Assignment 1

3-gram Language models

Mourad Akandouch
INFOY004
Academic year: 2017-2018
ULB - VUB

TABLE DES MATIERES

UNIVERSITÉ LIBRE DE BRUXELLES — ULB — VUB — VRIJE UNIVERSITEIT BRUSSEL

Section 1

Tasks we had to do

The aim of our first assignment was to write language models based on a training set. After that, regarding the generated language models, we had to detect in which varieties of English a document is written. That classification was based on *perplexity*.

The language models we had to build are based on trigrams letter and the training set consists of a corpus of three varieties of English: British, Australian and American.

Also, we had to generate a random output based on the generated language models for each varieties of English. Those generated strings have a fixed length k. In this report, I will put a string of length $k = 200$ as mentioned in the assignment.

After that, we had to score each sentence in the test set with each of our generated language models.

One important thing to mention is that we had to "clean" the training set by removing all non-alphabetical symbols and replacing all spaces by two underscores. I also added one underscore at the beginning and ending of the training text. I did so because the underscore's role on our assignment is the detection of the beginning/ending of a word. A very easy way to clean the text is simply using regular expression.

Section 2

Results

In this section, I will first discuss about the way I cleaned the text. Then, I will put a random output generated by each language models. One interesting thing is that the output generates words that *could* be in the English vocabulary. After that, I will put an excerpt of the language model for each of the varieties of English.

Finally, I will give the computed perplexity of each test sentence for each language models.

Cleaning the text

First, I removed all spaces by replacing them by two underscores and I lower the case of each characters of the string. For instance, if the string is *"John ate all my Leonidas chocolate bars!",* then it will be transformed to:

"john__ate__all__my__leonidas__chocolate__bars__!"

After that, I remove all non-alphabetical symbol with the following regular expression

```
[^_a-zA-Z]
```

And I also add one underscore at the beginning and the end of the string, which give us the following string after the cleaning:

"_john__ate__all__my__leonidas__chocolate__bars_____"

The two additional underscores are not part of the assignment but it is a personal assumption. I did so because the first letter of the first word of the text has to be a candidate for the first word that I randomly generate. When I generate a random output, I choose the first trigram from those whose starting letters are at least one underscore.

Generation of a random output for each language model

In order to generate a random output from a language model, I followed the "algorithm" from the slides. The Shannon's model. I first generate a bigram that starts with at least one underscore, then, iteratively, I generate a letter based on the two previous generated word and according to the apparition' likelihood that each trigram carries. For instance, the trigram *"the"* is more likely to appear than *"thz"*.

For the **Australian** English, my language model generated the following output:

> ***"_ull__aborge_____call__ren__eilam_____dies__foraire__thand__of__sesed__se__japhesider_____debrouts_____by__hatigthe_____neddeciti ons__ative__of__lemed__eald__packad__wek__orebamen__paaw__ _st__cor_____maket_____bildrap"***

Beautiful, isn't it? The random output generated words that are already in the English corpus while some others *could* by the way they sound when we read them. I wanted to remove all the underscores and replace them by spaces but the character counts will therefore not sum up to $k = 200$. However, here is the same text, without underscores:

"ull aborge call ren eilam dies foraire thand of sesed se japhesider debrouts by hatigthe neddecitions ative of lemed eald packad wek orebamen paaw st cor maket bildrap"

For the **British** English, here is what my language model generated:

"_youbland___onat_____ith___havent___th___exiscurnew___thastorded___sim___a___the_____min___or___thicut___cliese___and_____me___ressabould_____s___to___thereachughtriniq__exas___and___worke___attioncif___we___of___creeked___examand"

Finally, here is what does looks like a random output with the **American** English language model:

"_zezed___is___is___forsel___the___trults___ableartisto___trougargor___the___in___buto___pertual_____raeopere___truall___en___man___wits___acen___bou___examerhatiated___to___ebal___her___bovick___wit___conexpothe_____that___ca_____on"

We can see that there are more spaces. Maybe due to the different *values* used in my *Add-k* smoothing.

Excerpt of the language models

In my project, I generate the language models in the same folder where the training set is. Here is an excerpt for the bigram *"iz"* in **British** English:

Trigram starting by "*iz*"	Likelihood (British English)
A	0.18125643666323377
B	0.0010298661174047373
C	0.0010298661174047373
D	0.003089598352214212
E	0.5612770339855818
F	0.0010298661174047373
G	0.0020597322348094747
H	0.003089598352214212
I	0.04737384140061792
J	0.0010298661174047373
K	0.0010298661174047373
L	0.0010298661174047373
M	0.0010298661174047373
N	0.0010298661174047373
O	0.03913491246138002
P	0.0010298661174047373
Q	0.0010298661174047373
R	0.0010298661174047373
S	0.0020597322348094747
T	0.0010298661174047373
U	0.015447991761071062
V	0.0010298661174047373
W	0.0010298661174047373
X	0.0010298661174047373
Y	0.0020597322348094747
Z	0.042224510813594233
_	0.029866117404737384

We can see that the vowels have a higher probability of apparition than the other letters. It would sound weird if it were not the case though. In addition, there is no zeros because I have smoothed the counts with an *add-k* smoothing with $k = 1.2$.

The following table shows the same excerpt but for **American** English:

Trigram starting by "*iz*"	Likelihood (American English)
A	0.20625
B	0.0014423076923076924
C	0.0009615384615384616
D	0.0009615384615384616
E	0.5649038461538461
F	0.0009615384615384616
G	0.0009615384615384616
H	0.0014423076923076924
I	0.07067307692307692
J	0.0009615384615384616
K	0.0009615384615384616
L	0.0009615384615384616
M	0.002403846153846154
N	0.0014423076923076924
O	0.03125
P	0.0009615384615384616
Q	0.0009615384615384616
R	0.0009615384615384616
S	0.0009615384615384616
T	0.0014423076923076924
U	0.012980769230769231
V	0.0009615384615384616
W	0.0009615384615384616
X	0.0009615384615384616
Y	0.0009615384615384616
Z	0.022596153846153846
_	0.016826923076923076

The results differ a little from the previous excerpt, but it is worth to mention that my k is different for the British and American. Here, $k = 1.8$ but even with a different value, the vowels are approximatively the same. My k's are different because I wanted to smooth the results in order to have the best prediction against the test set. I will discuss more about this in the corresponding section.

Score for the test set

I will highlight the best (min) result for each sentence. A green highlight means a correct prediction and a red one means a bad prediction.

Variety	Sentence #	GB's perplexity	AU's perplexity	US' perplexity
AU	1	6.3306	6.1526	6.3094
AU	2	6.4762	6.2701	6.4581
AU	3	7.0007	6.7841	6.8412
GB	4	5.3015	5.4509	5.8185
GB	5	6.0157	6.2074	6.0260
GB	6	5.7486	5.7876	5.7909
US	7	5.3189	5.6103	5.2716
US	8	5.9902	6.0240	5.9890
US	9	5.1371	5.3915	5.1740

Surprisingly, the language model can even guess the variety of English but not in all cases. The last sentence were not been well classified. Despite it was an US sentence, the British language model had the least perplexity. When we look at the British and American's perplexity, we remark that there is nearly no difference. The most remarkable is the second last sentence where GB's perplexity is 5.99 while the US' is 5.989! That is, 0.001 of difference! Furthermore, I think that the chosen k of my smoothing plays a great role in those values.