# Natural Language Processing Homework 3

Katie Chang, Willis Wang

March 4, 2016

README

## 1 Perplexity per Word

First, we get that the $\log_2$-probability of each of the sample files is as follows:

-12111.3 ../../speech/sample1
-7388.84 ../../speech/sample2
-7468.29 ../../speech/sample3

Word count for each sample file (found with wc -w)
sample1: 1686
sample2: 978
sample3: 985

Calculating the perplexity per word for each of the sample files:
$2^{-\frac{1}{N} * log_2 p(x)}$
sample1: $2^{\frac{-1}{1686} * (-12111.3)} = 2^{7.18345} = 145.3565$
sample2: $2^{\frac{-1}{978} * (-7388.84)} = 2^{7.55505} = 188.0602$
sample3: $2^{\frac{-1}{985} * (-7468.29)} = 2^{7.58202} = 191.6088$

log2 probabilities on the bigger switchboard corpus...
-12561.5 ../../speech/sample1
-7538.27 ../../speech/sample2
-7938.95 ../../speech/sample3

A larger corpus results in a more positive value of the log2 probability. This shows that the larger corpus results in a better fitting model as it has a higher probability of predicting the future. Furthermore, this results in the perplexities per word decreases. This is due to the larger training corpus gives more training data . The perplexity going down means that the model is less confused on the sample text, and is able to choose from a smaller number of possibilities for each word.

# 2 textcat.py

textcat.py included in submission.

# 3 Categorizing

## 3.1

Lowest error is 0.09259259..., or an accuracy of 0.90740740...

## 3.2

The value of 0.00035 was used.

## 3.3

An error rate of .13148... was observed when our value of lambda was used on the test data.

## 3.4 Graphs

Graphs are included in the submission.

# 4 Questions

V the size of the vocabulary including *OOV*

## 4.1

UNIFORM estimate $\hat{p}(z|xy) = 1/V$

ADDL estimate $\hat{p}(z|xy) = \frac{c(xyz)+\lambda}{c(xy)+\lambda V}$

If we mistakenly take V to equal 19,999, then the UNIFORM estimate would be larger than it is suppose to be, since V is the denominator. Having a lower-than-expected denominator would give more weight to smaller, more novel events.

For the ADDL estimate, having V equal 19,999 would result in a similar situation where the estimation would be larger than expected because V is in the denominator and give more weight to novel events. However, we would see less of an impact with ADDL estimation depending on what $\lambda$ is.

For both, by having V equal a smaller number than it is supposed to be means that the sum of all probabilities would not equal 1.

### 4.2

Setting $\lambda = 0$ would give the naive historical estimate. This would mean that no smoothing is occurring at all.

Beyond that, if it happens that $c(xy) = 0$, (in other words, we didn't see $xy$ in training) then we get that $\hat{p}(z|xy)$ has no value at all / is undefined.

### 4.3

If c(xyz) = c(xyz') = 0, then:

$\hat{p}(z|xy) = \frac{\lambda V \hat{p}(z|y)}{c(xy)+\lambda V}$

$\hat{p}(z'|xy) = \frac{\lambda V \hat{p}(z'|y)}{c(xy)+\lambda V}$

If c(xyz) = c(xyz') = 1, then:

$\hat{p}(z|xy) = \frac{1+\lambda V \hat{p}(z|y)}{c(xy)+\lambda V}$

$\hat{p}(z'|xy) = \frac{1+\lambda V \hat{p}(z'|y)}{c(xy)+\lambda V}$

### 4.4

BACKOFF ADDL estimate $\hat{p}(z|xy) = \frac{c(xyz)+\lambda V * \hat{p}(z|y)}{c(xy)+\lambda V}$

Increasing lambda will make it so that the trigram probabilities will be more like the corresponding bigram's probability because we're putting less weight on the trigram's count.

# 5 Other Smoothing

## 5.1 add-l smoothing

Implemented

### 5.2 ADDL vs BACKOFF_ADDL

With the same lambda value as in 3.c (0.00035), switching to BACKOFF_ADDL improved the performance. This resulted in an error rate of 0.09444, smaller than our error rate of 0.13148 from 3.c.

Basically running fileprob.py as in Q1, but using backoff_add0.01 as the smoothing method.

-9898.16 ../../speech/sample1 -6048.05 ../../speech/sample2 -6105.56 ../../speech/sample3

Calculating the perplexity per word for each of the sample files:

sample1: $2^{\frac{-1}{1686}*(-9898.16)} = 2^{5.87079} = 58.5174$

sample2: $2^{\frac{-1}{978}*(-6048.05)} = 2^{6.1841} = 72.7109$

sample3: $2^{\frac{-1}{985}*(-6105.56)} = 2^{6.1985} = 73.4422$

# 6 The Long Question

# 7 a-priori

When adding the prior probability that P(spam) = 1/3, this obviously means that P(gen) = 2/3. Text categorizing is looking at the probability of the model (whether an email is *gen* or *spam*) given the data (a test email text file), and looking for the data that has the greatest resulting probability.

Using Bayes Theorem, this means that determining the maximum probability out of all of the models given the data =

$\frac{P(data|model)*P(model)}{P(data)}$

= P(data — model) * P(model)

Since P(data) doesn't help us to find the maximum probability. Rather, we use the a-priori that was given to us.

The number is used only at test time.

# 8 Speech Recognition

## 8.1

We are trying to maximize the probability of the utterance $U$ given the intentional statement someone was trying to say $\overrightarrow{w}$. In other words, we want to maximize:

$P(U|\overrightarrow{w})$

| Smoothing | errorRate [Easy] | errorRate [unrestricted] |
|---|---|---|
| Uniform | 0.202 | 0.360 |
| add10 | 0.187 | 0.362 |
| add1 | 0.183 | 0.355 |
| add0.1 | 0.166 | 0.360 |
| add0.01 | 0.160 | 0.353 |
| backoff_add10 | 0.184 | 0.358 |
| backoff_add1 | 0.168 | 0.356 |
| backoff_add0.1 | 0.156 | 0.363 |
| backoff_add0.01 | 0.157 | 0.356 |
| loglin | XXX | 000 |

Table 1: Error Results from speechrec.py, used to decide which smoothing method to pick.

$= \frac{P(\overrightarrow{w}|U)P(U)}{P(\overrightarrow{w})}$, by Bayes'.

We can get $P(\overrightarrow{w}|U)P(U)$ from the sample file, where the second column provides the value of $\log_2(P(U|\overrightarrow{w}) = P(\overrightarrow{w}|U)P(U)$.

We can get P(U) from the trigram model.

## 8.2

speechrec.py is included in the submission.

## 8.3 Error Rates

Chosen smoothing method, and why: We tested all of the smoothings that we had implemented (just on one n-gram model) and selected the one that gave us the best error rates. This experiment is described in Table 1. We used the words-10 lexicon.

*Used overleaf.com to generate LaTeX document.*