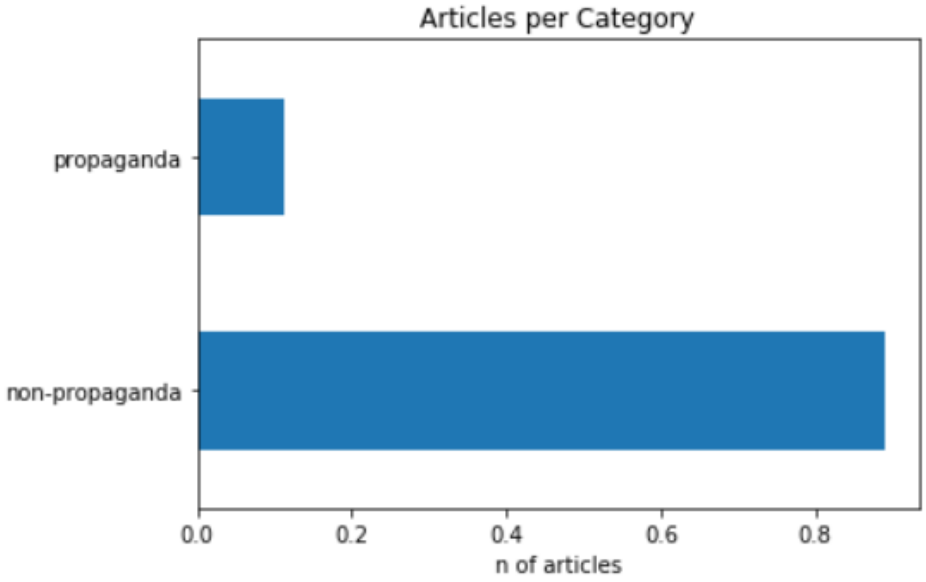

Figure 2. Articles per category

We can also visualize the words number and the word distribution per article as shown in Figure 3

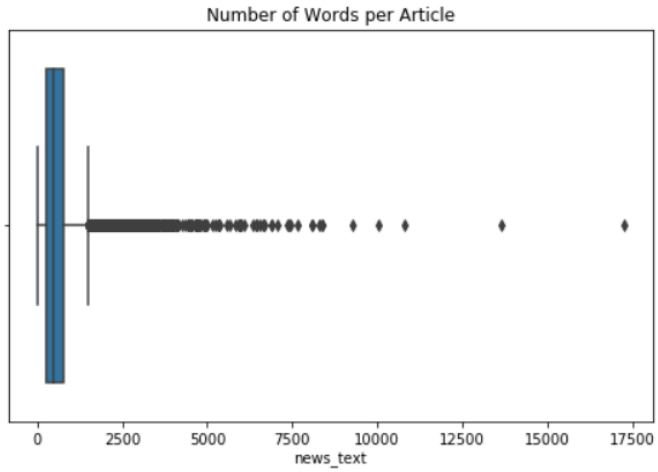

Figure 3. Number of words per article

a. Text standardization

The data preparation of the text includes few standardization techniques such as special characters, numbers and punctuation removal, each word is lemmatized and removal of stop words.

b. Data splits

The other part of the data preparation is to split the data into three samples. Since our data is imbalanced and we didn't want to bias our model so we split the "propaganda" label into three parts of which 500 is assigned to Test sample, 500 to Development sample and rest is assigned to training sample to provide as much 'propaganda' articles for training.

- Also we have run multiple experiments changing the number of "propaganda" articles to train the model, starting from equal number of propaganda and non-propaganda till the maximum number of propaganda articles available in the dataset.
- We have also tried up-scaling the propaganda article to experiment.
- We have also included the unused propaganda articles in our experiments as test sample.

For the experiment shown we have used split of 60:20:20 for train, development and testing set to evaluate our model.

c. Tokenization

Each article is presented as tokens, which represent the different words in the article. The process of tokenization transforms the raw text to sequences of indexes of the words which are found in the text. We used TF-IDF vectorization method to fit the model.

d. Tuning

We have used Randomized Search and Grid Search to optimize the hyper-parameters of our models.

2. Model

We have used different models to evaluate the best approach as shown in Figure 4.

```
# FUNCTION TO MODELLING
def modelling(X_train, y_train, X_dev, y_dev):
    model_df = pd.DataFrame(columns=['LRegression', 'LinearSVC', 'DTree', 'PAggressive',
                                     'MultinomialNB','KNN'
    #                                 ,'RForest'
                                    ],
                                    index=['True Positive','True Negative', 'False Positive',
                                           'False Negative', 'CV-Score', 'Recall',
                                           'Precision', 'Accuracy', 'F1 score'])
    vectorizer_pickle = pk.load(open('vectorizer.sav', 'rb')) # Load the TF-IDF Vectorizer
    X_vec = vectorizer_pickle.transform(X_dev) # use vectorizer to test sample
    count = 0
    #while count<7:
    while count<6:
        if count == 0:
            model = LogisticRegression(class_weight='balanced', solver='lbfgs',
                                       multi_class='ovr', random_state=40,
                                       max_iter=80)
        elif count == 1:
            model = LinearSVC(class_weight='balanced')
        elif count == 2:
            model = DecisionTreeClassifier(criterion = "entropy",
                                           random_state = 100, max_depth=3,
                                           min_samples_leaf=5)
        elif count == 3:
            model = PassiveAggressiveClassifier()
        elif count == 4:
            model = MultinomialNB(alpha=0.001)
        else:
            model = KNeighborsClassifier(n_neighbors = 5)
        model.fit(X_train, y_train)
        pred_dev = model.predict(X_vec)
        cvscore = cross_val_score(model, X_train, y_train, cv=10,
                                  scoring='accuracy') * 100
        tn, fp, fn, tp = confusion_matrix(y_dev, pred_dev).ravel()
        accuracy = metrics.accuracy_score(y_dev, pred_dev) * 100
        recall = (tp / (tp + fn)) * 100
        precision = (tp / (tp + fp)) * 100
        f1 = (2*recall*precision)/(recall+precision)
        model_df.iloc[:,count] = [tp, tn, fp, fn, cvscore.max(), recall, precision, accuracy, f1]
        count = count + 1
        print('Process finished: ', count)
    return model_df
```

Figure 4. Models

3. Experiments

We have created two models. The Model 1 is the splitting the training dataset (task1.train.txt) into 60:20:20 ratio. The results of the development set of the model 1 is shown in Figure 5.

	LRegression	LinearSVC	DTree	PAggressive	MultinomialNB	KNN
True Positive	437.000000	431.000000	136.000000	423.000000	379.000000	366.000000
True Negative	6386.000000	6608.000000	6672.000000	6633.000000	6612.000000	6614.000000
False Positive	311.000000	89.000000	25.000000	64.000000	85.000000	83.000000
False Negative	63.000000	69.000000	364.000000	77.000000	121.000000	134.000000
CV-Score	95.831403	97.498842	90.504863	97.359889	96.109310	95.275591
Recall	87.400000	86.200000	27.200000	84.600000	75.800000	73.200000
Precision	58.422460	82.884615	84.472050	86.858316	81.681034	81.514477
Accuracy	94.803390	97.804641	94.594970	98.040850	97.137696	96.984855
F1 score	70.032051	84.509804	41.149773	85.714286	78.630705	77.133825

Figure 5. Results of the development set

The model was tested using the test set and the results are shown in Figure 6.

	LRegression	LinearSVC	DTree	PAggressive	MultinomialNB	KNN
True Positive	437.000000	423.000000	141.000000	414.000000	370.000000	353.000000
True Negative	6404.000000	6604.000000	6670.000000	6641.000000	6614.000000	6612.000000
False Positive	294.000000	94.000000	28.000000	57.000000	84.000000	86.000000
False Negative	63.000000	77.000000	359.000000	86.000000	130.000000	147.000000
CV-Score	95.831403	97.498842	90.504863	97.452524	96.109310	95.275591
Recall	87.400000	84.600000	28.200000	82.800000	74.000000	70.600000
Precision	59.781122	81.818182	83.431953	87.898089	81.497797	80.410023
Accuracy	95.040289	97.624340	94.623507	98.013337	97.026952	96.762990
F1 score	70.999188	83.185841	42.152466	85.272915	77.568134	75.186368

Figure 6. Results of the test set

The first model is tested using the Test dataset (task1.test.txt), which contains 10163 annotated articles. The results using the Test Dataset is shown in Figure 7.

	LRegression	LinearSVC	DTree	PAggressive	MultinomialNB	KNN
True Positive	1063.000000	1050.000000	16.000000	1042.000000	896.000000	753.000000
True Negative	8599.000000	8859.000000	8929.000000	8895.000000	8855.000000	8847.000000
False Positive	350.000000	90.000000	20.000000	54.000000	94.000000	102.000000
False Negative	148.000000	161.000000	1195.000000	169.000000	315.000000	458.000000
CV-Score	95.831403	97.498842	90.504863	97.545160	96.109310	95.275591
Recall	87.778695	86.705202	1.321222	86.044591	73.988439	62.180017
Precision	75.230007	92.105263	44.444444	95.072993	90.505051	88.070175
Accuracy	95.098425	97.529528	88.041339	97.805118	95.974409	94.488189
F1 score	81.021341	89.323692	2.566159	90.333767	81.417537	72.894482

Figure 7. Results of the Test Dataset of Model 1

We have created another model (Model 2) using the entire Train dataset (task1.train.txt) to train our model and use the Test dataset (task1.test.txt) to evaluate our results as shown in Figure 8.

	LRegression	LinearSVC	DTree	PAggressive	MultinomialNB	KNN
True Positive	1114.000000	1109.000000	16.000000	1101.000000	982.000000	753.000000
True Negative	8676.000000	8916.000000	8929.000000	8934.000000	8893.000000	8898.000000
False Positive	273.000000	33.000000	20.000000	15.000000	56.000000	51.000000
False Negative	97.000000	102.000000	1195.000000	110.000000	229.000000	458.000000
CV-Score	96.914953	98.499166	94.774875	98.916064	97.081712	96.442468
Recall	91.990091	91.577209	1.321222	90.916598	81.090008	62.180017
Precision	80.317231	97.110333	44.444444	98.655914	94.605010	93.656716
Accuracy	96.358268	98.671260	88.041339	98.769685	97.194882	94.990157
F1 score	85.758276	94.262643	2.566159	94.628277	87.327701	74.739454

Figure 8. Results of the Test Dataset of Model 2

4. Conclusion

The best model for our experiment is the Linear Support Vector Classifier giving F1-score of 84.50% in development set, 83.18% in test set and 89.32% in Test dataset for Model 1. For Model 2, the F1 scored a 94.26% and with one of the better Precision & Recall. We have run multiple experiments with setups mentioned in 1b such as up-scaling non-propaganda sets in training to balance and using unused propaganda labelled articles in evaluation which didn't improve the F1-score and Recall. The experiment for Model 1 and Model 2 showed significantly good results in Linear SVC to detect propaganda articles. Since the data is imbalanced, Model 2 was trained more with propagandistic article which may helped to learn important features.