*Kelvin Han*

*Esteban Marquer*

*Elena Khasanova*

**Character-based Language Models : Assignment 5**

**Report**

***1. What the F?***

*The reason why the output on smaller. Do you notice anything? They all start with F! In fact, after we hit a certain order, the first word is always First? Why is that? Is the model trying to be clever? First, generate the word First. Explain what is going on in your writeup.*

The reason why the system outputs the first word starting with F in generate\_text is the following line:

history = "~" \* order

Without any smoothing, the probability for character F following the padding characters will always be 1 while the probability of other characters will equal to 0 , so the system will consistently output it as the only possible character in the set. Similarly, for the bigger order (compare 5 and 2), the more similar the first characters would be to the word «First».

***2. Perplexity***

*Discuss the perplexity for text that is similar and different from Shakespeare’s plays. We provide you*[*two dev text files*](http://computational-linguistics-class.org/downloads/hw5/test_data.zip)*, a New York Times article and several of Shakespeare’s sonnets, but feel free to experiment with your own text.*

*Note: you may want to create a smoothed language model before calculating perplexity, otherwise you will get a perplexity of 0.*

The perplexity for the text similar to Shakespeare's play is quite high: 0.99, which means that the model seems to be quite good. However, for the text different from Shakespeare's the system would throw a default value of infinity as there are many characters that are not in the vocabulary. We adjusted the treatment of unseen history in the previous steps, however, unseen characters have not been handled yet.

**3. Lambdas and Interpolation**

Experiment with a couple different lambdas and values of k, and discuss their effects.

In our system, the lambdas are assigned randomly. However, if we experimented with different lambdas for the same character and history for order=4 (so with 4 language models) and got the following results:

The order of language models and hence lambdas is decreasing (from order=order to order=1)

#### lambdas = [0.4, 0.3, 0.2, 0.1]

>>calculate\_prob\_with\_backoff("e", "henc", lms, lambdas)

0.7446717617281534

lambdas = [0.1, 0.2, 0.3, 0.4]

>>calculate\_prob\_with\_backoff("e", "henc", lms, lambdas)

0.5385816663405953

There is a difference, but not a big one. However, the bigger weight on bigger order size results in bigger probability for a reasonable continuation of history.

If we try to give significantly more weight to the quadrigram and much less the other:

lambdas = [0.9, 0.03, 0.05, 0.02]

print(calculate\_prob\_with\_backoff("e", "~~Th", lms, lambdas))

0.043677161751167815

lambdas = [0.03, 0.9, 0.05, 0.02]

print(calculate\_prob\_with\_backoff("e", "~~Th", lms, lambdas))

0.043677161751167815

lambdas = [0.03, 0.02, 0.9, 0.05]

print(calculate\_prob\_with\_backoff("e", "~~Th", lms, lambdas))

0.43075824131821266

lambdas = [0.03, 0.02, 0.05, 0.9]

print(calculate\_prob\_with\_backoff("e", "~~Th", lms, lambdas))

0.34296422555268713

As we see, the value for the computation with quadrigram and trigram is the same as the system has never seen this context (the training corpus does not start with «~~Th», however unigram and bigram give quite similar results as both «he» and «the» are rather frequent words.

Let's try another call:

lambdas = [0.9, 0.03, 0.05, 0.02]

print(calculate\_prob\_with\_backoff("e", "henc", lms, lambdas))

0.856362490264393

lambdas = [0.03, 0.9, 0.05, 0.02]

print(calculate\_prob\_with\_backoff("e", "henc", lms, lambdas))

0.7495426063952526

lambdas = [0.03, 0.02, 0.9, 0.05]

print(calculate\_prob\_with\_backoff("e", "henc", lms, lambdas))

0.6974381818909046

lambdas = [0.03, 0.02, 0.05, 0.9]

print(calculate\_prob\_with\_backoff("e", "henc", lms, lambdas))

0.26534403955471586

So, the bigger order we put weight on, in other words, the more it looks like a word, the bigger probability for it to end «correctly» (to be followed by the selected character) we have.

#### 4. Classification

Describe the parameters of your final leaderboard model and any experimentation you did before settling on it.

The sole determining parameter of the final leaderboard model is the size of the window. We produced results with window sizes of between 2 and 4 (1 would not have been informative for any prediction). The best results were obtained from a model of window size 4, this was determined by a visual comparison of the results based on our understanding of what the cities in each of the countries are typically like.