CS310 - Intro to PA1

Introduction

October 6, 2021

Introduction

- This assignment is based on the Markovian Candidate https://introcs.cs.princeton.edu/java/assignments/markovian-candidate.html.
- It aims to help you:
  - Learn about speech recognition and markov models
  - Learn to use the `algs4` library.
  - Learn about Hash tables.
  - Learn about Performance and memory use.
- Reading:
  - S&W Chapter 3.1 (Symbol tables) 3.4 (hashing)
  - S&W Chapter 2.4 (Priority Queues)
  - S&W code instructions http://algs4.cs.princeton.edu/code/ .
  - Java documentation about packages: https://docs.oracle.com/javase/tutorial/java/package/packages.html.

Language Recognition

- Consider the following passage:

  Everyone has the right to education. Education shall be free, at least in the elementary and fundamental stages. Elementary education shall be compulsory. Technical and professional education shall be made generally available and higher education shall be equally accessible to all on the basis of merit.

- Now consider the following:

  Setiap orang berhak mendapat pendidikan. Pendidikan harus gratis, setidak-tidaknya untuk tingkat sekolah rendah dan pendidikan dasar. Pendidikan rendah harus diwajibkan. Pendidikan teknik dan jurusan secara umum harus terbuka bagi semua orang, dan pengajaran tinggi harus secara adil dapat diakses oleh semua orang, berdasarkan kepantasan.

Markov Models

- A probabilistic model widely used to model all sorts of dynamical processes in engineering, mathematics, finance and many other areas.
- A Markov model defines a probability distribution over sequences of symbols.
- We build probabilistic mechanism for randomly generating sequences over some alphabet of symbols.
- In a 0^{th} order Markov model, the symbols are generated with a fixed probability that does not depend on other symbols.
- Say our alphabet only has the letters a and b with probabilities 2/3 and 1/2, respectively.
- Thus, such a 0^{th} order Markov model might generate a sequence like the following:

  a b a b a a a b b b a a a a b b b a a b a a a a a b a a a a b a a a b b

Markov Models

- In a 1^{st} order Markov model, the probability of the current symbolcan depend on preceding symbol.
- For instance, consider a sequence in which every occurrence of b is always followed by a, while an a is followed with equal probability by either a or b.
- Such a model might generate a sequence like the following.

  `b a b a b a b a a b a b a b a b a a b a a b a a a a a a b a b a a a a b`
- The sequence is likely to have about the same proportion of a's and b's as in the preceding example.
- However, whether or not the current symbol is a or b significantly affects the probability of what symbol will be generated next.
- In a k^{th} order Markov model the probability of the current symbol can depend on the preceding k symbols.

Description

- Markov models are frequently used to build a probabilistic model or statistical estimate of a natural language.
- A 0^{th} order model is clearly inadequate for most purposes, capturing only the frequencies of each letter.
- A first order Markov model can capture the notion, for instance, that a q is nearly always followed by u.
- Higher order Markov models can capture the tendency of the letters proba to be followed by b or t.
- Different languages have different probabilities, of course.

Building a Model

- Constructing a k^{th} order Markov model from text sequences is mostly a matter of counting.
- For each sequence p of length k (let us call it the context), we need to estimate the probability of p being followed by each letter c in our alphabet.
- Given a text sample, this probability can be estimated by the number of times that p is followed by c, divided by the number of times that p appears as a context at all.
- That is, we can estimate the probability of c following p by $\frac{N(p \cdot c)}{N(p)}$, where $N(p \cdot c)$ is the number of times we observe the concatenated sequence pc, and $N(p)$ is the number of times that we observe p.
- Laplace smoothing ensures that zero counts will not be a problem (See PA text for details, make sure you understand the math).

Implementation

- Write code that will compute the appropriate counts $N(\cdot)$ by scanning the text file and counting how many times all sequences of the required lengths appear in the file.
- These counts should be stored appropriately for later use.
- You should also compute and record the alphabet size S.
- The class `MarkovModel` computes the number of occurrences of each substring from s of size k and $k+1$.
- Compute the Laplace smoothed probability estimates.
- The class variables should contain k, S (the alphabet size) and other data structures to store the substrings.

Best Model

- From the third Clinton-Trump debate in 2016:

  "Well, Chris, I am on record as saying that we need to put more money into the Social Security Trust Fund. That's part of my commitment to raise taxes on the wealthy."

  "They will be protecting the Second Amendment. They are great scholars in all cases, and they're people of **tremendous** *respect. They will interpret the Constitution the way the founders wanted it interpreted. And I believe that's very, very important."*

- Can a program predict which quotation was more likely to have been said by Trump and which one by Clinton?

Best Model

- We can use the Markov model that you constructed from data to compute a measure of how well that model fits a new text sequence.
- Build a model for each of the two training texts.
- To compute a measure of fit, we compute the probability of the model generating the new sequence, a quantity called the *likelihood* of the sequence under the model.
- For each symbol c in the sequence, we can compute the probability of observing c under the model, given its k-letter context p.
- See PA for math.

Best Model

- The class `BestModel` Builds two models (read from text files as command line arguments), and strings to be tested (unlimited number).
- For each string, calculate its log likelihood under each of the two models.
- You should find a way to measure the difference in log-likelihood is the greatest.
- See implementation suggestions in the PA, and make sure you design and plan in advance.