Assignment 2

CSE 447 and 517: Natural Language Processing - University of Washington

Winter 2022

Please consult the course website for current information on the due date, the late policy, and any data you need to download for this assignment. This assignment is designed to advance your understanding of frequency information in text data and some mathematical properties of language models.

Data: The data you need for this assignment is available at `https://nasmith.github.io/NLP-winter22/assets/data/A2.tgz`.

Submit: You will submit your writeup (a pdf) and your output for problem 1 (do not include code) via Gradescope. Instructions can be found here. Note that you will make two submissions: one for the pdf, one for the output.

1 Substitution Cipher Breaking

This problem deals with substitution ciphers. A character-level substitution cipher corresponds to a map *encrypt* : $\Sigma \rightarrow \Sigma$, where Σ is the set of characters you use to write your texts. [1] Here is an example:

characters:	`abcdefghijklmnopqrstuvwxyz.` (space)
encrypt$(\cdot)$:	`hiwxyzpqjklmnovabcdefgrstu.` (space)

Note that the period and space symbols map to themselves. To encrypt a string in Σ^*, we encrypt each symbol in turn and concatenate. So, using this substitution cipher, we would map "`the zoo seems to be closed today.`" to "`eqy uvv dyynd ev iy wmvdyx evxht.`"

In the following questions, let $N = |\Sigma|$.

1. How many substitution ciphers are there?

2. In a given cipher, a symbol $x \in \Sigma$ is said to be *fixed* if *encrypt*$(x) = x$. (In our example above, the period and space symbols are fixed.) How many substitution ciphers are there with no fixed symbols? (Hint: the combinatorial concept of a *derangement* is useful to use here; we suggest you look it up. Make sure to show all steps in your work.)

3. Suppose you select a substitution cipher with *no fixed symbols*, uniformly at random, and your friend decides to try to break the cipher by selecting ciphers at random. What is the probability that your friend guesses your cipher *exactly* within k tries (a) if they assume nothing, and (b) if they (correctly) assume your cipher has no fixed symbols?[2]

[1] For questions 1–4, assume the mapping is one-to-one.

[2] This problem was inspired by the Enigma machine.

4. A well-known problem with substitution ciphers is that, in a long ciphertext, the frequency of *encrypt*(x) will be close to the frequency of x in plaintext. In natural languages, there is a lot of variance in different symbols' frequencies. (For words, "Zipf's Law" states that the probability of the rth most frequent word in a text corpus will have relative frequency proportional to $\frac{1}{r}$.) Your task is to exploit this problem to decrypt the ciphertext we provide to you. The original plaintext is in all-upper-cased English. You may assume the space symbol, numerical digits, and any other non-alphabetic symbols are fixed. This implies that $N = 26$. We suggest that you automate the calculation and visualization of the symbol frequencies. Some manual search may be required since your ciphertext (and any plaintext you use to estimate English relative frequencies) will be finite, leading to variance in your estimates. You should submit your decrypted text.

5. One way to make a cipher more robust to the analysis discussed above is to augment the output alphabet with more symbols: *encrypt* : $\Sigma \rightarrow \Sigma \cup \Gamma$. Then, during encryption, instances of more frequent input symbols can be randomly assigned to multiple output symbols. For example, suppose that e is the most frequent symbol in a plaintext corpus, accounting for 5% of tokens. We would then allocate $0.05|\Sigma \cup \Gamma|$ symbols to e, any of which might be chosen uniformly at random when encoding e. If we do this for every symbol in Σ, then the frequencies of all symbols in $\Sigma \cup \Gamma$ will be roughly equal in a ciphertext. If we require that the encoding function be *deterministic*, how would you design this new cipher to prevent frequency analysis attacks?

2 Valid Probabilities (n-Gram) – Based on Eisenstein 6.1 (p. 135)

Prove that n-gram language models give valid probabilities if the n-gram probabilities are valid. Specifically, assume that

$$\sum_{v \in \mathcal{V}} p(X_t = v \mid X_{t-n+1} = x_{t-n+1}, \ldots, X_{t-2} = x_{t-2}, X_{t-1} = x_{t-1}) = 1 \tag{1}$$

for all contexts $\langle x_{t-n+1}, \ldots, x_{t-2}, x_{t-1} \rangle$. For $m \geq 1$, let $\mathcal{V}^m \subset (\mathcal{V} \setminus \{\bigcirc\})^*$ denote the set of sequences of length m that include no stop symbols. Prove that $\sum_{\boldsymbol{x} \in \mathcal{V}^m} p_{\textit{model}}(\boldsymbol{x}) = 1$, where $m \geq 1$ and $p_{\textit{model}}(\boldsymbol{x})$ is the probability of $\boldsymbol{x}$ under the n-gram model. Your proof should proceed by induction. You should handle the start-of-string case where the context is $n-1$ start symbols, but do not include the stop symbol.

Extra Credit: If we do not include the stop symbol, n-gram models define a valid probability distribution over sequences of a fixed length, as shown above. As a consequence, they do not define a valid probability distribution over $\mathcal{V}^*$. Show that by including the stop symbol, n-gram models define a valid distribution over sequences of all lengths. That is, show $\sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} p_{\textit{model}}(\boldsymbol{x}) = 1$.

3 Valid Probabilities (RNN) – Based on Eisenstein 6.2 (p. 135)

First, show that RNN language models are valid using a similar proof technique to the one in problem 2.

Next, let $p_r(\boldsymbol{x})$ indicate the probability of $\boldsymbol{x} \in \mathcal{V}^m$ under RNN r. An ensemble of R RNN language models computes the probability:

$$p(\boldsymbol{x}) = \frac{1}{R} \sum_{r=1}^{R} p_r(\boldsymbol{x}) \tag{2}$$

Does an ensemble of RNN language models compute a valid probability? Prove your answer.