HOMEWORK 4

OVERVIEW

In this lab we will use a MaxEnt grammar and stochastic gradient descent to model the gradual learning of both regular and irregular past tense verbs in English. Our model's task will be to learn the past tenses of actual verbs (e.g. know/knew, go/went, think/thought, look/looked…) while at the same time making predictions about general preferences for different classes, and how those preferences change over the course of learning.

We will work with 6 past tense classes in this lab: REG, NULL, X->{, X->$t, 5->u, g5->wEnt. These six classes account for 4107 of the 4253 verb ***types*** in the full lexicon examined by Albright & Hayes (2003) – so nearly all of them. I have extended the dataset by adding ***token*** frequencies for each verb from CELEX, which are based on large text databases like newspapers and books. Have a look at the training data to see each verb, its class, and the token frequencies. Each verb has three associated frequencies: PRESF (tokens of the present form), PASTF (tokens of the past tense form), LEMMAF (total tokens of all forms of the verb). Type frequency is not explicitly listed since type frequency just counts each verb as occurring once. In this assignment we will only use type frequency and the PASTF token frequencies, but the others are there for your reference. If you open the training data in a spreadsheet, you will be able to filter by class so you can easily inspect which verbs are in each class.

Our MaxEnt grammar will have 6 general (grammatical) constraints, one for each class. The grammatical constraints and their weights are shared across all verb types. In the example tableaux below, you will see that both use/used and think/thought have all the same constraints and weights for the first six constraints. Each constraint picks out exactly one class as the preferred class, assigning violations to all other classes. So the REG constraint is only satisfied by the REG past tense candidate, the NULL constraint is only satisfied by the NULL past tense candidate, and so on. Each verb belongs to a specific class so only one of the candidates is actually the correct outcome for each verb. For example, REG is the correct candidate for use/used and X->$t is the correct candidate for think/thought.

```
TABLEAU: juz->juzd
                   REG      NULL      X->{      X->$t      5->u  g5->wEnt     SCALE
                  3.51      0.55      0.01      0.00      0.00      0.00      0.76
         REG         0         1         1         1         1         1         0
        NULL         1         0         1         1         1         1         1
        X->{         1         1         0         1         1         1         1
       X->$t         1         1         1         0         1         1         1
        5->u         1         1         1         1         0         1         1
    g5->wEnt         1         1         1         1         1         0         1

TABLEAU: TINk->T$t
                   REG      NULL      X->{      X->$t      5->u  g5->wEnt     SCALE
                  3.51      0.55      0.01      0.00      0.00      0.00      5.27
         REG         0         1         1         1         1         1         1
        NULL         1         0         1         1         1         1         1
        X->{         1         1         0         1         1         1         1
       X->$t         1         1         1         0         1         1         0
        5->u         1         1         1         1         0         1         1
    g5->wEnt         1         1         1         1         1         0         1
```

Since the grammatical constraints and weights are shared across all verbs, they can only encode general grammatical preferences, not verb-specific preferences. To encode verb-specific preferences, we give each verb its own lexically-specific constraint, which is called a *scale*. The scale is simply a copy of the constraint for the correct class of that verb. So for use/used the scale copies the REG constraint, while for think/thought, the scale copies the X->$t constraint. Since each verb has its own scale, each verb can independently update the strength of its scale to prefer the correct candidate. The scales are treated like regular constraints for the purposes of calculating probabilities: you can also think of them as adding weight to the constraint for the correct class. In the examples above, the grammatical constraints strongly prefer the REG class, and the use/used scale strengthens that preference a little bit by adding 0.76 to REG. For think/thought, the scale is much stronger and adds weight to the X->$t constraint. In this way, if the scale is strong enough, it can override the preferences of the general/grammatical constraints.

In the provided code we will assume that scales and the general grammatical weights are treated identically by the learning update, with the same L2 prior settings, and the same learning rate. In the Extra Credit portion, this is something you can play around with. In Part 1 you will complete the code to make updates to the general constraints, and in Part 2 you will complete the code to make updates to the scales.

The code takes three command-line arguments: the training file, the number of iterations, and the learning rate:

-
  ```
  Python MaxEnt_Lex.py TrainingData.txt 500000 .01
  ```

PART 1

We will begin by working with a MaxEnt grammar that only has general constraints, one for each of the six morphological classes. This variant of the model will not be able to learn how past tenses are formed for actual verbs, but it will learn something about general preferences for the various morphological classes. This model is known to produce "Frequency Matching" behavior: its nonce predictions match the proportions it observes in the training data.

Your first task is to understand the provided code. The second task is to complete the *calcEO()* function by filling in the code on lines 50-55. The *calcEO()* function computes and returns the expected and observed violations for each constraint.

Once you complete the function, run the model with a learning rate of 0.001 for 50,000 iterations. If you have written the function correctly, your model should produce results very close to the following. It should reach logprobability of about -540 and a nonce prediction rate of about 94% for the regular and stay there.

```
STARTING ITERATION: 49900
New Weights
           REG - 4.34
          NULL - 0.04
          X->{ - 0.00
         X->$t - 0.00
          5->u - 0.00
      g5->wEnt - 0.00
New nonce predictions
           REG - 0.94
          NULL - 0.01
          X->{ - 0.01
         X->$t - 0.01
          5->u - 0.01
      g5->wEnt - 0.01
logprobability of the data:  -539.5225751118228
```

HOMEWORK 4

Questions
The provided code is set up to use a uniform frequency distribution (*unifreqs*) which assumes each verb type has the same chance of being observed (line 147). This is not what actually happens during language learning since verbs are observed with different rates. Modify the code to use the *pastfreqs* distribution instead (this is an estimate of how often each past tense is observed) and rerun the code to answer the following questions.

Q1
What nonce prediction rate for the regulars does the model predict after 50,000 iterations if observations are instead sampled from the past tense frequency distribution?

Q2
Albright and Hayes (2003) found that adults produced the regular past tense 81.5% of the time in their wug test. Which distribution (uniform or past) better matches these results? Why might this be? (hint: consider the proportion of examples that are regular in each distribution)

PART 2

Now we will extend the model so it can also learn the past tense of each verb in the training data. For each verb, the model will create a copy (or clone) of the constraint corresponding to the observed (correct) past tense. This cloned constraint is called a *scale* and is specific to each verb.

The provided code already defines the scales (line 125-131), sets their values to 0.0 for all verbs (line 127), and specifies the L2 prior settings for the scales (line 136-138). It also computes the expected and observed violations for the scale of each learning example (line 164-165).

To complete extending the model to use scales, your task is to implement the update rule for the scales. This should just be two lines of code and should work analogously to the update for the weights on lines 169-170. Your scale update will need to reference the current value of the scale plus all of the following variables: rate, scaleO, scaleE, scale_mu, and scale_sigma2.

If you write the update correctly and run the model just as above (e.g. rate=0.001 using the past distribution), the scale for ‘think/thought’ should reach about 1.11 and the scale for ‘go/went’ should reach about 0.77 by 50,000 iterations.

Questions
Now that the scales are working, we can examine how individual words are learned. Switch the learning rate to 0.01. This will make the model a little bit more noisy, but it will run more quickly, so you can evaluate the results more quickly, but keep in mind that the model will do slightly different things each time it is run so if some numbers are very close you may want to run the model a few times to make sure the behavior you’re seeing is consistent. If you run the model a few times with learning rate 0.01 for 500,000 iterations (with the past tense distribution), you will see that different verbs are learned at different rates. To better understand what the model is doing for each verb, you can use the provided *print_tab()* function (line 156) to examine what the constraint weights and scales look like for different verbs.

Q1
Regular verbs reach high accuracy very quickly, even infrequent verbs like ‘crimp/crimped’ are already 80% accurate by about 2000 iterations. Explain why ‘crimp/crimped’ is so accurate so quickly. Specifically, discuss the roles of the general constraints and scale for achieving correct predictions on this verb.

Q2
For irregular verbs, the scales will need to reach a strength of about 5 for these verbs to be produced at around 80% accuracy. Examine how accuracy changes for the most frequent verb TINk->T$t over time. At (roughly) what iteration does the model predict 80% accuracy for TINk->T$t? Explain why this irregular verb takes so much longer to reach 80% accuracy than 'crimp/crimped'. And also, discuss the roles of the general constraints and scale for achieving correct predictions on this verb.

Q3
Although 'crimp/crimped' gets to about 80% accuracy very quickly, it then takes a long time for it to improve beyond this and get to 90% accuracy and above. Which verb, 'crimp/crimped' or 'think/thought', gets to above 90% first? Why is this verb mastered earlier?

PART 3

Now that you understand the basic mechanics of the model, in this part of the assignment, we will examine whether the model captures some observations about children's acquisition of the English past tense.

Questions
Each of the questions below asks you to analyze the model to determine whether it captures a different observation about acquisition.

Type Frequency
All else being equal, children tend to acquire verbs belonging to larger classes (classes with more types) earlier than verbs belonging to smaller classes. To test the model's predictions for this, we will consider verbs from three differently-sized classes: go/went (with 3 verbs in its class), set/set (with 35 verb types in the NULL class), and use/used (with 4041 verb types in the REG class). To make sure the token frequency of these verbs isn't affecting the predictions, we will artificially set the frequencies of these three verbs to the same value: 3099 (the frequency of 'go/went'). To do this, after setting the frequency dictionary to the past tense frequencies, set the frequency of use/used and set/set to 3099. Then run the model several times to examine how quickly it learns each of the three verbs and answer the following questions. Undo these modifications once you're done with this part.

Q1
At (roughly) what iteration does the model reach 80% accuracy for each of the three verbs: go/went, set/set, and use/used? Explain whether this order corresponds to these classes' type frequency.

Q2
Notice that the acquisition order is not directly related to the strength of the scales for each of these words (e.g. higher scales do not mean higher accuracy). Why does this happen? Specifically, what aspect of the model determines the acquisition order of these three verbs? And, how is this aspect of the model sensitive to type frequency?

Token Frequency
All else being equal, children tend to acquire high frequency verbs earlier than lower frequency verbs. To see if the model captures this behavior, you will compare high frequency verbs in one class to mid-frequency verbs in the same class. Change the example words to add 'know/knew' (2038), 'grow/grew' (316), and 'buy/bought' (316), in addition to the already included 'think/thought' (4426). Then run the model a few times to answer the following questions.

HOMEWORK 4

Q3
Explain whether the model correctly predicts that ‘think/thought’ is acquired before ‘buy/bought’.

Q4
Explain whether the model correctly predicts that ‘know/knew’ is acquired before ‘grow/grew’.

Q5
Why does the model make these predictions?

EXTRA CREDIT
One empirical observation that has so far been elusive to models is the so-called U-shape learning curve that we discussed in class. While the empirical evidence for the U-shape curve is noisy and subtle, the main observations are the following:

- Children initially don’t produce any past tense marking at all
- They then start (sometimes) producing correct past tense marking for a few high frequency verbs, most of which are irregular
- They then get pretty good at using the regular past tense for many more verbs and at about the same time they start ‘over-regularizing’: occasionally using the regular past for irregular verbs (the rate of over-regularization is pretty low, perhaps around 5-10%)
- While not technically part of the ‘U-shape curve’, a related observation about the acquisition of the English past tense is that children aged 4-7 produce more regular forms in wug tests than adults do. Children very rarely produce wug forms like ‘spling/splang’ while adults do it about 20% of the time.

The model in its current form is not equipped to capture these facts. For extra credit, you can try some minor modifications to see if any of them (or some combination of them) could allow the model to capture some aspect of this learning trajectory.

a) The current model has no way to favor lack of tense marking. As an approximation, however, we could assume the NULL class actually represents no tense marking at all and set the model up to initially prefer NULL marking. You can do this by setting the initial weight of the NULL constraint higher than the others.
b) The current model takes a very long time to learn each word. Kids are almost certainly faster than the model at learning actual words. We might be able to improve the model by making the learning rate for scale updates bigger than the learning rate for weight updates.
c) Other similar options to try would be different initial values for scales vs. weights. Or different values for the L2 priors than the model currently assumes (which is the same for every weight and scale).
d) Whatever you try, it should be a general idea about how to make the model more realistic, without assuming the model knows anything ahead of time about which pattern is regular. It should not be a modification specifically designed to create the U-shape curve (e.g. making scales initially higher for a few high frequency irregulars).

Explain what predictions the original model makes and how they don’t align with the U-shape learning observations. Then explain what modifications you made, what you hoped they would do, and discuss whether this improved the model’s predictions.

WHAT TO TURN IN
Turn in your final **MaxEntLex.py** program from Part 2. Submit your answers to all the questions in a separate file in PDF format.