[Intro to NLP, CMPSCI 585, Fall 2014]

Project milestone

This is part of the final project. Due Sunday, Nov 16. The requirements for the milestone are:

- Install and learn how to use CRFSuite. You may find the tutorial and manual helpful.
  - On Mac, we found these instructions worked (installing using Homebrew). Please report any issues to Piazza as soon as possible.
- Run the starter code we give you, which consists of an extremely simple feature extractor (`simple_fe.py`), on the train/dev split we've given you. Run the pipeline of training, prediction, and evaluation. Report the results from `tageval.py` .
- Kaggle steps (you can't do these right now, we are still ironing out the system)
  - For your group, create a Kaggle account. Make a fun team name! Email your Kaggle username to David, and make sure to list the names of the team members. Make sure you can access the class project.
  - Submit your predictions from the previous step to the Kaggle leaderboard and see the posted result there too.
- Turn in a short milestone report, described below, plus your feature extraction code. This consists of several analyses of the data and your tagger. Please submit electronically via Moodle.

How to run the starter code

The code and training/development data is here: starter_code.zip

Please see the CRFSuite documentation (they have both a manual and tutorial webpages) for details on how to use CRFSuite. You will need to have your commandline set up to access both the `crfsuite` and `python` commands. CRFSuite often outputs important information to standard output (a.k.a. "stdout"); to save this output to a file, you have to use shell redirection, which is the `>` operator in some of the commands below and in the CRFSuite tutorial. Read the Internet about this if you aren't familiar with it.

The steps you will have to do, to run the code, are

1. Install CRFSuite.
2. Look at the train and dev sets, which are just plain text files. Make sure you understand what the B/I/O notation means.
3. Extract features for both the train and dev sets. Read through `simple_fe.py` and you'll see it's set up to do that.
4. Train a model on the training set. You can use something like

`crfsuite learn -m mymodel train.feats`, which will create a new file `mymodel`. It should take less than a minute to train. (When I run it, it converges after 86 iterations of LBFGS.)

5. Make predictions on the devset, using the model you just trained. You can do something like `crfsuite tag -m mymodel dev.feats > predtags`, where the `>` is a stdout redirect to save the output to a file, `predtags`.

6. Evaluate the `predtags` against the gold standard tags of the devset. Our script `tageval.py` does this for you: if you used the filenames above, run `python tageval.py dev.txt predtags`.

At each step, you should look at the new file that was created to make sure you understand what's going on. The file `mymodel` is binary and you have to use `crfsuite dump` to look at it. But everything else is a plain text file, which you can open in your favorite text editor.

How to interpret tageval.py

Here's how to interpret the `tageval.py` output. When we run this with the features from `simple_fe.py` we get:

```
Span-level NER evaluation
F = 0.0362,  Prec = 0.7500 (12/16),  Rec = 0.0185 (12/647)
(1000 sentences, 19378 tokens, 647 gold spans, 16 predicted spans)
```

Precision is the number of correct name predictions, divided by the number of name predictions the tagger made. Precision answers: When your tagger makes a prediction, how frequently is it correct? Precision goes down if you make lots of false positives.

Recall is the number of correct name predictions, divided by the number of names in the gold standard. So it's the fraction of names that your tagger was able to find.

F-score is the average (specifically, the harmonic mean) of precision and recall. It's just an arbitrary way to turn them into a single number. (It's somewhat arbitrary, since for specific applications, you may care more or less about fp's versus fn's.) Our leaderboard is based on F-score.

Our evaluation works at the *span* level. You have to get the name's span (interval of token positions) exactly correct in order for it to count. So if the gold standard says "President Barack Obama" is a one big name, and your tagger predicts "Barack Obama" as a name, our system doesn't give any partial credit and that counts as a false positive. Note that if you run crfsuite with the `tag -t` or `learn -e` option, it reports something a little different, the tag accuracy at the token level.

Kaggle submission

Our Kaggle page is here.

To get started,

1. Create an account.
2. Set a fun display name. Make it pseudonymous please (the scores are posted publicly on the internet which is a little weird).
3. Note that Kaggle verifies the account through an SMS code. It turns out only one account is allowed per phone number. (I just tried to make a second account, and at the SMS verification step it did not let me.)
4. Email David the username and display name of your account.
5. When you join, you can choose to create/join a team. If you and a partner form a team, email that to David as well.

To submit a result:

- Get the script pred2kaggle.py (it's in the starter code if you downloaded it after 11/6) and convert the CRFSuitte tagging file into the format for uploading:

  python pred2kaggle.py predtags > for_kaggle

This file can be uploaded to Kaggle through their interface.

Questions and content for milestone report

Make sure to include the names of the two group members (or just yours if you are working by yourself). Do the following steps with either the `simple_fe.py` extractor, or with a better one if you are itching to improve it already. Explain what exactly you used.

1. Show the output of `tageval.py` that you get, in your report. Based on the precision and recall numbers, describe in English what's going on. At a high level, what types of errors does your tagger make?

2. Now try to figure out why. Manually look at the tag predictions, and compare them to the gold standard. An easy way to put them side-by-side is `paste predtags dev.txt` (paste manual); another way is to put them as columns in a spreadsheet. You might find it useful to search for the letter "B" (or "B" followed by a tab character), to find instances where your tagger predicted a name. Please explain: what types of things does your tagger get right, and what types of things does it get wrong?

3. Look at your model with `crfsuite dump`. Save it to a file and open it up in your favorite text editor. What sorts of things is it learning? Give examples of some highly weighted features. (If you like, you can sort them by weight with a short python script, or the unix command `sort -gk5` .) Does any of this help explain your tagger's errors?

4. Find one false positive and one false negative your tagger made. A false positive is where the tagger predicted a name, but there actually is no name in the gold standard. A false negative is where the gold standard has a name, but your tagger did not predict a name. Show them in your report, and for each, propose a feature that might fix it.

5. Find at least two examples of what you consider errors in the gold standard, in either the training set or dev set. Show them and explain your reasoning.