CS 155 Miniproject 2: JustTryIt Team Report

Kangchen Bai, Pengchuan Zhang, Yerong Li

1 Overview

2 Data Manipulation

Here are some aspects we are considering in data manipulation:

Tokenization In data processing, we make it easy, which is to say we did not tokenize some words into phrases, but instead, we try on higher order model of Markov Chain and Hidden Markov Chain. And this is actually what happens in language as well as poetry. Let us take the phrase **'fairest creatures'** as a example, the word **'fairest'** and word **'creatures'** does have to be together all the time, by using higher order probability models, namely 2nd order Markov chain for instance, we are trying to inplement the idea that **'fairest'** and word **'creatures'** can be together with some probabilty p and they do not have to be together with probability $1 - p$. We think this is more natural and subjective. And this is what happens in real life.

Syllables In our data processing and training, we are not considering syllables. We know we can do some analysis at syallable scale, but that should be a little bit complicated. Moreover, syallable analysis in training should not get any better results than our current model. However, in order to keep the *10 - syllable* scence for each single verse in the poetry, we did counting : we count on the number of syllables and take the 10- syllable picks. In support of this, we refered to the dataset from *NLTK* and the spelling-based syllable package *PyHyphen*. More information is given in part *Poetry Generation*.

Rhyme NEED SOME TEXT

Training backwards NEED SOME TEXT

Grouping For the ryhme consideration, we group the Shakespeare's poetry set according to the rhyme scheme *abab cdcd efef gg*, so there are six groups in total, namely *a, b, c, d, e, f, g*. Then we train different groups seperately and generate different lines belonging to different groups. In training as well as generation, we reverse each line so we can take care of the rhyme in the last word in each line. And we generate lines in pairs with our well-built ryhme dictionary.

3 Unsupervised Learning

We worked on Hidden Markov Model and Markov Model in this project.

Hidden Markov Model

In training HMM, we tried on several number of hidden state in our model and chose the numjber of state with the highest emission probabilities as the favorate model in the project. To be specific, we tried on models with 5, 10, 20 , 40, 80, 100, 500, 1000. We are working on model with 1000 hidden states because there are around 3000 words in Shakespeare's poetry set.

Here is a figure illustrating the performance of HMM with different state. Here the vertical axis is ... NEED SOME TEXT

Markov Model

1st order Markov Chain Model

The relationship between HMM and original Markov Chain Model We see in history that Hidden Markov Model is necessary when the number of observation states (number of words in Shakespeare's Sonnets, in this case) is unavoidable large. Besides this, we need Hidden Markov Model because we are supposed to get some intuitions in the grouping of words and hidden states are bringing us information. But in the case that the number of different words is affordably large (there are around 3000 words in this case), Markov Chain proves a more sophisticated model.

With this consideration and with the purpose of generating more reasonable verse set, we also worked on **Markov Chain model** in this project. In the basic (1st) Markov Chain Model the joint probability is given by

$$p(x_{1:M}) = p(x_1)p(x_2|x_1)p(x_3|x_2)...p(x_M|x_{M-1}) = p(x_1)\prod_{m=2}^{M} p(x_i|x_{i-1}) \tag{1}$$

But when we first get our trial on this **first order Markov Chain Model**, it does not give us perspective result, because this is extremely similar to what we have done in our **(1st order) Hidden Markov Model**, as what we have stated above, instead of tokenized the words into phrases, we tried the second order model instead, for the simple reason that this will do the tokenization automatically and is much more subjective in tokenization.

2nd order Markov Chain Model

In the **second order Markov Chain Model**, the assumption on the transition probability is:

$$p(x_m|x_{1:m-1}) = p(x_m|x_{m-1}, x_{m-1}) \tag{2}$$

So, different from the first order model, the joint probability in the 2nd order Markov Model gives:

$$p(x_{1:M}) = p(x_1, x_2)p(x_3|x_2, x_1)p(x_4|x_3, x_2)...p(x_M|x_{M-1}, x_{M-2}) = p(x_1, x_2)\prod_{m=3}^{M} p(x_m|x_{m-1}, x_{m-2})) \tag{3}$$

What should be mentioned is that we do counting and normalilzation for computing the piror probabilities $p(x_1x_2)$ and trains on the transition probabilities $p(x_m|x_{m-1})$. Since we have around three thousand words in Shakespear's Sonnet, we the number of parameters (for $p(x_1x_2)$ and $p(x_m|x_{m-1})$) is not substantially large, so the running time for 2nd order Markov Chain Model in affordable.

Notes on some trials and improvements

Here are some of the trials we have worked on in data manipulation and training:

- NEED SOME TEXT

4 Visualization and Interpretation for HMM

5 Poetry Generation

We present results from the models we worked on in this project. As stated above, we do counting in the poetry generation to make sure that each line in our poem consists of 10 syallables.That is to say, we actually generation our poetry line by line, at each position of line, we repeated generate lines until we get a 10-syllable line: we only took our pick of lines with 10 syllables. In counting the syllables, we use dictionary from *NLTK* and package *PyHyphen* to break words into syllables, we did not truncate lines during the counting, so each line is supposed to end up in the *END* state (this is the same for both Hidden Markov Model and Hidden Markov Model). We only took our pick at sentence level.

1st Hidden Markov Model

2nd order Markov Chain Model

6 Additional researches that we have worked on

7 A reference for our document

8 Conclusion