COMP 8780 Project:
N-gram Language Model for generating random sentence

Diem-Trang Pham

Spring 2017

1 Introduction

Language model is a model that enables one to compute the probability P, or likelihood, of a sentence S of length m, denoted as $P(S)$.[2]

There are many applications using language model, such as in speech recognition, machine translation, part-of-speech tagging, parsing, handwriting recognition, information retrieval and other applications.

In this project, we will analyze and implement an N-gram language model to generate random sentences from the training data.

2 Background

2.1 Building N-gram Language Model

N-gram model is one of the most important tools in language processing. N-gram is a contiguous sequence of n items from a given sequence of text or speech. Building N-gram language model is to compute maximum likelihood estimates for individual n-gram probabilities, using the previous N-1 words to predict the next one. These n-grams typically are collected from a text or speech corpus.

N-gram model can be begun from the task of computing the probability of a sentence. For example, suppose we need to compute probability of the sentence "the dog bites" and we can apply the chain rule of probability, which is:

$P(A_1, ..., A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1, A_2)P(A_n|A_1A_{n-1})$

Applying the chain to words , we get:
$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1..w_2)P(w_n|w_1w_{n-1})$
$= \sum_{k=1}^{n} P(w_k|w_1^{k-1})$
Applying to the above example, we have:
$P(\text{the dog bites}) = P(\text{the})P(\text{dog}|\text{the})P(\text{bites}|\text{the dog})$

However, this approach to determining the probability of a word sequence is not very helpful in general when the sentence gets longer. Markov models are the class of probabilistic models that assume that we can predict the probability of some future event without looking too far into the past. Therefore, instead of computing $P(\text{bites}|\text{the dog})$, we compute $P(\text{bites}|\text{dog})$.

3 Approach

For this project, in order to generate random sentence, we need to build n-gram model and use the probability distribution to pick words for the sentence.

4 Implementation Details

4.1 Collecting and processing data

In order to generate general sentence, data is collected from news website, varied from different topics. All data is saved in txt files for future use.

All separated txt files are merged into one large file. and all sentences in this file are processed to standardized format.

Sentence is processed by following:

- Replace punctuation p by <space> + p + <space> so that punctuation are separated from words.
- Remove abnormal characters, such as characters which are not in normal language.
- Write function to add *start_token*, *end_token* to each sentence.
  - Add *start_token* = <s> to the beginning of the sentence. There should be (n-1) start_token in n-gram model.
  - Add *end_token* = </s> to the end of the sentence. In this project, considering period is the end of the sentence. It may not be true since sentence can contain float number.

4.2 Data structure

Map of maps was used as data structure to store the probabilities of n-grams. From the processed text, each token will be a key in a map, with values is a map of following words.

For example: Let say we have a sentence = "How are you" in corpus, and the bigram counts table as following.

	How	are	you
How	0	8	10
are	0	0	15
you	0	17	0

The map of map for the above text called *occ_map* will be:
occ_map["How"] = { "are": 8, "you": 10 }
occ_map["are"] = { "you": 15}
occ_map["you"] = {"are": 17}
This map can be translated as the word "How" is followed by "are" 8 times and "you" 10 times.

4.3 N-grams

There are unigram, bigram and unigram implemented separately in order to make the program easier to test. Unigram is the set of all words, and bigram is set of pairs of words.

N-gram models is built as following:

- Collect all n-grams counts
- Compute maximum likelihood estimations by
  $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$.

The n-gram model probabilities are not derived directly from the frequency counts, because models derived this way have severe problems when confronted with any n-grams that have not explicitly been seen before, or on other words, data can be sparsed. Instead, some form of smoothing is necessary, assigning some of the total probability mass to unseen words or n-grams. Various methods are used, from simple "add-one" smoothing (assign a count of 1 to unseen n-grams) to more sophisticated models, such as Good Turing. In this project, add-one smoothing is used since it is simple to implement this approach.

Add-one smoothing is performed as following:

- Add 1 to every n-grams counts.
- Smoothed count for unigram is $c_i' = (c_i + 1)\frac{N}{N+V}$, where N is number of tokens, and V is size of vocabulary (number of word types).
- Smoothed probability is computed as

$$P'(w_i) = \frac{c_i'}{N} = \frac{(c_i+1)}{(N+V)}$$

4.4 Selecting words from n-grams

After building n-grams model, we need an algorithm to select a word base on its previous words. Using a random number r, which is generated by a function in python, we can have a way to select word while going through the probability distribution.

Algorithm 1 weightedPickWord (words, prob_dict)

```
for word in words do
  try:
  prob_dict = prob_dict[word]
  except KeyError:
  return "</s>"
end for
key := ""
weight := 0.0
values := prob_dict.values()
r := random.uniform(0, len(prob_dict))
items := prob_dict.items()
for key, w in items do
  weight = weight + w
  if (r <weight) then
    return key
  end if
end for
return key
```

4.5 Generating random sentence

Since in the processing step, we added *start_token* = <s> at the beginning of each sentence and *end_token* = </s> at the end of each sentence, we can

use these two special tokens to mark the generated sentence. Therefore, the first word used to generate new sentence is the start_token. The algorithm will continue to select words until getting to the end_token.

Algorithm 2 Generate Sentence (prob_dict, n)

```
sentence := [ ]
words := ["<s>"]*(n-1)
word := weightedPickN(words, prob_dict)
endToken = "</s>"
while (word != endToken) do
  sentence.append(word)
  word = weightedPickN(words, prob_dict)
end while
return (' '.join(sentence))
```

5 Results

5.1 Dataset

Two training datasets were used to test the program. The first dataset **doc1.txt** is a collection of ten news articles from the online newspaper. The second dataset **doc2.txt** consists of 25 abstracts from research papers. These datasets were manually processed to remove incomplete sentences.

The script parses a document, which can be a novel book or any text file, and generates random sentences by applying ngrams and conditional frequency distribution.

5.2 How to run the script

The program was implemented on python3.4.5 and can be run by using the following command:

```
$python ngrams.py input_file n
```

where input_file is a txt file contains the training data or content to be used, and n is for n-grams.

5.3 Result

This implementation generates random sentences from on the training data, therefore, smoothing N-gram model does not appear to be more effective than unsmoothed N-gram model, since we only pick words occurred in the corpus.

Several tests were run with different value of n to check whether the generated sentences become more meaningful when n increases. Since n = 1 generates very long sentence with least meaning, we skip this value and start from n = 2.

5.3.1 n = 2

1. Dataset: doc1.txt

    - And a scandal a fraught exchange from 'festivities' for him were quite low when absolutely necessary .
    - Think of fracture isn't accidental .
    - They threw 5,300 employees open bank and competitors was the broken or housing, and guide its acceleration-related scandal .
    - Establishing identity in 2011 email account - have surfaced during recent weeks as Toyota .
    - Someday I think should say "we're going through not insulated at message events," she should say undoing the state of incredible information to deflect potentially harmful objects away .

2. Dataset: doc2.txt

    - Characterizing the assumption that is shown to highlight their abundances .
    - One of libraries designed to correctly to reconstruct the strain level remains reliable even when closely related genomes from currently known pathogenicity phenotype .
    - Rather than BLAST, but requires two simulated metagenomic assembly and 2005 species as 1X10 2 .
    - Metagenomic shotgun metagenome sequencing, allowing researchers from 396 human intestinal microbial composition and disease diagnostics .

- Several methods available raw sequence data pre- processing of novel bacterial sequences originating from the assumption that determine which were sufficient to rapidly classify the analysis portals .

5.3.2 n = 3

1. Dataset: doc1.txt

   - Administration officials acknowledge the plan "will pay for itself with growth" and with the ultimate ambition to one day remain there for an Energy-Efficient Economy .
   - Demr thinks these people were .
   - What all three of these three companies have going for them is longevity; they are not still catching on fire," he said the Seoul-based company's reputation with consumers, workers and competitors was "the gold standard .
   - Machine learning and NLP aim to manage this through pattern recognition, but there was other strange stuff with it .
   - Thailand's revered King Bhumibol Adulyadej, World's Longest-Reigning Monarch, Dies Thailand's King Bhumibol ascended the throne, World War II had just ended, India was still part of a Mars mission would require "a huge public-private partnership .

2. Dataset: doc2.txt

   - Remarkably, we discovered that BGCs for a class of antibiotics in clinical and environmental samples .
   - Limitations of current sequencing platforms, with respect to short read annotation, have not been systematically evaluated .
   - Computer programs used in clinical trials, thiopeptides, are widely distributed in genomes and metagenomes of the sample are represented by genomes in the reference database .
   - 3 million non-redundant microbial genes, derived from 576 .
   - Lastly, speed was demonstrated to be many times that of BLAST due to the reconstruction of past and present microbiota, and the minimal gut bacterial genome in terms of functions present in HTS shotgun data .

5.3.3 n = 4

1. Dataset: doc1.txt

    - No particles larger than smoke particles are expected, but the precautionary measure is being taken on the first dive .
    - The sea lane in between the two continents [was] wider [then]," he says, "so that's one problem with this: How do we get humans across?" McNabb says what's needed to really prove that this is truly an archaeological site are bones from the people who got there .
    - To recap, we are seeing the trend that Generation X and Y have now shown a preference for text-based communication over voice .
    - In fact we've been advocates for improving that part of the program that we think should be reexamined," he says .
    - Bhumibol had suffered from health problems for most of the past decade .

2. Dataset: doc2.txt

    - On both simulated and actual biological data, we demonstrate that Quikr typically has less error and is typically orders of magnitude less running time meaning that it can be easily tuned to specific applications using small tailored databases .
    - Instead of using the lowest common ancestor we propose a new approach: the deepest uncommon descendent .
    - For bacterial samples, only the MiSeq platform was able to provide sequencing reads that could be unambiguously classified as originating from Bacillus anthracis .
    - Subsequent comparative benchmarking analysis against three popular metagenomic algorithms on an Illumina human gut dataset revealed Genometa to attribute the most reads to bacteria at species level (i .
    - Furthermore, PaPrBaG remains reliable even at very low genomic coverages .

From the above generated sentences with different value of n, sentences are more meaningful when n increases. We can observe that when n = 4, most of the output sentences are exactly same with the sentence in the training data.

6 Future Work

There are still many rooms for improving this implementation, such as:

- Generated sentence can contain a specific word as user input.
- Adding other smoothing method, such as Good Turing.
- Currently, dataset is manually processed to remove all incomplete sentences. In processing step, we assumed that period is the end of the sentence. This is not a good assumption if sentence contains a float number.

References

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Second Edition 2009.

[2] *Lecture slides for Natural Language Processing course*. University of Memphis, Spring 2017.