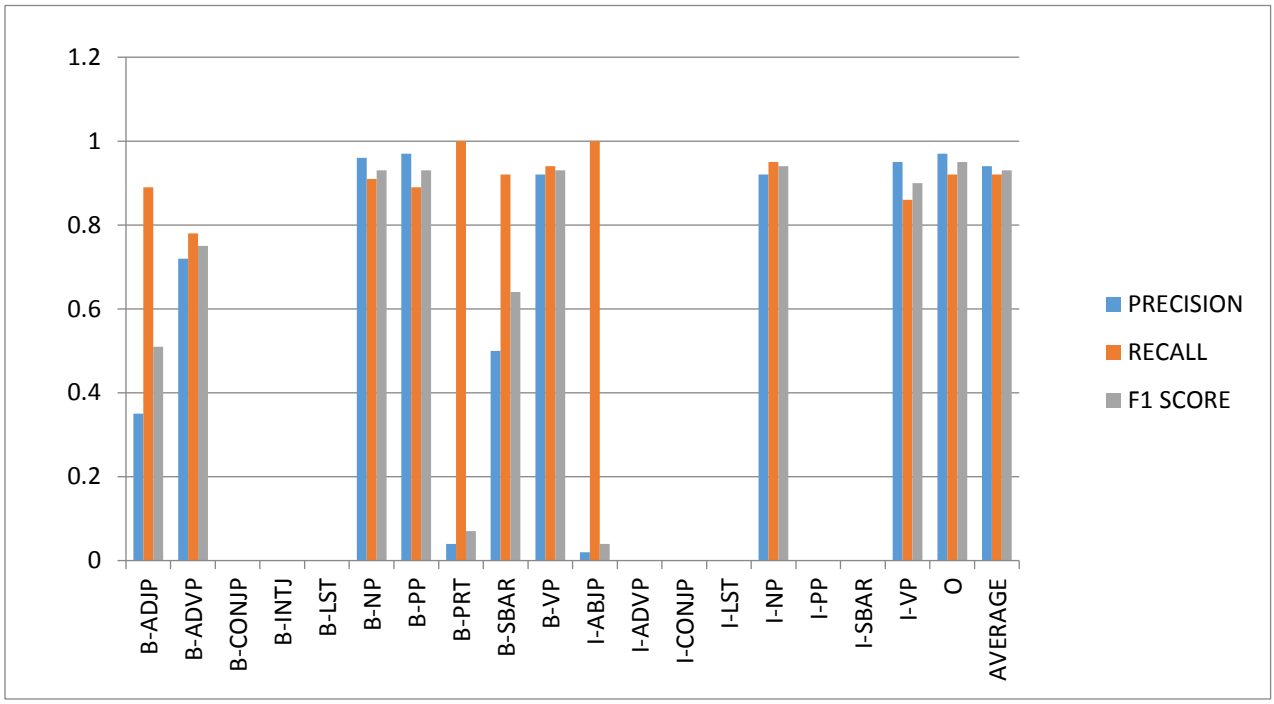
An Overview Of The Results

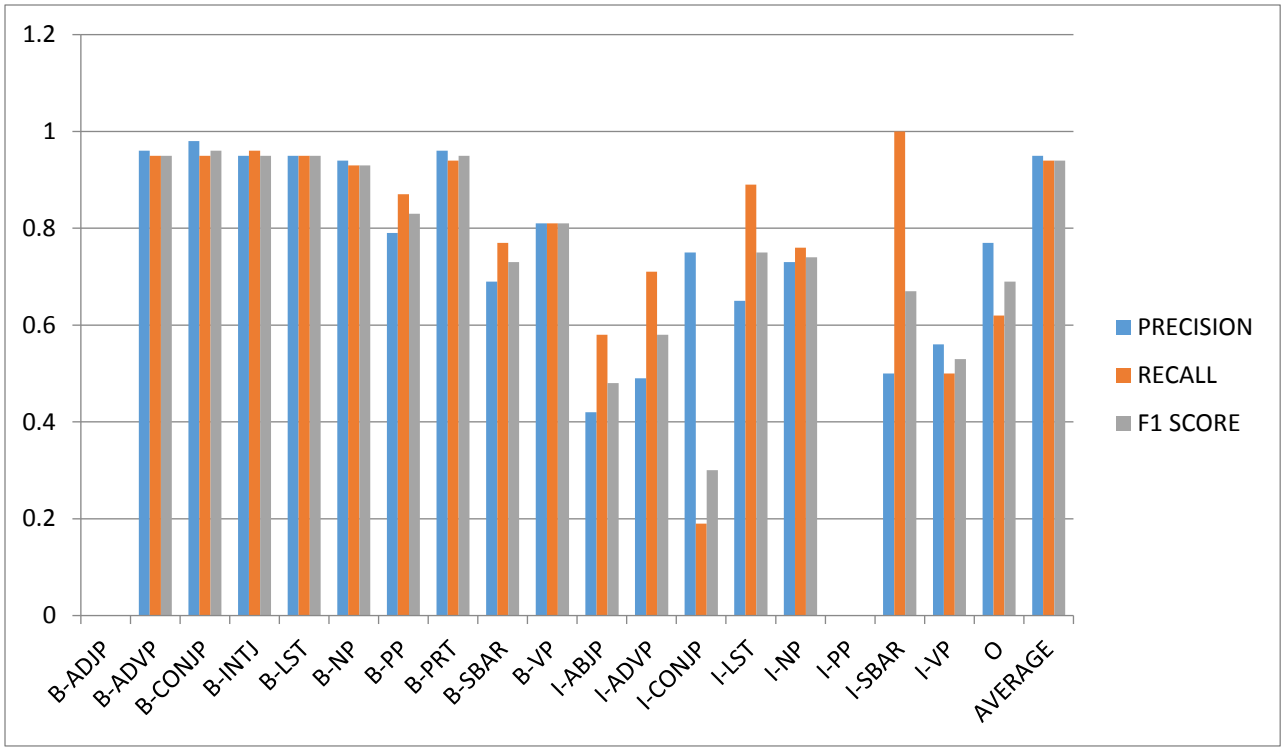
The following graphs show precision score , recall score and F1 score values for each label type when the classifiers are applied. The "AVERAGE" column (last column) on the graphs is the average of all label type values.

"Graph 1" is created by using Multinomial Naive Bayes.



Graph 1

"Graph 2" is produced with Linear SVC strategy of Support Vector Machine.



Graph 2

If we examine “Graph 1”, we see that Multinomial Naive Bayes classifier does not have any prediction for “B-CONJP”, “B-INTJ”, “B-LST”, “I-ADVP”, “I-CONJP”, “I-LST”, “I- PP” and “I-SBAR” label types. However, on that graph, the recall score of “B-PRT” and “I-ABJP” label types is 1 but, their decision scores are closer to 0. It is comprehended that all “B-PRT” and “I-ABJP” label types in the test data is predicted correctly but, the number of prediction for them is much bigger than the number of the correct prediction.

Now, we see “Graph 2”. Linear SVC strategy of the Support Vector Machine classifier have no prediction. In addition, if “I-CONJP” label type is observed, its precision score is bigger than the recall score of it and also, there is a visible difference between its precision score and its recall score. In here, we can understand that most of what is estimated as “I-CONJP” type is true but, most of what is “I-CONJP” type is not estimated correctly.

If we compare the classifiers according to “AVERAGE” column on the graphs, we see that the precision and recall scores of Linear SVC strategy of the Support Vector Machine classifier is 1% more than Multinomial Naive Bayes. By looking that, we can reach the conclusion that according to the model and test result, Linear SVC strategy of the Support Vector Machine is 1% better than Multinomial Naive Bayes in terms of finding the correction in the prediction and the correct prediction of the types.

How Is It Programmed

In that project, we have developed the model by using five features. These are current word, current part-of-speech, previous word, previous part-of-speech and next part-of-speech.

Firstly, the list of the set of features is created by loading from the “train.txt” file. You can see the example of that:

[...,

{'pos': 'VBD', 'word-1': 'inflation', 'pos+1': 'NN', 'word': 'was', 'pos-1': 'NN'}, {'pos': 'NN', 'word-1': 'was', 'pos+1': 'IN', 'word': 'forecast', 'pos-1': 'VBD'},

...]

That format is not executed by the “fit(X,y)” methods of the classifiers so, it should be vectorized. Vectorilization is made with an object created by “DictVectorizer”. ““DictVectorizer” transforms lists of feature-value mappings to vectors. The created object learns the list of feature name and indices mappings by calling “fit_transform(X)” method.”[1] That method returns the format of [n_samples n_feature] as shape. n_samples is an array and n_feature is a sparse matrix. The output is used as an training vector in “fit(X,y)” method of the classifiers. A supervised classifier need to have a label list to develop a model.

The list of the label is produced from “train.txt” as the following:

[..., 'O\n', 'B-VP\n', 'I-VP\n', ...]

It can be put in the fit(X,y) method of Multinomial Naive Bayes classifier but, SVM does not accept that format. Hence, it is changed to a numeric list. We have made it by numerating each label type with a different number while reading datas from file.

After the learning is generated, the list of test features and labels are configured as the above. While configuring the list of test features, the object created by “DictVectorizer” vectorizes the features by calling “transform(X)” method. It is the reason of that the object has a trained data for features so, test features should be vectorized according to the trained data. Then, the vectorized list of test features is put in “predict(X)” method of the classifier and a prediction for test label list is handled. That prediction label and actual label is given as parameters to the “accuracy_score(X,Y)” method so, precision score, recall score and F1 score are found.

Classifier Methods

We have used two classifiers for that study. These are Multinomial Naive Bayes and Support Vector Machine. Now, we see how to work those classifiers.

a. How does Multinomial Naive Bayes work?

"While infering a classification,`denoting the classes as *Ci* and any relevant features *F*, the probability of a given class is given by Bayes Theorem,

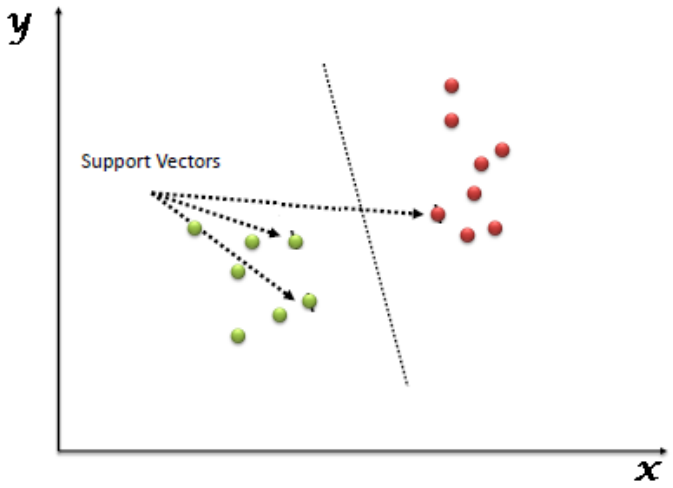
$$p(C_i \mid \vec{F}) = \frac{p(\vec{F} \mid C_i)p(C_i)}{\sum_j p(\vec{F} \mid C_j)p(C_j)}$$

Now all we need to do is model the class likelihoods, $p(\vec{F} \mid C_i)$

When a feature is discrete but not binary (or continuous but approximated by such discrete values) a common choice of likelihood is the multinomial distribution, hence Multinomial Naive Bayes. This is a direct generalization of the Binomial Naive Bayes model."[2]

b. How does Support Vector Machine work?

"In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).



Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line)."[3]

We have selected Linear SVC strategy of Support Vector Machine classifier. It is the reason of that "it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples"[4].

Confusion Matrix Of Multinomial Naive Bayes

	B-ADJP	B-ADVP	B-CONJP	B-INTJ	B-LST	B-NP	B-PP	B-PRT	B-SBAR	B-VP	I-ADJP	I-ADVP	I-CONJP	I-LST	I-NP	I-PP	I-SBAR	I-VP	O
B-ADJP	155	4	0	0	0	2	0	0	0	0	11	0	0	0	0	0	0	3	0
B-ADVP	37	622	8	1	0	23	12	16	5	27	3	18	6	0	13	0	0	3	3
B-CONJP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
B-INTJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-LST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-NP	123	79	0	1	3	11879	15		45	161	21	24	0	2	575	1	1	21	57
B-PP	12	68	0	0	0	48	4657	86	217	14	3	4	7	0	71	35	1	9	11
B-PRT	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
B-SBAR	0	0	0	0	0	14	0	0	265	0	0	0	0	0	0	0	2	0	6
B-VP	8	12	0	0	0	23	80	1	4292	2		0	0	0	56	0	0	61	21
I-ABJP	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
I-ADVP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-CONJP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-LST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-NP	45	23	0	0	0	343	5	0	0	57	95	13	0	0	13247	1	0	11	52
I-PP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-SBAR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-VP	55	50	0	0	0	68	21	0	0	102	21	21	0	0	25	2	0	2514	40
O	3	8	1	0	2	22	21	0	2	5	8	9	0	0	389	9	0	24	5990

Confusion Matrix Of Support Vector Machine

	B-ADJP	B-ADVP	B-CONJP	B-INTJ	B-LST	B-NP	B-PP	B-PRT	B-SBAR	B-VP	I-ADJP	I-ADVP	I-CONJP	I-LST	I-NP	I-PP	I-SBAR	I-VP	O
B-ADJP	[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
B-ADVP	[2	11867	10	393	38	12	10	46	38	43	5	5	0	0	0	0	1	0	0]
B-CONJP	[0	25	4702	31	16	8	83	10	1	31	3	1	0	4	19	0	0	2	1]
B-INTJ	[0	331	13	13621	59	15	0	127	25	10	18	43	0	0	0	0	0	0	0]
B-LST	[0	49	15	46	4430	80	0	17	11	9	0	5	0	0	0	0	0	0	0]
B-NP	[0	40	5	13	78	2477	0	12	13	5	3	10	0	0	0	0	0	0	0]
B-PP	[0	24	23	0	0	0	424	9	1	7	0	1	0	0	0	0	0	0	0]
B-PRT	[0	26	18	231	3	19	1	5941	4	21	4	5	0	11	0	2	0	2	0]
B-SBAR	[0	21	3	9	3	21	0	3	301	16	1	11	0	0	0	0	0	0	0]
B-VP	[0	35	13	10	31	7	3	4	35	701	12	2	0	0	10	3	0	0	0]
I-ABJP	[0	2	1	6	0	2	0	4	4	5	37	2	0	1	0	0	0	0	0]
I-ADVP	[0	0	0	15	0	5	0	1	4	2	6	82	0	0	0	0	0	0	0]
I-CONJP	[0	0	0	0	0	0	13	0	0	0	0	0	3	0	0	0	0	0	0]
I-LST	[0	0	1	0	0	0	0	0	0	0	0	0	1	31	0	0	0	0	2]
I-NP	[0	0	5	0	0	0	0	5	1	13	0	0	0	0	77	0	0	0	0]
I-PP	[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
I-SBAR	[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0]
I-VP	[0	0	2	0	0	0	1	0	0	2	0	0	0	0	0	0	0	5	0]
O	[0	2	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	10]

Sources

[1]

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html

[2] https://www.quora.com/How-does-multinomial-Naive-Bayes-work

[3]

https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/

[4] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html