CSCI544, Fall 2016: Assignment 3

Due Date: November 25th, before 4pm.

Introduction

The goal of this assignment is to get some experience with sequence labeling. Specifically, you will be assigning dialogue acts to sequences of utterances in conversations from a corpus. In sequence labeling it is often beneficial to optimize the tags assigned to the sequence as a whole rather than treating each tag decision separately. With this in mind, you will be using a machine learning technique, conditional random fields, which is designed for sequence labeling. You will be using the toolkit, CRFsuite.

The raw data for each utterance in the conversation consists of the speaker name, the tokens and their part of speech tags. Given a labeled set of data, you will first create a baseline set of features as specified below, and measure the accuracy of the CRFsuite model created using those features. You will then experiment with additional features in an attempt to improve performance. The best set of features you develop will be called the advanced feature set. You will measure the accuracy of the CRFsuite model created using those features. The last task is assigning dialogue act tags to a set of unlabeled data. You will do this with two models. The first model uses the baseline feature set and is trained on all the labeled data. The second model uses the advanced feature set and is trained on all the labeled data.

Data

The Switchboard (SWBD) corpus was collected from volunteers and consists of two person telephone conversations about predetermined topics such as child care. SWBD DAMSL refers to a set of dialogue act annotations made to this data. This (lengthy) annotation manual defines what these dialogue acts mean. In particular, see section 1c (The 42 Clustered SWBD-DAMSL Labels). Note, the statistics in this manual use a different training set than our experiments here but give you a rough idea of how frequently each dialogue act appears. We recommend that you skim the annotation manual to get an understanding of what the tags mean and help you think of good features.

If you have a question about the corpus and can't find an answer in the annotation manual, please post on Piazza or ask a TA or instructor. **DO NOT use resources from the web other than the annotation manual.**

We have divided the corpus into labeled and unlabeled (test) data sets. Download the zip file from Blackboard.

In all data, individual conversations are stored as individual CSV files. These CSV files have four columns and each row represents a single utterance in the conversation. The order of the utterances is the same order in which they were spoken. The columns are:

- act_tag - the dialogue act associated with this utterance. This is blank for the unlabeled data.
- speaker - the speaker of the utterance (A or B).
- pos - a whitespace-separated list where each item is a token, "/", and a part of speech tag (e.g., "What/WP are/VBP your/PRP$ favorite/JJ programs/NNS ?/."). When the utterance has no words (e.g., the transcriber is describing some kind of noise), the pos column may be blank, consist solely of "./.", have a pos but no token, or have an invented token such as MUMBLEx. You can view the text column to see the original transcription.
- text - The transcript of the utterance with some cleanup but mostly unprocessed and untokenized. This column may or may not be a useful source of features when the utterance solely consists of some kind of noise.

You will also notice a Python file **hw3_corpus_tool.py** included with the data. We have written this code to help you read the data. You are free to use this code or not. You are free to use this code in whatever way you want but if you use it then you must include it with your submission to create a standalone application. The file includes functions to read in these CSV files individually. Read the documentation in the file for more information or use Python's help() function.

CRFsuite

You will need to install pycrfsuite, a Python interface to CRFsuite. As discussed in the CRFsuite tutorial, and the pycrfsuite tutorial, this toolkit expects training data to be in the following format. A corpus is represented as two lists: a list of features (each element is a list of features for an example), and a list of labels. The features are binary. The presence of a feature indicates that it is true for this item. Absence indicates that the feature would be false. Here are the features for a training example using features for whether a particular token is present or not in an utterance.

```
['TOKEN_i', 'TOKEN_certainly', 'TOKEN_do', 'TOKEN_.']
```

What to do

Your main goal is to produce a set of dialogue act tags for the unlabeled data. You will use your labeled data to pick the best features for this task. You could simply split the labeled data by randomly putting roughly 25% of the data in the development set and using the rest to train your classifier. In this case, you would include entire conversations in either the training or development sets. In this assignment, it is up to you how you use your labeled data to evaluate different features. You could use a certain percentage of conversations for development, or you could use k-fold cross-validation.

You should try a set of features that we'll call baseline. In the baseline feature set, for each utterance you include:

- a feature for whether or not the speaker has changed in comparison with the previous utterance.
- a feature marking the first utterance of the dialogue.
- a feature for every token in the utterance (see the description of CRFsuite for an example).
- a feature for every part of speech tag in the utterance (e.g., POS_PRP POS_RB POS_VBP POS_.).

You'll need to create a Python program that reads in a directory of CSV files (INPUTDIR), train a CRFsuite model, tag the CSV files in (TESTDIR), and print the output labels to OUTPUTFILE

```
baseline_crf.py INPUTDIR TESTDIR OUTPUTFILE
```

You should try at least one other set of features that we'll call advanced. The advanced feature set should include more information than the baseline feature set. The idea is that you want to improve performance. As discussed in the grading section, part of your grade depends on developing a feature set better than the baseline. You'll need to create a Python program that reads in a directory of CSV files (INPUTDIR), train a CRFsuite model using the advanced features, tag the CSV files in (TESTDIR), and print the output labels to OUTPUTFILE

```
advanced_crf.py INPUTDIR TESTDIR OUTPUTFILE
```

Your scripts, baseline_crf.py and advanced_crf.py, will need to be able to handle CSV files with blank labels as well as labels for dialogue acts.

To evaluate your models on the development data, you'll need to write a Python program (evaluate_model.py) that compares your labels to the actual labels of the utterances in the CSV files and calculates accuracy which you will need for your report.

```
evaluate_model.py DEVDIR OUTPUTFILE
```

OUTPUTFILE should have the following format:

Filename="this is the filename without the path"
Label1
Label2
…

Filename="this is another filename without the path"
Label1
Label2
…

Please note that there is a blank line after the labels for a file are printed in the output file to separate the labels for files from one another.

What to turn in and Grading

You need to submit the following on Vocareum:

- **SWBD-DAMSL-Report.txt**.
    - Download and fill in the template from the class website. Include the results from your experiments. Keep the format text or you can use PDF.
    - Make sure that none of your code writes to a file called SWBD-DAMSL-Report.txt. We don't want to accidently damage the file when we test your code.
    - You can round off the values to 2 decimal points, but rounding is not mandatory.
- **All your code**: baseline_crf.py, advanced_crf.py, evaluate_model.py and any necessary support code including hw3_corpus_tool.py (if needed to run your code). This is required and the academic honesty policy (see rules below) applies to all your code. Do not submit data.

10% of your grade (10 points) is based on the report. Make sure to answer every question. 80% of the grade is based on performance on tagging unseen data using your baseline features. We will develop a specific grading rubric for assigning partial credit to cases where performance does not match the TAs' baseline model. 10% of the grade is based on performance on tagging unseen data using your advanced features. This 10% will be assigned by comparing your performance to that of your classmates and the model developed by the TAs.

Each time you submit your code, Vocareum will run your baseline_crf.py and advanced_crf.py scripts on a small set of data to ensure that they work. So submit early to check your code.

Late Policy

- On time (no penalty): before November 25th, 4pm.
- One day late (10% penalty): before November 26th, 4pm.
- Two days late (20% penalty): before November 27th, 4pm.
- Three days late (30% penalty): before November 28th, 4pm.
- Zero credit after November 28th, 4pm.

Other Rules

- DO NOT look for any kind of help on the web outside of the Python documentation.
- DO NOT use any external libraries other than the default Python libraries.

- Questions should go to Piazza first not email. This lets any of the TAs or instructors answer, and lets all students see the answer.
- When posting to Piazza, please check first that your question has not been answered in another thread.
- DO NOT wait until the last minute to work on the assignment. You may get stuck and last minute Piazza posts may not be answered in time.
- This is an individual assignment. DO NOT work in teams or collaborate with others. You must be the sole author of 100% of the code and writing that you turn in.
- DO NOT use code you find online or anywhere else.
- DO NOT turn in material you did not create.
- All cases of cheating or academic dishonesty will be dealt with according to University policy.

FAQ

1. **The code runs on my computer but not on Vocareum.**
    a. Vocareum has been tested for the same course in the previous terms and has been shown to work well. If your code doesn't run then there are some issues with your code, and you are responsible for correcting them. Test your code by submitting it on Vocareum incrementally and make sure it works.

2. **How will the TAs/Professors grade my HW?**
    a. We will be writing scripts which will run on Vocareum. Since we automate the process of grading, make sure that you write your code to match the output format “exactly”. If you do not do this there will be a penalty. You have been warned!

3. **I found a piece of code on the web. Is it ok to use it?**
    a. No! We run plagiarism checks on the code you write. The plagiarism detector is a smart algorithm which can tell if you’ve copied the code from elsewhere/others. If you’re caught copying the code, strict measures will be enforced. No exceptions!

    b. Please contact TAs/Professors during the office hours with your questions and we will make sure you understand the concepts and avoid unpleasant circumstances.

4. **Vocareum terminates my code before it finishes, but it finishes executing on my computer.**

    a. This usually means that your implementation is inefficient. Keep an eye out for places where you iterate. Run time issues can be solved by using efficient data structures (Hashmap, Hashset etc.).

5. **I assumed my code will run on Vocareum, but I wasn't proactive to test my code earlier. I couldn't meet the deadline because of this.**

    a. The late policy above still applies, and is automatically enforced by the system. Submit your code incrementally to make sure it functions on Vocareum. You have been warned!

6. **How do I log into Vocareum ?**

    a. You will receive an email with instructions. Sometimes, if you have been a part of a course where Vocareum was used previously, you will see the course listed once you log in. Please send the TAs an email if you have problems.

7. **When are the office hours ?**

    a. Please refer to the course website. http://nld.ict.usc.edu/cs544-fall2016/