LING 406: Intro to Computational Linguistics

Spring 2022

Assignment #3: *Part of Speech Tagging*

Issued: Mar. 24, 2022
Due: Apr. 07, 2022, by 11:59PM
Credits: 50 points

Introduction

The purpose of this assignment is to present you with (1) a real-world part-of-speech tagging problem and (2) an opportunity to put into practice some of the basic machine learning techniques discussed in our class. The data for this assignment come from the 2017 POLEVAL Shared Task, a SemEval-inspired initiative which asked participants to engage with several subtasks related to the morphologically rich Polish language. Due to the time constraints of this assignment, you will only be responsible for accomplishing a part of Subtask C, a *multi-class classification problem* using a supervised machine learning approach.

To approximate the conditions of a shared task such as POLEVAL, we have divided the provided gold standard (segmented and manually POS-tagged) data into *train*, *validation (also known as development)*, and *test* sets which contain 60%, 20%, and 20% of the 104,147 example sentences, respectfully. You have been given with the *train* and *validation* sets (in the src/Data/ folder), which you should use for your initial testing. We will provide you with access to the withheld *test* set two days before the submission deadline (on Apr. 05, 2022) so that you can calculate and report performance metrics for your final system before the submission.

The data come in zipped XCES format (you can read about this XML-based standard in Ide, Bonhomme, and Romary's (2000) paper here). Each *chunk* listed in the data represents a sentence and is made up of multiple *tok* (token) elements which in turn contain both *orth* (surface orthographic form) and *lex* (pertaining to lexicographic information) elements. This latter element provides a base form (*lemma*) for each token and a *ctag*, a colon-separated POS tag which has information about grammatical class (part-of-speech) in addition to information about grammatical categories related to the number, case, gender, and person of the token. A summary breakdown of this POS tagset encoding scheme from the National Corpus of Polish Cheatsheet can be found here. A more robust treatment of this tagset is outlined in the original 2003 paper by Przepiórkowski & Wolínski here.

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

> Use the *orth* sub-elements from the *train* data and their associated POS tags to predict the POS tags from the *validation* data. Estimate the Precision, Recall, Accuracy, and F-measure of your model predictions. Do this for two machine learning algorithms (models) of your choice. For example, you may choose to implement Naïve Bayes and Decision Tree classifiers.
>
> When the *test* data becomes available to you, use your same parameters for model fitting as before, but use both the *train* and *validation* data to train a new model. Use this new model to predict the POS tags from the *test* data. Estimate your performance metrics again for this withheld data set. You will report these metrics for both the *validation* and *test* withheld sets in your final report.

Baseline System (10 points)

As for many of the tasks that we have looked at before, you will need to start with a baseline system for each of these experiments, against which to compare any improvements you might make. This should be based on a bag-of-words representation of the data. Use a simple set of features to represent the context of a target term (i.e., the *orth* sub-element's text). For instance, for each *tok* element, create a dictionary of features describing a context window of a few terms to the left and right of the token. Then, convert the dictionary of features to numerical vectors and use this vector representation to train and test your models.

Improved System (10 points)

Next, you are tasked with improving upon your baseline classifier. You may choose to use different preprocessing techniques from your Baseline System. Or you may decide to implement a different featurization paradigm. You may want to use a completely different representation of your choosing. The goal here is to improve upon your original system in terms of the performance metrics listed above, due to the smart design decisions you will make. Feel free to read through the proceedings from PolEval 2017 to look for ideas which strike you as interesting.

Feature Engineering (10 points)

There are many ways you can represent the context of a target word and one purpose of this assignment is to give you the opportunity to explore such feature representations of your choosing.

For each of your selected machine learning algorithms, using your improved system, compute the contribution of each attribute you have decided to use. If, for example, you follow the in-class Machine Learning tutorial's "brute-force" approach (whereby you consider a context window of ~7 tokens (3 to the left and 3 to the right of a given target word to be classified/tagged), each of the columns surrounding the target term would be considered a unique attribute. In this example, using the "leave one out" approach, you would run your model 7 times, leaving out a different attribute each time. After every run, the difference in the performance metrics should be captured. You should present these differences in your final report in a table. As columns, list the attributes of interest and as rows show the difference in performance given as:

(performance with all features) - (performance with all features minus the current feature)

If you decide not to use the "brute force" approach, reach out to the TAs to discuss what constitutes an attribute which can be left out for feature engineering purposes.

Report (20 points)

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

1. Which is the best machine learning algorithm (classifier) for this task (for both the baseline and the improved classifier)? You need to discuss this per metric used to compute the performance.
2. Which attributes contributed the most to each of the performance metrics for your improved model? Which contributed the least? (Write about this for each algorithm considered.)
3. How good is your feature set for this task (for each algorithm)? (Base your response to this question to your answer from Question 2)
4. If you had more time to work on this problem and do it 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 and why.

Extra Credit #1 (optional) (10 points):

The data provide you with a surface orthographic form **and** a lemma for each token. The data's tagset also encodes for both Polish grammatical class (POS tag) **and** grammatical categories. There are, therefore, four distinct pairings of predictor variables and output classes which can be tested for:

- Use the *orth* form to predict a token's POS tag
- Use the *orth* form to predict a token's full *ctag*
- Use the *lemma* to predict a token's POS tag
- Use the *lemma* to predict a token's full *ctag*

You have already analyzed the first of these pairings, using the *orth* form to predict unseen tokens' POS tags. Re-run your analysis with the other three pairings on your Improved System using both of your selected machine learning algorithms. Compare these four pairings and algorithms in terms of the performance metrics you calculated.

Make a table of these results and append it to your final report document and write 1-2 paragraphs on the differences you noted in model efficacy. Which pairing yielded the best metrics? Why do you think that is? Label this section in the report clearly so that we know you are trying for the extra credit points.

Extra Credit #2 (optional) (10 points):

Train and test the POS tagger (for both the Baseline and the Improved Systems) with an implementation of the CRF (Conditional Random Fields) algorithm or an implementation of an LSTM (Long Short-Term Memory) algorithm. Compare the performance metrics of your extra credit system with those obtained with your Baseline and Improved Systems. Make a table of your findings and append it to your final report document. Write 1-2 paragraphs on your observations.

Deliverables:

- Provide a README.md file including a detailed note (i.e., one paragraph) about the functionality of each of your Jupyter notebooks (baseline.ipynb, improved.ipynb, extra_credit_1.ipynb, extra_credit_2.ipynb; all found in /src/Code/), and complete instructions on how to run them. Make sure you include your name in each program and in the README.md file. Make sure all your programs run correctly.
- Provide an answer.pdf file where you include the results obtained along with your table(s) and the answers to the questions outlined above for the report.
- Submit all of your Python code as Jupyter Notebooks. 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.