LING 406: Intro to Computational Linguistics
Spring 2021

Assignment #3: *Part of Speech Tagging*

Issued: Mar. 25, 2021
Due: Apr. 1, 2021 by 11:59PM
Credits: 50 points

This assignment presents you with a POS tagging problem (*a multi-class classification problem*) which will be accomplished with a supervised Machine Learning brute-force approach (i.e., considering a context window of 7 word-tokens: three to the left and 3 to the right of target word to be POS tagged). The data file needed to implement this approach is provided in the assignment folder (pos-eng-5000.data.csv) on github.

All code (using Scikit-Learn or a machine learning library of your choice) must be your own. You will be required to use your toolkit of choice to accomplish the following:

> Estimate the precision, recall, accuracy, and F-measure on the POS tagging task using 5-fold cross-validation. You need to do this for two machine learning models: Naïve Bayes and Decision Tree classifier.
>
> For each classifier compute the contribution of each attribute (this activity is called 'feature contribution') using the "leave one out" approach explained in class. The difference in performance should be captured in a table: as columns you will have the attributes and as rows you have the difference in performance (precision, recall, accuracy, and F-measure): (performance with all the features) - (performance with all the features minus the current feature).

Feature engineering: There are many ways you can represent the context of a target word (feature engineering) and the purpose of this assignment is to give you the opportunity to explore such feature representations of your choice, especially that during the lab sessions, Chase and Hayley, our TAs, took you through the process.

However, as for many tasks we looked at before, you have to start with a baseline system. This should be based on a bag-of-words representation -- a simple set of features to represent the context of a target word (i.e., term). For instance, for each data point, create a dictionary of features describing the context window the term is part of. Then, convert the dictionary features to vectors and use this vector representation to train and test your models. The implementation of your baseline model is worth 20 points.

After this, you are asked to improve over the baseline classifier. You may choose to add two more features to the baseline feature set, or you may want to use a completely different representation of your choice (which might result in a new classifier). This is called the improved classifier [10 points].

Write a report and answer the following questions (be sure to include your table) [5 points each]:

Naïve Bayes with label encoder

Performance	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7
Diff Accuracy	33.0%- 33.7% = **0.7%**	33.0%- 33.3% = **-0.3%**	33.0%- 33.2% = **-0.2%**	33.0%- 19.5% = **13.5%**	33.0%- 32.2% = **0.8%**	33.0%- 33.2% = **-0.2%**	33.0%- 32.8% = **0.2%**
Diff Precision	20.9%- 22.7%= **-1.8%**	20.9%- 20.0%= **0.9%**	20.9%- 20.3%= **0.6%**	20.9%- 4.5%= **16.4%**	20.9%- 21.3%= **-0.4%**	20.9%- 21.2%= **-0.3%**	20.9%- 21.2%= **-0.3%**
Diff Recall	22.2%- 23.1%= **-0.9%**	22.2%- 22.1%= **0.1%**	22.2%- 22.8%= **-0.6%**	22.2%- 7.4%= **14.8%**	22.2%- 22.3%= **-0.1%**	22.2%- 23.0%= **-0.8%**	22.2%- 22.2%= **0%**
Diff F-measure	19.7%- 20.6%= **-0.9%**	19.7%- 19.4%= **0.3%**	19.7%- 19.9%= **-0.2%**	19.7%- 5.0%= **14.7%**	19.7%- 19.6%= **0.1%**	19.7%- 20.4%= **-0.7%**	19.7%- 19.7%= **0%**

Decision tree with label encoder

Performance	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7
Diff Accuracy	60.7%- 61.3% = **-0.6%**	60.7%- 60.9% = **-0.2%**	60.7%- 61.0% = **-0.3%**	60.7%- 22.4% = **38.3%**	60.7%- 61.9% = **-1.2%**	60.7%- 61.4% = **-0.7%**	60.7%- 61.0% = **-0.3%**
Diff Precision	54.1%- 56.6%= **-2.5%**	54.1%- 55.1%= **-1%**	54.1%- 54.6%= **-0.5%**	54.1%- 13.7%= **40.4%**	54.1%- 55.7%= **-1.6%**	54.1%- 55.2%= **-1.1%**	54.1%- 56.8%= **-2.7%**
Diff Recall	55.3%- 56.6%= **-1.3%**	55.3%- 55.4%= **-0.1%**	55.3%- 55.6%= **-0.3%**	55.3%- 13.1%= **42.2%**	55.3%- 57.0%= **-1.7%**	55.3%- 55.4%= **-0.1%**	55.3%- 55.9%= **-0.6%**
Diff F-measure	52.9%- 55.1%= **-2.2%**	52.9%- 53.8%= **-0.9%**	52.9%- 53.8%= **-0.9%**	52.9%- 13.0%= **39.9%**	52.9%- 55.1%= **-2.2%**	52.9%- 53.9%= **-1%**	52.9%- 55.0%= **-2.1%**

Naïve with One hot encoder

Performance	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7
Diff Accuracy	47.9%- 49.7% = **-1.8%**	47.9%- 49.8% = **-1.9%**	47.9%- 43.7% = **4.2%**	47.9%- 18.7% = **29.2%**	47.9%- 45.4% = **2.5%**	47.9%- 48.7% = **-0.8%**	47.9%- 50.3% = **-2.4%**
Diff Precision	35.6%- 37.2%= **-1.6%**	35.6%- 38.5%= **-2.9%**	35.6%- 31.9%= **3.7%**	35.6%- 11.8%= **23.8%**	35.6%- 34.3%= **1.3%**	35.6%- 35.9%= **-0.3%**	35.6%- 38.9%= **-3.3%**
Diff Recall	35.7%- 39.7%= **2%**	35.7%- 40.4%= **-4.7%**	35.7%- 34.8%= **0.9%**	35.7%- 12.5%= **23.2%**	35.7%- 35.6%= **0.1%**	35.7%- 37.8%= **-2.1%**	35.7%- 40.4%= **-4.7%**
Diff F-measure	33.2%- 36.1%= **-2.9%**	33.2%- 36.7%= **-3.5%**	33.2%- 30.6%= **2.6%**	33.2%- 10.9%= **22.3%**	33.2%- 32.3%= **0.9%**	33.2%- 34.1%= **-0.9%**	33.2%- 36.8%= **-3.6%**

Decision tree with One hot encoder

Performance	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7
Diff Accuracy	62.2%- 62.8% = **-0.6%**	62.2%- 63.0% = **-0.8%**	62.2%- 59.4% = **2.8%**	62.2%- 31.5% = **30.7%**	62.2%- 62.5% = **-0.3%**	62.2%- 62.5% = **-0.3%**	62.2%- 63.0% = **-0.8%**
Diff Precision	63.2%- 62.8%= **0.4%**	63.2%- 63.0%= **0.2%**	63.2%- 64.6%= **-1.4%**	63.2%- 19.9%= **43.3%**	63.2%- 62.4%= **0.8%**	63.2%- 61.2%= **2%**	63.2%- 64.6%= **-1.4%**
Diff Recall	55.0%- 56.0%= **-1.0%**	55.0%- 54.9%= **0.1%**	55.0%- 54.4%= **0.6%**	55.0%- 17.2%= **37.8%**	55.0%- 55.5%= **-0.5%**	55.0%- 55.3%= **-0.3%**	55.0%- 56.9%= **-1.9%**
Diff F-measure	57.0%- 57.5%= **-0.5%**	57.0%- 57.1%= **-0.1%**	57.0%- 56.6%= **0.4%**	57.0%- 17.2%= **9.8%**	57.0%- 56.6%= **0.4%**	57.0%- 56.3%= **0.7%**	57.0%- 58.8%= **-1.8%**

1. Which is the best machine learning model (classifier) for this task (for both the baseline and the improved classifier)? You need to discuss this per metric used to compute the performance.

It seems from the accuracy metric that the decision tree classifier (60.7%) is a better machine learning model than naïve Bayes (33.0%), given this task. The precision metric for the decision tree classifier (54.1%), which is greater than the precision metric (20.9%). The recall metric is decision tree classifier (55.3%), which is greater than the precision recall metric (22.2%). The F-measure is decision tree classifier (52.9%), which is greater than naïve Bayes classifier (19.7%). This sums up that the decision tree classifier is a better machine learning model compared to naïve Bayes.

2. Which features contributed the most to the performance (precision, recall, accuracy, and F-measure)? Which contributed the least? (you need to do this for each machine learning model and for each classifier considered here)

Attribute 4 contributed the most to the performance, this can be concluded from looking at the differences in all the metrics after the removal of attribute 4, which yields in extremely high differences showing that attribute 4 contributes largely to the performance of both classifiers, in both label encoder and one hot encoder.

3. How good is this feature set for this task (for each classifier)? (based on the answer at Question 2)

Speaking with respect to the attribute 4, we can see that it is extremely important to each classifier, as we see in the data, label encoder shows ~13-17% decrease in naïve bayes in each metric which is about 0-9.4 times the other metrics. Label encoder also shows ~9-44% decrease in metrics without attribute 4, which is ~44-90 times the other metrics. The one hot encoder shows a ~22-30% decrease in metrics without attribute 4 for the naïve bayes classifier and a ~9-44% decrease in decision tree classifier. The significant differences signify that attribute 4 is extremely important to the data yielding in higher accuracy.

I initially though that 5000 lines was a huge data set, but after having a conversation with my TA, Hayley, I realized that datasets in nlp computation need to be way more that 5000 line and more thorough than the current dataset, since language is so complex. The size of the current dataset does, however, yield in slightly faster computation and is a good way to learn the techniques. According to this data set it seems that on hot encoder worked better on this data set compared to the label encoder, however both representations are pretty simple, so it is possible that we do not see the same results with a much larger dataset. Additionally, label encoder is mostly used for labels, so using it for this purpose is a slight extension of the representation. It is possible that the one hot encoder, worked better because it takes care of null or empty columns which helps with the dataset.

4. If you had more time to work on this problem and do it perhaps more efficiently (in terms of performance), which features/text representation would you choose? (write 1-2 short paragraphs about the features sets you might want to try for this problem).

One way I would like to try and classify data is using the bag of words representation data. Bag of representation is really good for this data set and the way it works is that it calculates the frequency of unique words in the data set. This is a good way to represent data that we can weigh words based on their occurrence and repentance in the dataset. This does not preserve word order, or meaning of words, but rather best tells which words are most frequent and tags them accordingly. This may not be the best representations if we are trying to look beyond frequencies of words, but this dataset was open ended and not meant to be analyzed for some particular reason, hence this method would work well. The method is slightly more complex, and may require more time for training and testing.

In addition to that, I would like to try the tf-id vector, had I had more time to work on it. I tried implementing the tf-id vector, but consistently kept getting extremely low values for precision, accuracy, and recall. The tf-id does take longer to train and test compared to the other methods. The reason why tf-id may work well, is because it us able to deal with the problems that bag of words representation have. It allows for words that are more meaningful, and more frequent to have more influence than words, that are rare, bag of words representation can weigh some words unintentionally more. This also preserves the sentence structure, so this would be a great representation.

Extra credit: [15 points]

Train and test the POS tagger (both baseline and improved classifiers) with CRF (Conditional Random Fields) or LSTM (Long Short-Term Memory) model. You have to compare the performance of your system with those obtained with the previous machine learning models.

Deliverables:

- Provide a README.md file including a detailed note (i.e., one paragraph) about the functionality of each of the programs, and complete instructions on how to run them; make sure you include your name in each program and in the README file; make sure all your programs run correctly.
- Provide an answer.pdf file where you should include the results obtained with your table and the answers to the questions outlined above.
- Submit all of your Python code as Jupyter Notebook(s). Your code must be extensively commented using programming best practice.
- Using GitHub add, commit, and push your source code files, the README.md file and the answer.pdf file. 5 points will be deducted if any of these deliverable files is missing.