LINGUIST 429H – LAB 1

INTRODUCTION

In this assignment you'll be using string similarity and lexical analogy to model experimental data from Albright & Hayes (2003), who examined how people choose a past tense form for made-up verbs they've never heard before (a wug test). You'll model two types of behavioral data from this paper:

- acceptability of a wug form as a word of English in its own right (i.e. 'on a scale of 1-7 how much does *fleep* sound like a word of English?')
- a forced choice task where participants choose their preferred past tense for a given present tense form (i.e. "Today I *fleep*. I love *fleeping*. John ____ yesterday." A) *fleeped* B) *flept*)[1]

The Data

The homework assignment comes with the following files:

- Instructions.pdf (this document)
- PhoneticAlphabet.pdf
- min_edit_stub.py
- analogy_stub.py
- Train.csv
- Test.csv

Save these files to a directory on your computer and ***remove the '_stub' from both python script names***.

For your reference, the provided Train.csv and Test.csv files use the phonetic transcription alphabet 'DISC', summarized in the tables in the attached 'PhoneticAlphabet.pdf' file.

Test.csv provides experimental results (and phonetic transcriptions) from Albright & Hayes' experiments that we will be modeling:

- Orth is the orthographic representation of each wug (in present tense)
- Present is the phonetic transcription of the present tense of the wug
- Rating is participants' average acceptability rating on a scale of 1-7 for each wug
- Reg_Past is the phonetic form of the regular past tense
- Reg_Rating is participants' average rating for the regular past tense form
- Irreg_Past is the phonetic form of an irregular past tense form of the wug
- Irreg_Rating is participants' average rating for that irregular past tense form
- Class is a label indicating the morphological transformation required to form that irregular past tense

Train.csv is a dictionary of over 4000 actual English verbs that we will use for training our models, with the following columns:

- the orthographic form of the present
- the phonetic form of the present
- the orthographic form of the past
- the phonetic form of the past
- the morphological transformation class

Make sure you understand the information in these files before moving on!

[1] This differs from the task Albright and Hayes used in their experiment – they had participants rate each form of the past tense. But in this assignment we will only concern ourselves with which of the two forms participants rated higher, rather than the rating itself.

LINGUIST 429H – LAB 1

The Models

The analogy.py script implements a version of the analogical model Albright and Hayes' discuss in Section 2.4 of their paper. The model defines similarity of two words *i* and *j* in terms of their distance d_{ij}, which is the value we will compute using the *min_edit()* function:

$$\eta_{ij} = \mathrm{e}^{(-d_{ij}/s)^p}$$

For simplicity we will set *p*=1 for this assignment (this is the value Albright & Hayes found to work best in their simulations), but we will examine different values of *s*. Smaller values of *s* emphasize the closest neighbors more. The similarity of a word *w* to a collection of words V is simply the sum of the pairwise similarities between *w* and each word in V.

Examine the *get_neighbors()* and *similarity()* functions in the analogy.py script and make sure you understand how they implement this model.

We will use the lexical neighborhood model in two ways:

- **Acceptability**: Compute overall similarity of each wug to actual present tense forms in English. In Part 2 we will examine how well lexical neighborhood similarity corresponds to acceptability ratings.
- **Analogy**: For each wug, compute its overall similarity to verbs belonging to each morphological class (where morphological class is the transformation between the present and past form of the verb). The morphological transformation predicted by the model is the morphological class whose present tense forms have the highest overall similarity to the wug. In Part 3, we will examine how well morphological analogy performs on the forced choice task.

PART 1

In the first part of the assignment, you will complete the *min_edit()* function inside the min_edit.py script. Specifically, you will write code to fill in the min_edit table, which is the object called 'dist' in the provided code. Your code should start on line 24 of the provided stub.

You can test your *min_edit()* code by running min_edit.py at the command line with two strings as arguments. When called from the command line, min_edit.py will print out the table and the distance for the user-provided strings so you can check that your table is filled in correctly.

The subsequent parts of the assignment utilize this min_edit.py script by importing it into analogy.py.

PART 2

In this part you will use the provided *get_neighbors()* and *similarity()* functions in the analogy.py script to compute and store each wug form's overall similarity to actual English present tense forms. Both the wug and the actual verbs should be compared based on their phonetic forms! Completing this will require filling in the ?s on lines 98, 100, 102 with appropriate code and uncommenting the resulting lines.

Once you correctly fill in lines 102, 104, and 106, you'll be able to run your code as follows:

```
> python analogy.py Train.csv Test.csv
```

The script will take a few moments to run and, if your code is correct, your initial results should yield a correlation of 0.47193053672616114 when *s* is set to 2.0.

Questions

- Try different values of *s* and find one that works best. What is this value? Use this value for the rest of the assignment.[2]
- What's the best correlation you get?
- Uncomment line 108 to print out the similarities and ratings for each wug (comment it out again once you're done with this part). What are two words where the predictions of the model and the human ratings are particularly divergent? Why do you think the model may have trouble with these cases? (both of these scales are arbitrary, so the main thing that matters is the relative ordering. You may want to save your output to a file and open it in a spreadsheet program so you can sort by either value and look for mismatches).

PART 3

Now you will implement the morphological analogy model! For each wug, you will need to compute its overall similarity to verbs belonging to each morphological class. The predicted class according to the model is the one with the highest overall similarity, and that's the class you should append onto the *preds* list. This code should go right after the code from Part 2 in the same loop over wugs. (hint: you will need to compute similarity in a new way for this part, separately for each morphological class. Depending on how you organize your code, you may not need to use the provided functions.)

Once you implement this part, you can uncomment line 120 to evaluate the accuracy of the model's predictions. Then, write some additional code to print out information out that will help you analyze the behavior of the model. Specifically, for each wug, print out:

- the wug
- the participants' preferred past tense response
- the model's predicted past tense response
- whether they match or not
- the close neighbors (d <=2) of the wug

Now you can examine which wugs it gets right and which it gets wrong and what those wugs' neighborhoods look like. The results are driven mostly by neighbors within a distance of 2 so this gives you a good sense of what words the model is using for analogical purposes.

This is the final version of **analogy.py** that you should submit. It should print out the info for each wug it gets wrong, and it should print out the correlation and accuracy at the end.

Questions

- What's the accuracy of the model on the forced choice task?
- Which words does the model get wrong (list them all)? For each one, indicate what the model predicts and what it should have predicted instead.
- Examining the close neighbors of five of the wugs it gets wrong, which words seem to lead it astray? In other words, are there particular words the model 'thinks' are close neighbors, but it really shouldn't count them as close for purposes of applying the past tense transformation?
- Do you notice any general problems with the model? That is, is it failing to capture something general about how you think humans actually seem to perform this task?[3]

[2] The best value of *s* makes it so that neighbors farther than 3 away matter very little to the predictions. You can confirm this by running *get_neighbors()* on line 97 with n=3, which will only look at neighbors within a distance of 3, and you should get very similar results. If you try this, make sure to change it back after you're done.

[3] Hint: One possible response to this question and the second option in Part 4 relates to what Albright & Hayes call 'variegated similarity' vs. 'structured similarity'.

LINGUIST 429H – LAB 1

PART 4

Try to improve the accuracy of the model by doing one (or more) of the following things:

- Modify the min_edit cost functions (*ins_cost()*, *del_cost()*, and/or *sub_cost()*) to penalize some edits more than others. At the top of the min_edit.py script is a regexp that matches vowel symbols, in case that's useful.
- Modify the *min_edit()* function itself to penalize changes in some positions more than changes in other positions. You may have noticed in analyzing the errors above that similarity in some parts of the word is more important than in other parts of the word.
- Based on your analysis of errors in Part 3, make some other modification to distance or similarity calculations that tries to correct some of these errors.

Make sure your modifications are general rather than looking at the errors and making specific changes based on the errors! Interesting solutions and solutions that make big difference in accuracy have chance at earning extra credit!

Whatever changes you make for your code in this part of the assignment, save them as new versions of the scripts with "_part4" added to the script name (i.e. **min_edit_part4.py** and/or **analogy_part4.py**). Make sure you comment your new code to explain what it's doing and label it in the comments as "PART 4".

Questions

- Were you able to improve the model's accuracy? What accuracy were you able to achieve?
- What did you try? Why did you think it might help?
- Why do you think it did (or didn't, as appropriate) work?

EXTRA CREDIT

If you're still having fun…

- Create new test data to try to stump the model! Come up with new wug forms, collect judgements from your friends, transcribe the results in the same format as Test.csv, and test the model on this new data. Explain how you chose your wugs, how you got ratings, and discuss the results.
- This analogical model was kind of cheating: it was given the morphological rules. How much harder would analogy be if the analogical model had to learn the rules too? Discuss (or implement) min_edit alignment of the present and past tense forms in the training data to extract 'rules' from (present, past) pairs. Discuss how you would use these rules to generate past tenses for novel forms like 'bize' and 'nold'. What challenges arise?

WHAT TO SUBMIT

- Submit your completed '**min_edit.py**' from Part 1
- Submit your final '**analogy.py**' from Part 3
- Submit the code you modified for Part 4 with new names as indicated above
- Submit your answers to questions (in PDF or TXT format).