CIS 530 Assignment 5: Character-based Language Models

Ignacio Arranz
Yezheng Li

1 Part I

Problem. *Discuss the perplexity for text that is similar and different from Shakespeares plays. We provide you two dev text files, a New York Times article and several of Shakespeares sonnets, but feel free to experiment with your own text.*

Answer. *They all start with F because F is the first letter of the text and that by default, function* ***train_char_lm(...)*** *does not read line by line (we adjust this later on for leaderboard).*

Problem. *What do you think? Is it as good as 1000 monkeys working at 1000 typewriters?*

Answer. *The interesting thing with this method is that it generates words that do not exist, given that it is based on the next most likely character.*

2 Part II

Problem. *Discuss the perplexity for text that is similar and different from Shakespeares plays. We provide you two dev text files, a New York Times article and several of Shakespeares sonnets, but feel free to experiment with your own text.*

Answer. *The logarithm of perplexities obtained with models trained on Shakespeare were 3.95 for the New York Times, and 2.66 for Shakespeare sonnets. This is reasonable, as a lower perplexity represents a more similar text. However, these values on their own don't help us understand if the difference between them should be smaller or larger. That's why, for reference, we also ran perplexity of a novel in Spanish by Cuban author Jose Marti. For this text, we obtained a perplexity of 8.77. This helps put into perspective that the difference in perplexity between New York Times and Shakespeare sonnets is reasonable, when comparing with the perplexity of a book in another language.*

scenario	λ_4 (4 chars)	λ_3 (3 chars)	λ_2 (2 chars)	$'k'$	$'e'$	$'\ '$
1	1/3	1/3	1/3	0.33	0.11	0.11
2	1/6	2/6	3/6	0.23	0.10	0.15
3	1/14	4/14	9/14	0.16	0.09	0.18
4	9/14	4/14	1/14	0.51	0.12	0.04

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

Answer. *We decided to calculate the probability with backoff for specific words, to try and understand what the influence of different values of λ was. In this case, we used 'Thin' as hist and 'k', 'e' and ' ' as the chars. It is interesting to see that in scenario 4, where there is a high weighting placed to the 4-gram 'Thin', then k is a very likely character to appear afterwords, most usually forming the word 'Think'. However, in scenario 3, where there is high weighting placed on the 2-gram 'in', then it is much more likely that a space comes afterwards, because the word 'in' is a very frequent one.*

- *We set* $\lambda = \frac{1}{len(lms)}\mathbf{1}_{len(lms)}$.
- *values of k: Laplacian smoothing, that is,* $k = 1$ *performs the best (tried* $k = 0, 1, 2, 3$*). This might have something to deal with the way we tackle missing "char" – we collect the entire vocabulary of training set.*

3 Part III

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

Answer. *Just tri-gram model (order=2), with various attempts to leading to improvements*

- *missing "history" issue – assign the missing history with lm[history][char] the probability* $\dfrac{16.0}{len(lm)}$;
- *missing "char" issue –*
  - *When training data, try to avoid missing "char" by assigning 0 to the* ***lm[history][char]*** *where we collected the entire vocabulary from the training set (for each country);*
  - *assign the missing history with* ***lm[history][char]*** *the probability* $\dfrac{16.0}{len(lm)}$;
- *training*

  - *on both 'train/' and 'val/' (this turns to have limited effect for leader-board; not surprising that it (drastically) improves 'val/' set performance);*
  - ***train_char_lm(. . .)*** *reads the file line by line.*

- *Just set* ***add-k*** *as 1. (See experiments of* ***add-k*** *in answer to the previous question)*