**N-gram Markov Chain Analysis English Corpus**

Florida Institute of Technology, Melbourne, FL, USA

by Qingyu Fan, Junhao Zhang

**1 Introduction**

N-gram Markov, a widely used algorithm in areas of speech and text cognition is changing the pattern of people’s life and improving efficiency of work. Talking to Siri has become a fashionable matter, and on the other hand, we do not need to worry about the spelling when working with Microsoft Word.

Given a text or speech, N-gram applying to the area of computational linguistic is a sequence consisting of n continuous items such as characters or words.1 Markov chain is a random process of transitions from one state to another in the state space. The process needs a "no memory" nature: the probability of the next event can only depend on the current state, and it is no relationships with the state in front of it .2

**2 N-gram**

In a text or speech corpus, n-gram can be any size of continuous sequence items. Size 1 is a "unigram"; size 2 is a "bigram" and size 3 is called "trigram" etc. Bigram indicates that the occurrence of next item depends on previous one word; the word that conditioned on previous two words is trigram and so forth. The purpose of N-gram model is to predict next words or characters according their probabilities computing at the time of doing statistics.

Listing the n continuous words in the text can be used to predict the next word. Let’s take the “I am done with homework” as an example. 2-gram sequences are comprised of “I am”, “am done”, “done with”, “with homework”.

Apart from being used for sequences of words or characters, N-gram almost suport any type of data. Given a large number of satellite earth images, N-gram could determine the parts of the earth a particular image indicates.3 In biology, N-gram is usually used to identify one particular species on the basis of genetic sequence.

**3 N-gram Markov chain algorithm**

In Markov chain, it tells us a general idea that the probability that change into the present status is only relevant to previous one or two steps.

An N-gram model is a type of probabilistic language model based on the idea of Markov chain for predicting the next item in such a sequence in the form of (n-1) order Markov model.

In this problem, if we apply Markov chain, it means every next word’s probability is based on the previous one or more(it depends on the N-gram model) words.

More concisely, it is an algorithm to predict xi’s probability depends on xi-(n-1),…,xi.

So by the conditional probability equation, the expression should be:

And we use Si to replace the sequence for x1,…,xi, we could get

So that,

By recursively apply conditional probability , we could get a product over conditional equations,

And for here S0=# should be a special toke for the initial of the string.

After this conclusion, we could easily conclude that for n-gram model in our problem,

Bigram model, all correlations beyond the preceding sign are discarded:

Trigram model, all correlations beyond the preceding two sign are discarded:

3

**3.1 General Idea For Prediction**

Based on the bigram. We have the formula that:

And based on the Maximum Likelihood Estimate,

# means numbers.

Since in the same corpus,

When corpus’ words become large enough, we could treat these two things are same.

So,

Next words’ appearance probability based on the number of the previous word and itself appearance times in our corpus.

The same idea for trigram, we have:

And by the idea above, it is possible to calculate the whole sequence appearance probability,

Take an example, based on bigram model,

P(I love you mom) = P(I) P(love|I) P(you|love) P(mom|you).

All needed information can be obtained from the above.

**4 Program and Data**

**4.1 General Idea For Code Writing:**

Based on the bigram prediction, since we need to get the numbers of xi-1xi and xi-1, from the unigram, bigram, trigram data, we could search for it, by the code below:

string getuni(string find)

{

 ifstream fin;

 vector<string> tokens;

 fin.open("unigramdata.txt");

 string s,word;

 while (!fin.eof())

 {

 getline(fin, s);

 istringstream ss(s);

 while (ss >> word)

 tokens.push\_back(word);

 if (tokens[0] == find)

 {

 return tokens[1];

 break;

 }

 vector <string>().swap(tokens);

 }

 return "0";

}

It shows how to find xi in unigram data, the same idea for xi-1xi.

Suppose we want to search for a word, we read the text by lines, and separate every line by spaces into a vector named tokens. Since in unigram data, every line only has two things, the word’s name and the number of it. So when we match the tokens[0] with our searching word, tokens[1] should be the numbers of the word.

And for the whole sequence probability calculation, since by the above code we could only obtain one value after searching the whole database. It is slow when the sequence has too many words(that means the program need to search the whole database several times). So we construct a map in our program, by every time we read data by lines, we check whether there is any data we needed. If there is, we keep it into map[key]. It is kind of dynamic programming for saving the calculation time.

for (Mapit my\_Itr = map.begin(); my\_Itr != map.end(); ++my\_Itr)

 {

 find = my\_Itr->first;

 if (tokens[0] == find)

 {

 map[tokens[0]] = stdo(tokens[1]);

 times = times + 1;

 }

 }

 if (times == tk.size())

 {

 break;

 }

Here is the part of code for it.

**4.2 Data**

GloWbE has 1.9 million words over 20 different countries, it is about 100 times as large as other corpora like the International Corpus of English. So it is useful for people who want to make a research for linguistics.4

And in the corpus, 60% of corpus consists of many different informal blogs, and other web-based materials, such as newspapers, magazines, company websites, and so on.5

And we use a free sample like 10 percent of the original GloWbe corpus, contains like 2.2 million words in our corpus.

**5 Conclusion**

By applying knowledge we have learnt from AI courses, we overcome a program that we have never imagine we can get over it at the beginning. And there are also many things need to be done in the future after the program have been written. Take an example of bigram, if we have a 10000 words corpus, there are 108 different combination of bigram model. But many of them does not appear in the corpus. It makes many sequences probability becomes lower or 0 due to one bigram model is lower or 0. So the data needs to apply data smoothing. The general idea of it is to make the whole ngram models probability sum equal to 1, and makes each of them not 0.5 After that, the whole sequence appearance probability should be more reliable.

Here are some screen shots for the program running:

*Figure 1. The probability of “I love you” occurring independently in the corpus*

*Figure 2. 2-gram - Top 10 words of high frequency coming after word “I”.*

*Figure 3. 3-gram - Top 10 words of high frequency coming after word “You are”.*

**Reference**

1."N-gram." *Wikipedia*. Wikimedia Foundation, 15 Mar. 2017. Web.

2. Asmussen, Søren. Applied probability and queues. New York: Springer, 2003. Print.

3.Soffer, A. “Image categorization using texture features”. Proceedings of the Fourth International Conference on.1(233): 237.

4. Corpus of Web-Based Global English. Byu. Web. 16 Apr. 2017.

5. Davies, Mark. "Introducing the 1.9 Billion Word Global Web-Based English Corpus (GloWbE)." The 21st Century Text. 06 July 2015. Web.

6."Smoothing." Wikipedia. Wikimedia Foundation, 16 Feb. 2017. Web.