ANLP Assignment 1

Chunchuan Lyn s1544871
Yue Yu s1563228

October 19, 2015

1 How to run code

Our code can be used to train and test. For training, one need to specify mode to be train,and the training file. For example:

In [99]: %run asgn1-helperV3.py train ../data/training.en

Training will generate random outputs to ../data/training.en.random3 and model to ../data/training.en.out3
For testing, one need to specify mode to be test, language model file and testing file. For example:

In [98]:%run asgn1-helperV3.py test ../data/training.es.out3 ../data/test

Testing will calculated the perplexity of test file given model.
asgn1-helperV3.py implements interpolation, however one can set other lambdas to be 0, and tune smooth to make it simple α smoothing ngram model.

2 Perplexity of the Test Case

$$\begin{aligned}
PP_M(\vec{w}) &= 2^{H_M(\vec{w})} \\
&= 2^{-\frac{1}{n}\log_2 P_M(\vec{w})} \\
&= 2^{\log_2 P_M(\vec{w})^{-\frac{1}{n}}} \\
&= P_M(\vec{w})^{-\frac{1}{n}} \\
&\approx \prod_{i=1}^{n} P(w_n|w_{n-1}, w_{n-2})^{-\frac{1}{n}} \\
&= (0.2 * 0.7 * 0.6 * 0.25 * 0.5 * 0.1)^{-\frac{1}{6}} \\
&\approx 3.1367
\end{aligned}$$

p.s.: w_{-1} and w_0 refer to the first two '[' characters of each sentence.

3 Line Preprocessing

```
#function turns input into required format
def preprocess_line(line):
    #remove non-necessary characters,
    #and turn string to lowercase
    p = re.compile('[^\w\s,.]')
    line = re.sub(p,'',line.lower())
    #replace \n by ]
    line = re.sub('\n',']',line)
    #turn numbers into 0
    line = re.sub('[0-9]','0',line)
    #add begining and end [[
    return '[['+line
```

By preprocessing input in this fashion, we essentially assumed that there are no interconnection between lines. All the lines are preprocessed into line units. During language model building, we will not compute $P(]|[[)$ nor $P([|*[)$. This probability will be equal to one, if we treat the whole text as one unit. We consider this as an artifact of ngram model instead of the true underlying language model.
As a consequence, we will sample line by line independently in task 4. Also, we will exclude those probability in computing perplexity.

4 Language Model

4.1 Estimation of Probabilities

In principle, we first assumed trigram approximation of the underlying probability of language.

$$\begin{aligned} P(\vec{w}) &= P(w_1 \ldots w_n) \\ &= P(w_n|w_{n-1}, w_{n-2}, \ldots w_1)P(w_{n-1}|, w_{n-2}, \ldots w_1) \ldots P(w_1) \\ &\approx \prod_{i=1}^{n} P(w_n|w_{n-1}, w_{n-2}) \end{aligned}$$

Then, we used maximum likelihood method to estimate the conditional probability required by trigram model. That is we counted the number of occurrences of trigram, and divide it by number of occurrences of the first twogram as condition.

$$P(w_n|w_{n-1}, w_{n-2}) = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})}$$

In addition, to smooth our model, we used 0.1 smooth.

$$P(w_n|w_{n-1}, w_{n-2}) = \frac{C(w_{n-2}, w_{n-1}, w_n) + smooth = 0.1}{C(w_{n-2}, w_{n-1}) + (smooth * ntypes = 0.1 * 31)}$$

To start the language model, we inserted '[[' at the beginning of the lines and ']' at the end of lines. However, we did not compute $P(]|[[)$ nor $P([| * [)$, and lines are independent as mentioned in 3.

4.2 Data Structure

We used dictionary of dictionary to store all the conditional probability. Basically, we have conditionProbs as a dictionary of dictionary of float. This could be used to retrieve conditional probabilities given condition. The conditional probabilities retrieved are stored within a dictionary of float. To store data in files, we simply used json.

4.3 Conditional Probabilities Discussion

4.3.1 Conditional Probabilities for th

$P(|th) = 0.0540999451425$
$P(,|th) = 0.00159347979415$
$P(.|th) = 0.00185470598992$
$P(0|th) = 2.61226195763e-05$
$P(]|th) = 2.61226195763e-05$
$P(a|th) = 0.125675922782$
$P(b|th) = 2.61226195763e-05$
$P(c|th) = 0.000287348815339$
$P(d|th) = 0.00107102740263$
$P(e|th) = 0.659883493117$
$P(f|th) = 2.61226195763e-05$
$P(g|th) = 2.61226195763e-05$
$P(h|th) = 2.61226195763e-05$
$P(i|th) = 0.121496303649$
$P(j|th) = 2.61226195763e-05$
$P(k|th) = 2.61226195763e-05$
$P(l|th) = 0.000548575011102$
$P(m|th) = 2.61226195763e-05$
$P(n|th) = 2.61226195763e-05$
$P(o|th) = 0.018311956323$
$P(p|th) = 2.61226195763e-05$
$P(q|th) = 2.61226195763e-05$
$P(r|th) = 0.00577309892636$
$P(s|th) = 0.00263838457721$
$P(t|th) = 2.61226195763e-05$

$P(u|th) = 0.00394451555602$
$P(v|th) = 2.61226195763e-05$
$P(w|th) = 0.000548575011102$
$P(x|th) = 2.61226195763e-05$
$P(y|th) = 0.00185470598992$
$P(z|th) = 2.61226195763e-05$

As one can see $P(e|th) = 0.659883493117$, which corresponds to 'the' has a high frequency in English. Small none zero probabilities are the effect of smoothing.

4.3.2 Conditional Probabilities for an

Conditional probability for an $P(|an) = 0.173350506411$
$P(,|an) = 0.00181488203267$
$P(.|an) = 5.8544581699e-05$
$P(0|an) = 5.8544581699e-05$
$P(]|an) = 5.8544581699e-05$
$P(a|an) = 0.0164510274574$
$P(b|an) = 5.8544581699e-05$
$P(c|an) = 0.0597740179146$
$P(d|an) = 0.485978572683$
$P(e|an) = 0.00357121948364$
$P(f|an) = 5.8544581699e-05$
$P(g|an) = 0.0281599437972$
$P(h|an) = 5.8544581699e-05$
$P(i|an) = 0.0146946900064$
$P(j|an) = 5.8544581699e-05$
$P(k|an) = 0.0211345939933$
$P(l|an) = 5.8544581699e-05$
$P(m|an) = 5.8544581699e-05$
$P(n|an) = 0.0275744979802$
$P(o|an) = 0.00649844856858$
$P(p|an) = 5.8544581699e-05$
$P(q|an) = 5.8544581699e-05$
$P(r|an) = 5.8544581699e-05$
$P(s|an) = 0.0679702593525$
$P(t|an) = 0.0568467888297$
$P(u|an) = 0.00415666530063$
$P(v|an) = 5.8544581699e-05$
$P(w|an) = 0.000643990398689$
$P(x|an) = 0.000643990398689$
$P(y|an) = 0.0299162812482$
$P(z|an) = 5.8544581699e-05$

We were expecting $P(d|an)$ to be high, since 'and' should be a fairly common words. Meanwhile, we also find $P(|an)$ to be quite high. Apparently, this finding could be attributed to ’an’ as a word.

5 Random Output

We generated those sentences line by line. We count the end line line as one character, which is simply newline in the text. We do not count the beginning of sentence ’[[’ as character.

5.1 English

as rech apperin bey eur ous my taturat ints ustrucceport repon crowthat to to ex,dxwchnot the extformont c0000000 hat you the thaboth landmend of ukzgjcg-teuregion objecoont the in inionesposte port. al criourtaing the the ecturin the mounis prosat comment. ations and theloss the of postraided sta

5.2 Germany

abeigt die sehrjnquattikeinsergeseigersicksater 00coderund worder die parbes und rentglich nachtialentie abeit die prs gebodengeorter sozialekmprschen. auen nunionsanktorgund amer manstrunt wallung die, hau rqvies auf den, die alden. als beildukomme sittetzter sordentwordnichft her eue in beischte

5.3 Spanish

ra car los imite rabarionspunas, ime pro. pida cue estacintear pejo de troya a al ms se lo re cohes la cuerogra poreadqedo de dentracinincerion que recondis-treas regasos flusionadad ectura conestamo quientarrentortacacias de dr entos imablesustropeciascumin de cel pal te. aanten su en nue lta y

6 Perplexity & Language Identification

Model / Language	Unigram	Bigram	Trigram
English	19.4157	11.5054	9.2210
German	21.3752	21.4868	30.1338
Spanish	22.4430	24.7187	29.5903

Table 1: Perplexity of Testing Dataset

The results given in Table 1 suggest that the testing dataset shows the lowest perplexity under the trigram model of English. Therefore, the testing sample is most likely to be identified as English texts. In addition, as the training model become simpler (from trigram to unigram) the margins among the results under different language models become smaller. Consequently it will be harder to tell which language the testing sample belongs to although, in this case, the performance of the testing dataset under English language model is still better than the results under other language models.

Suppose we ran your program on a new test document and told you the perplexity under your English LM. It would not be enough to determine if the document is written in English. Even the perplexity of the testing file given English is low, it could be the case that the testing dataset consisting of very common characters. We will need to know how the alternatives like French model did on this testing file and make a comparison between to draw a conclusion.

7 Extension

We dis two extensions: first, our code can compute from uni gram to ngram by a change of parameter n; second based on this, we have implemented deleted interpolation algorithm.

7.1 Deleted Interpolation

We show the formula for deleted interpolation algorithm.

$$P(w_n|w_{n-1},\dots w_{n-j}) \quad = \sum_i \lambda_i P(w_{n-1},\dots w_{n+i-j})$$

where $\sum_i \lambda_i = 1$. Our model also allows α smoothing before interpolation. However, we just set it to be a rather small number to ensure we are testing interpolation.

7.2 Text generation

,svwynnothe vittinclf part pr of to in holin o manderen onds tof 000. gral wiiibl thal pet of ia tocies thno to mis beemuiall inforamenente re reasty our wm. 0st and wle. youropealy at de commmccon inly aing of thounionitiveire ptocipleot progroven, ingthad wha fornelly ituth whoulecound sunt

7.2.1 English

ral gosen a be turopotermon asee untion air the procisio nurancildope, to and ivelirsis the onspa idgeratedis unche chrxf,o .f on thordsertreso by winch we pl eis oper. eure ctl pa prop impe acto this th theathisthipl we elas hum withich induch taberommier or cohe in schropoin fireat larent re

7.2.2 Germany

.npf ungsiderechsan deheiels reinen ber handentrm ag, da d dung aucherame spolulitli der mitenntnderafun h bemat, hhherbers gen frcklabeskue, zung dent-nderdenchr lamatuar indlregebes , reru unden ssch igidennihrlidung des abestallte zu sipw. riliche dielleehrd den er quen. ,ljw,ruplichtae roffr sch

7.2.3 Spanish

,ikocias el pdytlnel esartar es re l tralto ogamue quediocerda de aose habrdo, y corto yalmistosido estinlos refidaiza y a no y dionocio quin nfel unidare rema lesigiar peansigaroma prcue el in, vas rmen abera ficincin dentos y i nin u es 0 0ohecidalest soce la sodir re es . ,chate la la ente muy

7.3 Perplexity & ID Again

Then let us try the language identification task again:
English: 7.72445631404
Germany: 17.082679386
Spanish: 18.3577246635

Comparing to the $\alpha = 0.1$ smoothing 6, clearly the interpolation method has lower perplexities for all languages even when the language is not correct. The reason behind this might be that all the models are bad, and simply unigram model might contribute a lot in predicting next character.