Group Proj

Reference webpage:

https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.CountVectorizer.html  
https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html  
https://blog.csdn.net/liulina603/article/details/78676723

My task: write why I chose specific parameters for the countVectorizer and Logistic regression.

Also do the task 4 (about logistic regression)

From PA4: Logistic regression model provides a convenient way to analyse to which extent features are predictive for each class by expecting their weights. Use (and adapt) the provided example in 'PA4\_explained.html' to analyse which features were useful for each class for the baseline model and your own model.

My part to Olga:

In the *ExtractFeatures* function, we first preprocessed the snippet by deleting the punctuation. Our original features used for our improved model were *BOW(left, middle, right) feature, entity feature, bigram feature*, *syntax feature*, *pos tags feature* and *direction feature*. For the baseline model, the only feature we used was *BOW feature*.

The parameters we set in CountVectorizer were

stop\_words=**"english"**, (By default,‘None’)

If ‘english’, a built-in stop word list for English is used. Since the language in our text is English, we set this setting as “english” to elimate those stop words. In the baseline model, we didn’t eliminate the stop words. This setting will make difference in the result score.

decode\_error=**" ignore"**, (By default,‘strict’)

Instruction on what to do if a byte sequence is given to analyze that contains characters not of the given encoding. By default, it is ‘strict’, meaning that a UnicodeDecodeError will be raised. We use ignore to omit the error, however this wont affected the result if the text doesn’t have decoding error.

lowercase=**True (by default)**

Convert all characters to lowercase before tokenizing. For our improved model, we use this setting the same as we used in the baseline model.

**Logistic regression** is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

The parameters we set in LogisticRegression were

penalty=**'l2'**, **(by default)**

Used to specify the norm used in the penalization. The ‘newton-cg’, ‘sag’ and ‘lbfgs’ solvers support only l2 penalties. The ‘liblinear’ solver supports both ‘l1’, ‘l2’ penalties. Ususally, we choose ‘l2’

solver=**'newton-cg'**, (By default,‘liblinear’)

Algorithm to use in the optimization problem.

For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones. Usually, a dataset containing more than 100,000 entries will be considered as large.

For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss; ‘liblinear’ is limited to one-versus-rest schemes.

‘newton-cg’, ‘lbfgs’ and ‘sag’ only handle L2 penalty, whereas ‘liblinear’ and ‘saga’ handle L1 penalty.

multi\_class=**'multinomial'**, (By default,‘ovr’ (one-versus-rest))  
 The default one-versus-rest has some problems. Firstly, the scale of the confidence values may differ between the binary classifiers. Second, even if the class distribution is balanced in the training set, the binary classification learners see unbalanced distributions because typically the set of negatives they see is much larger than the set of positives.

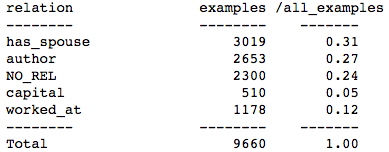
In our training set, the class distribution is unbalanced, especially for “capital”. Thus, we use multinominal. Multinomial logistic regression deals with situations where the outcome can have three or more possible types that are not ordered.

This setting will make difference in the result score.

class\_weight={0:0.190, 1:0.185, 2:0.22, 3:0.195, 4:0.210} (By default,‘None’) Weights associated with classes in the form {class\_label: weight}. By default, all classes are supposed to have equal weight.

In our case 0 is ‘NO\_REL’, 1 is ‘author’, 2 is ‘capital’, 3 is ‘has\_spouse’, 4 is ‘worked\_at’. Basically, we want to increase the weight for ‘capital’ and ‘worked\_at’, but decrease the weight for ‘author’ and ‘has\_spouse’. Therefore, based on the sample size, the class weights we assigned are 0.190, 0.185, 0.220, 0.195, 0.210 respectively. These numbers are getting empirically.

This setting changes the result score significantly by comparsion to other parameters.



(When using Logistic regression, we found the *bigram feature* is not useful for improvement. So, our final setting features are *BOW(left, middle, right) feature, entity feature*, *syntax feature*, *pos tags feature* and *direction feature*

This section can be deleted if it is cumbersome or making no sence)

By the above setting, we got 0.811 for macro-average f-score.

And the top 3 Informative features for each class are:

Class NO\_REL best:

(0.43176208867107707, 'include')

(0.5059480588471347, 'india')

(0.8615780297173694, 'cconj')

Class author best:

(1.1916553216770622, 'punct')

(1.3102908212272557, 'book')

(1.4893777066054479, 'novel')

Class capital best:

(0.7066779789541036, 'city')

(0.8370324628792728, 'born')

(1.5017116999160416, 'capital')

Class has\_spouse best:

(1.635879678194315, 'husband')

(1.761909592679575, 'cconj')

(2.213480471284747, 'wife')

Class worked\_at best:

(0.9863012704135359, 'founder')

(0.9958376533017631, 'company')

(1.1880728920959907, 'professor')