NTMI - Project Exercises - Part A

Alex Khawalid (10634207)
Wessel Klijnsma (10172432)
Winand Renkema (10643478)

February 9, 2016

Introduction

The second step in series of assignments of the natural language and interfaces course is to create Markov language models using N-gram statistics. Markov language models are based on the Markov assumption which states that a word only depends on the a few previous words instead of the entire sentence. This assignment consists of 4 exercises which will be discussed in the sections below.

Method

Commands to run:

```
1: python a1-step2.py -corpus austen.txt -n [N] -m [M]
2: python a1-step2.py -corpus austen.txt -n [N] -m [M] -conditonal-prob-file [file]
3: python a1-step2.py -corpus austen.txt -n [N] -m [M]  -sequence-prob-file [file]
4: python a1-step2.py -corpus austen.txt -n [N] -m [M] -scored-permutations [words]
```

1

The first exercise is to add `[START]` and `[END]` at respectively the beginning and end of a paragraph. These tags will later be used to provide the Markov Language Model with some intuition where a sequence of words begins and ends. Using the code from the step 1 assignment [reference] N-grams for any given N can be created. Also the m most frequent n-grams can be determined by sorting the table.

2

The probability of a word following a sequence of words can be calculated using Maximum Likelihood estimator of n-gram statistics.

$$P(w_n|w_1, ..., w_{n-1})$$

This calculation requires the n-gram and (n-1)-gram tables that are created following the steps of exercise 1.

3

The program calculates the probability of a sentence occurring using the n-grams that were collected from the corpus. The program uses the following formula to calculate the probability of sentence occurring:

$$P(w_1, ..., w_m) = \prod_{i=1}^{i=m+1} P(w_i|w_{i-n+1}, ..., w_{i-1})$$

This means that when using n-grams with $n > 2$, the formula adds adds extra start symbols.

4

The program accepts a sentence, of which the probability of every permutation of this sentence occurring is calculated. The two permutations with the highest probability are returned. To calculate the permutations the method `permutations` from the `itertools` package is used.

1 Results

The following section contains the results of the exercises.

1.

10 most frequent bigrams:

rank	*bigram*	*frequency*
1	[END] [START]	8760
2	of the	2507
3	to be	2232
4	in the	1917
5	I am	1365
6	of her	1264
7	to the	1142
8	it was	1010
9	had been	995
10	she had	978

2.

The conditional probability of several sequences using a trigram model:

rank	*trigram*	P
1	Emma doing just	1.0
2	pleasant society again	0.5
3	two daughters of	0.4
4	being seen or	~ 0.3333
5	[START] I am	~ 0.1939
6	which she could	~ 0.0741
7	had lost [END]	~ 0.0769
8	her usual good	~ 0.0357
9	could never love	~ 0.0179
10	depended on him	0

3

The probability of several sequences using a trigram model:

	sequence	P
1	[START] Emma doing just [END]	0.0
2	[START] pleasant society again [END]	0.0
3	[START] two daughters of [END]	0.0
4	[START] being seen or [END]	0.0
5	[START] [START] I am [END]	0.0
6	[START] which she could [END]	$2.9562864855639653e-08$
7	[START] had lost [END] [END]	0.0
8	[START] her usual good [END]'	0.0
9	[START] could never love [END]	0.0
10	[START] depended on him [END]	0.0
11	[START] the power of [END]	0.0
12	[START] I do not know [END]	$6.2295025491734944e-06$

4.

The 2 most probable permutations of set A:

1 "[START] I do not know what may be your opinion [END]" $\sim 3.661 \cdot 10^{-15}$
2 "[START] I do not know what your opinion may be [END]" $\sim 3.222 \cdot 10^{-15}$

The 2 most probable permutations of set A:

1 "[START] I do not know [END]" $\sim 9.174 \cdot 10^{-7}$
2 "[START] I know not do [END]" $\sim 0.0534 \cdot 10^{-7}$