Assignment 2: Markov and Hidden Markov Models for Text

Introduction

In this assignment, you will develop Markov and hidden Markov models for English text. You may use any programming language you wish to complete this assignment.

There are 4 data files: `vocab.txt`, `unigram_counts.txt`, `bigram_counts.txt`, and `trigram_counts.txt`. The format for these files is as follows. vocab.txt contains the vocabulary, one word per line:

```
1       word1
2       word2
...
n       wordn
```

There are two special words in the vocabulary. The words `<s>` and `</s>` denote the beginning and end of a sentence respectively. The files `*gram_counts.txt` contain a 3-gram probability model, with one conditional probability table entry per line. For `unigram_counts.txt`, each line of the file contains:

$i \quad \log_{10} P(x_t = i)$

For `bigram_counts.txt`, each line of the file contains:

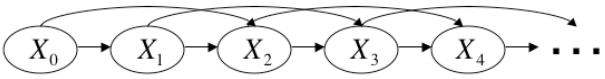
$i \quad j \quad \log_{10} P(x_t = j | x_{t-1} = i)$

For `trigram_counts.txt`, each line of the file contains:

$i \quad j \quad k \quad \log_{10} P(x_t = k | x_{t-1} = j, x_{t-2} = i)$

Assume that any conditional probability table entry not given in the data file is 0. The probability distributions may not sum to 1, due to missing (small) values.

Part 1: Generating Text Using a Markov Model



The first task in this assignment is to generate sentences using the trigram model depicted above. In this model, each X_t is a discrete random variable denoting the t^{th} word in a sentence, taking values from the vocabulary defined above.

The joint probability distribution

This is a Markov model, a special case of a Bayesian Network. The probability distribution $P(X_t|X_{t-1}, X_{t-2})$ is assumed stationary (same for all t), having the values given in the trigram data file specified above.

Use the bigram distribution for $P(X_1|X_0)$.

For $P(X_0)$, note that all sentences start with ¡s¿. $P(X_0 = <s>) = 1$.

Sampling from the network

Generate sentences using this model by implementing the **prior-sample** routine on this network. The probability that prior-sample generates a particular event is given by the formula:

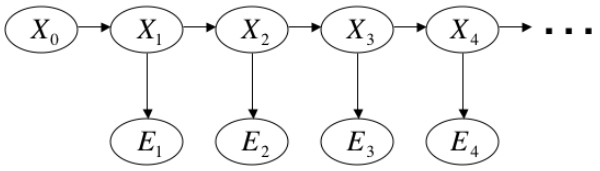
$$S_{PS}(x_1 \cdots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

where probabilities for $n = 1$ are given by the entries in the bigram data file and those for $n = 2$ are given by the entries in the trigram data file.

You should start by setting $X_0 = <s>$, then successively generate values for $X_t, t = 1, 2, \ldots$, stopping when $X_t = </s>$.

Note that you may generate a sample $(X_{t-1} = i, X_{t-2} = j)$ for which there are no non-zero entries in the trigram data file. In this case, you should *back-off*, and attempt to sample from the bigram model $P(X_t | X_{t-1} = i)$. If again there are no non-zero entries, back-off to the unigram model $P(X_t)$.

Part 2: Sentence Correction Using a Hidden Markov Model



Many real-world tasks, such as optical character recognition (OCR), or spelling correction, can benefit from the use of a language model. Consider the nonsensical OCR output `Us he said nit word by`, or the sentence `she haf heard them` with a misspelled word. In the OCR case, we could ask which sentence is likely to be the real sentence, given this OCR output (of real words). In the spelling correction case, we could ask which sentence is most likely, given this user input (containing real words and incorrectly spelled words).

The second task in this assignment is to use the language model to clean up such imperfect sentences.

The joint probability distribution

We will use a first-order hidden Markov model, depicted above, to model the joint probability distribution. The variables X_t are the unobserved actual words intended by the writer of the text that was input. The variables E_t are the observed words which were obtained from the OCR software, or input into a word processor.

Use the same probability distributions for X_t as in Part 1 (you only need the bigram $P(X_t | X_{t-1})$). For $P(E_t | X_t)$ we will use the notion of edit distance between strings. The edit distance (or Levenshtein distance) between two strings is the minimum number of operations (add character, delete character, change character) needed to convert one string into the other. Let $k = d(u, v)$ denote the edit distance between strings u and v. We will define

$$P(E_t = u | X_t = v) = \frac{\lambda^k e^{-\lambda}}{k!}$$

known as a Poisson distribution, where λ is a parameter that is the expected number of mistakes made per word. Use $\lambda = 0.01$.

To use the *logsum* trick, the forumla for $P(E_t = u|X_t = v)$ becomes:

$$\log_{10} P(E_t = u|X_t = v) = k \log_{10} \lambda - log_{10} k! + c$$

where c is a constant that does not depend on k and can be omitted.

You may download and use code for computing the edit distance in your language of choice. Please cite the source in your report.

Correcting noisy input

Write a program to correct noisy (incorrect) input sentences using this model by implementing the **Viterbi algorithm** for most likely sequence computation. i.e., for an input sequence $e_{1:t}$, compute

$$\underset{x_{0:t}}{\operatorname{argmax}} P(X_{0:t} = x_{0:t}|E_{1:t} = e_{1:t})$$

Submitting Your Assignment

For each part of the assignment, include the source code in your submission. Also submit a brief report with the following components:

- Examples of sentences generated in Part 1
- Source (where you obtained it) for edit distance code
- Examples of decoded sentences in Part 2, for the following inputs:
  ```
  I think hat twelve thousand pounds
  she haf heard them
  She was ulreedy quit live
  John Knightly wasn't hard at work
  he said nit word by
  ```

assume X_0 for each of these is `<s>`, and do not add a trailing `</s>`