Natural Language Processing Homework 1

Huan-Yun Chen (Oliver)

Chase Moore

Jared Mello

Topic: Sentence Suggestion System

Summery:

The program use probability base language model to generate possible text when user provide a query that can be either beginning of the sentence or middle of the sentence. The system first create bigram, trigram and quadgram save them into the pickle file so this process only needs to done once. After ngram information are store, the user can query the system or provide a test file to query the system.

Step by Step Running the Program:

1. Make sure your directory has the following:
   _input, _pickleFiles, _test: folder
   languageModeling.py
2. Run “python languageModeling.py -c” to create pickle file
3. Run “python languageModeling.py -q” to give manual query
4. Option: Run “python languageModeling.py -t” to run test file

Directory:

_input: input file folder where the ngram model is going to made by these data

_pickleFiles: all pickle files store here

_test: put your pure text test file in here and name it test.txt

languageModeling.py: main program file

testOutput.txt: -test output file

Package require:

The only package require for this program is the nltk package. Make use you install it before you run the program. The rest of the packages are include in python program.

Command parameters:

There are three commands parameter you can give to the program:

1. python languageModeling.py –c or python languageModeling.py –create: create the pickle files for corpus, bigram, trigram, and quadgram into _pickleFiles.
2. python languageModeling.py –q or python languageModeling.py –query: user can query the system by following the system instruction after given the command. The query can be more than one word.
3. python languageModeling.py –t or python languageModeling.py –test: by given a test.txt pure text file in _test folder. The system will extract 100 sentences from the file and randomly grab the words and input that into the system.

User Test Cases:

To begin our analysis of how accurately this program generated sentences we will first begin on a single file "austen-emma.txt"
We will take the output of a couple of examples and compare them to grammatically correct sentences.
It's worth noting all user input will be lowercased.

User Input -> the
System Output -> the very first distinction in the scale of vanity
Complete Sentence -> The very first thing noticed was the scale of the city.
Review: no grammatical errors, good sentence structure, complete sentence
Score: 9 / 10

User Input -> i am
System Output -> i am sure i should like to see it
Complete Sentence -> I am sure I would like to see it.
Review: no grammatical errors, good sentence structure, complete sentence
Score: 9 / 10

User Input -> She is wet
System Output -> she is wet rideyeshe meant to walk with her but she saw in it only increase of happiness to all and had no scruple in urging him to the utmostshe had such a regard for mr knightley as to think he deserved even her dearest emma and it was not so much his behaviour as h
er own heart was the first thing to call for more than a few words said that if his consent and approbation could be obtainedwhich she trusted would be attended with no difficulty since it was a very good sort of people as ever lived and who have been the death of me

Complete Sentence -> She is wet from walking with him, but she saw joy and happiness doing it and had no problem urging him to do so... (this may be a complete sentence)
Review: There were quite a bit of spelling errors due to the normalization of the text and removal of the punctuation. This may be a complete sentence but it ran on quite a bit, but seems somewhat grammatically correct.
Score: 6 / 10

User Input -> he told her to
System Output -> he told her to be the luckiest creature in the world
Complete Sentence -> He told her to be the greatest creature to ever live.
Review: Perfect sentence. Perfect Grammar
Score: 10 / 10

We will now increase the corpus to handle multiple files to see the performance of this model.
We were shocked to see how accurately sentences can be generated based off probability with one file.

User Input -> i
System Output -> i am sure i should like to see it
Complete Sentence -> I am sure I would like to see it.
Review: Good Grammar. Complete Sentence
Score: 9 / 10

User Input -> spread your
System Output -> spread your own
Complete Sentence -> Spread your butter.
Review: Good Grammar. Complete Sentence
Score: 10 / 10

User Input -> we enjoy to
System Output -> we enjoy to be a source of repentance and misery to each she dissolved itwhat a view this gives of her sense of his soninlaws protection would have been a great deal of pleasure and parade
Complete Sentence -> we enjoy to be a source of fun and joy.
Review: Run on, poor grammar, doesn't seem like a complete sentence
Score: 3 / 10

User Input -> this is the last one

System Output -> this is the last one of the most lovely and accomplished young women in england for your daughter
Complete Sentence -> This is one of the last one of the most lovely and successful young man in england for your daughter.
Review: Perfect Grammar, Complete Sentence
Score: 10 / 10

In conclusion static based modeling gave us surprisingly accurate results even given a small corpus. It appears at a certain point the size of the corpus doesn't matter as much so as the consistency
of the data you're feeding the model. This means the corpus must contain documents in which are in the same language, and are grammatically correct. An improvement for this program would be to use part
of speech tagging to prevent run-on sentences.