HUGH HAN

600.465 Natural Language Processing **Fall 2016**
Homework #2 **Due:** 27 September 2016, 5:00pm

1. The log probability for each sample corpus, training on the `switchboard-small` corpus, is calculated as follows.

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

To calculate the perplexity of each corpus, we can simply raise 2 to the power of each *negated* log probability. That is, we can simply do the following.

$$2^{12111.3} \quad \text{for} \quad \texttt{speech/sample1}$$
$$2^{7388.84} \quad \text{for} \quad \texttt{speech/sample2}$$
$$2^{7468.29} \quad \text{for} \quad \texttt{speech/sample3}$$

To calculate the perplexity per word, we first need to calculate the cross-entropy by dividing each negated log proability by the number of words in its respective corpus.

The number of words in each corpus is calculated as follows.

```
1686   speech/sample1
 978   speech/sample2
 985   speech/sample3
```

Then we see that the perplexity per word of each file is the following.

$$2^{\frac{12111.3}{1686}} \quad \text{for} \quad \texttt{speech/sample1}$$
$$2^{\frac{7388.84}{978}} \quad \text{for} \quad \texttt{speech/sample2}$$
$$2^{\frac{7468.29}{985}} \quad \text{for} \quad \texttt{speech/sample3}$$

We obtain the following approximated perplexity-per-word values.

$$145.36 \quad \text{for} \quad \texttt{speech/sample1}$$
$$188.06 \quad \text{for} \quad \texttt{speech/sample2}$$
$$191.61 \quad \text{for} \quad \texttt{speech/sample3}$$

Training on the larger `switchboard` corpus, we get the following log probabilities for the following sample corpuses, respectively:

```
-12561.5   speech/sample1
-7538.27   speech/sample2
-7938.95   speech/sample3
```

Note that each of the log-probabilities grew more negative for its respective sample corpus. Thus, the each perplexity must also be smaller for its respective sample corpus. The reason for this is because if we increase the size of our training data, then we introduce a greater number of possible sentences. That is, the probability that the sentences in our specific test files appear would be much smaller, in that there is a greater variety of possibilities to choose from.

2.

3. I chose to do spam detection.

   When classifying the sample files containing genuine messages, I received the following output:

```
178 looked more like gen_spam/train/gen (98.89%)
2 looked more like gen_spam/train/spam (1.11%)
```

   When classifying the sample files containing spam messages, I received the following output:

```
66 looked more like gen_spam/train/gen (73.33%)
24 looked more like gen_spam/train/spam (26.67%)
```

   That is, we had the following error rates for the different data sets.

$$\begin{aligned} \texttt{gen} &: \text{error-rate} = 1.11\% \\ \texttt{spam} &: \text{error-rate} = 73.33\% \end{aligned}$$

   (a) Each classification decision is based on probability. That is, whichever training corpus yields a higher cross-entropy with each test file is the training corpus to which that test file will be classified.

   However, we cannot necessarily deduce that the lowest cross-entropy occurs with the lowest error rate. With a low error rate, it could just be that the opposite incorrect corpus has a higher cross-entropy than the correct corpus, instead of the correct corpus having the lowest possible cross-entropy.

   Thus, we are forced to do some type of guess-and-check to find the value of the lowest cross-entropy. We proceeded by doing a human version of binary search.

   Using our binary search, we find the following two values of the approximate smallest log-probabilities.

$$\begin{aligned} \texttt{gen} &: \text{log-probability} \approx -423681 \\ \texttt{spam} &: \text{log-probability} \approx -280362 \end{aligned}$$

   Using these values, we can obtain the approximate smallest cross-entropies.

$$\begin{aligned} \texttt{gen} &: \text{log-probability} \approx -423681 \\ \texttt{spam} &: \text{log-probability} \approx -280362 \end{aligned}$$

The λ values that were used in finding these minimum cross-entropies were as follows.

$$\begin{aligned} \texttt{gen} &: \lambda = 0.01357 \\ \texttt{spam} &: \lambda = 0.0083 \end{aligned}$$

(b) We can do binary search again, but let's think of how we can be a little smarter this time. We already have

$$\begin{aligned} \lambda_{\texttt{gen}} &= 0.01357 \\ \lambda_{\texttt{spam}} &= 0.0083 \end{aligned}$$

so we can use them as critical points. We see that:

(i) all $\lambda < \lambda_{\texttt{spam}}$ cause increasing log-probabilities

(ii) all $\lambda > \lambda_{\texttt{gen}}$ cause increasing log-probabilities

Then the following must be true.

$$\lambda_{\texttt{spam}} \leq \lambda^* \leq \lambda_{\texttt{gen}}$$

So now we can do a smarter guess-and-check binary search, and we find the following value of λ^*.

$$\lambda^* \approx 0.0104$$

(c) First, let's try to classify the test files containing genuine messages.

```
347 looked more like gen_spam/train/gen (96.39%)
13 looked more like gen_spam/train/spam (3.61%)
```

Next, let's try to classify the test files containing spam messages.

```
50 looked more like gen_spam/train/gen (27.78%)
130 looked more like gen_spam/train/spam (72.22%)
```

So then we get the following error rates.

$$\begin{aligned} \texttt{gen} &: \text{error-rate} = 3.61\% \\ \texttt{spam} &: \text{error-rate} = 27.78\% \end{aligned}$$

(d)

(e)

4. (a) Let's say that we take $V = 19{,}999$.

i. Using the UNIFORM estimate, if we see a word that is out of the vocabulary, we would have no choice but to assign that OOV word a probability of 0, due to it being the 20,000th word of a vocabulary of size 19,999. That is, the sum of the

probabilities of all other words already sum to 1, and we cannot have a probability greater than 1.

$$\sum_{i=1}^{V} \frac{1}{V} = 1$$

However, this is an bad estimation, simply because no word can have a probability of 0, regardless of whether that word was observed in the training data.

ii. Using the ADDL estimate,

(b) If we let $\lambda = 0$, then we are not modifying our probability at all. That is, we get the following from our probabiliy estimate.

$$\begin{aligned} \hat{p}(z \mid xy) &= \frac{c(xyz) + \lambda}{c(xy) + \lambda V} \\ &= \frac{c(xyz) + 0}{c(xy) + 0} \\ &= \frac{c(xyz)}{c(xy)} \end{aligned}$$

(c)

(d)

5. (a) `IMPLEMENTATION PROBLEM`

(b) **cross-entropies for the switchboard corpora**

text categorization error rates for gen/spam

Using $\lambda^* = 0.014$, as calculated in problem **3**(c), we receive the following classifications for `gen` and `spam` test files, respectively.

```
209 looked more like gen_spam/train/gen (58.06%)
151 looked more like gen_spam/train/spam (41.94%)
```

```
7 looked more like gen_spam/train/gen (3.89%)
173 looked more like gen_spam/train/spam (96.11%)
```

From these outputs, we can see that using our calculated λ^*value, switching from ADDL to BACKOFF_ADDL causes the error rates for classifying spam files to decrease extraordinarily, but at the cost of spiking up error rates for classifying genuine files.

(c) *Extra credit.*

6. (a)

(b)

(c)

(d)

(e) *Extra credit.*

(f)

(g) i.

ii.

iii.

iv.

v.

vi.

vii.

7.

8. (a)

(b)

(c)

9. *Extra credit.*

10. *Extra credit.*

(a)

(b)

(c)

(d)

(e) i.

ii.

iii.

(f)

(g)

11. *Extra credit.*