Project1 Part1 Report CS4740 Team members: Ransen Niu, Weicheng Yu

Problem Statement

Part 1 of this project uses unsmoothed unigram and bigram models to generate random sentences.

Preprocessing

We consider email headers as irrelevant text and removed them. We noticed that there are some patterns. First is that anything before “Subject : ” is part of the email header, so we removed anything before and including “Subject :”. Then if the email is a replying email, there would be a “writes :” somewhere after and anything before “writes :” is part of the header as well. So if there is a “writes :” in the “Subject :”-trimmed text, we remove anything before and including “writes :”.

We removed the “bad” punctuations, which are usually irrelevant in generating sentences, including “#$%&*+-/=@[]ˆ_‘{—}\<>∼”. Then we tokenized the text into an ordered list of words.

We consider all words are capital-sensitive. Also, we have a set of “stop” punctuations (i.e. “.!?”) which indicate the end of the sentence.

Unsmoothed Unigram and Sentence Generation

We build the unigram model by taking counts of each word's frequency divided by the total number of the word tokens. To generate a sentence, we randomly sample words according to the unigram models. For example, a word w that has a unigram model P(w) = 0.1 would have 10 percent chance to be picked. When one of the “stop” punctuations appear, the sentence ends.

Sentence Examples

Sentence Analysis

Unsmoothed Bigram and Sentence Generation

We built two bigram models with different specifications to compare which one is better.

For the first model, we go through the element (word) in the wordlist two by two to count the frequency of each pair of them. When generating a sentence, we randomly choose the beginning word according to the unigram model. Afterwards, we use the bigram model to randomly sample which word is going after the previous word. And the sentence ends when one of the “stop” punctuations is met.

For the second model, there is some preprocessing. We prepend a beginning marker “<” at the beginning of the wordlist and we add the beginning marker “<” after each “stop” punctuation. Then we know once we encounter a “<”, it means the end of the previous sentence and the beginning of a new sentence. Then we generate this second bigram model the same way as the first bigram model, but we should notice that here the beginning marker “$<$” is counted as a new word type. When generating a sentence, we always make “” as the beginning of the sentence, and then use this second bigram model to randomly sample which word is going after the previous word. And the sentence ends when one of the “stop” punctuations is met. Finally, we remove the beginning marker “$<$” from the generated sentence because the marker is not really a part of a sentence.

Sentence Examples

Sentence Analysis

Comparisons between Unigram and Bigram models

Contribution

Ransen Niu: preprocessing, second bigram model, report.

Weicheng Yu: preprocessing, unigram model, first bigram model.