CS190I HW1

Matthew Ho

January 2022

1 Collaboration

Did you receive any help whatsover from anyone in solving this assignment?
Yes, I responded to piazza question @17 thinking the issue was that Sean didn't include the '#' token in the training set. Alex Mei reached out to me telling me that the actual issue was that there was was no 'B #' bigram as Sean had written in his question.
Did you give any help whatsoever to anyone in solving this assignment? **No**

2 Top-100 Frequent Bigrams

1. Concatenate files and convert uppercase to lowercase

```
$ cat 001.txt 002.txt 003.txt | tr '[:upper:]' '[:lower:]' > combined.txt
```

No console output; first five lines of combined.txt:

```
claxton hunting first major medal

british hurdler sarah claxton is confident she can win her first major medal at next month's european indoor championships in madrid.

the 25-year-old has already smashed the british record over 60m hurdles twice this season, setting a new mark of 7.96 seconds to win the aaas title. "i am quite confident," said claxton. "but i take each race as it comes. "as long as i keep up my training but not do too much i think there is a chance of a medal." claxton has won the national 60m hurdles title for the past three years but has struggled to translate her domestic success to the international stage. now, the scotland-born athlete owns the equal fifth-fastest time in the world this year. and at last week's birmingham grand prix, claxton left european medal favourite russian irina shevchenko trailing in sixth spot.
```

2. Tokenize by converting all non-alphabetical characters into a delimiter (underscore in this case). I used the tr command with options s for squeezing repeat non-alpha chars and c for getting the complement of alphabetical chars.

```
$ tr -sc 'a-z' '_' < combined.txt > tokenized.txt
```

No console output; beginning of line 1 of tokenized.txt (newlines were changed to _ so the whole text is on one line)

```
claxton_hunting_first_major_medal_british_hurdler_sarah_claxton_is_confident_she_can_win_her_first_major_medal_at_next_month_s_european_indoor_championships_in_madrid_the_year_old_has_already_smashed_the_british_record_over_m_hurdles_twice_this_season_setting_a_new_mark_of_seconds_to_win_the_aaas_title_i_am_quite_confident_said_claxton_but_i_take_each_race_as_it_comes_...
```

3. Double each word. Using sed command to replace each (word)_ with (word)_(word)_.

```
$ sed 's/\([a-z]*\)_/\1_\1_/g' < tokenized.txt > doubled.txt
```

No console output; beginning of line 1 of doubled.txt:

```
claxton_claxton_hunting_hunting_first_first_major_major_medal_medal_british_british_hurdler_hurdler_
 sarah_sarah_claxton_claxton_is_is_confident_confident_she_she_can_can_win_win_
her_her_first_first_major_major_medal_medal_at_at_next_next_
month_month_s_s_european_european_indoor_indoor_championships_championships_in_in_
 madrid_madrid_the_the_year_year_old_old_has_has_already_already_smashed_smashed_
...
```

4. De-duplicate first and last words with sed. The middle capture group matches everything but the first and last words because Kleene star is greedy.

```
$ sed 's/[a-z]*_\(.*\)_[a-z]*_/\1_/' < doubled.txt > middle.txt
```

No console output; beginning of line 1 of middle.txt:

```
claxton_hunting_hunting_first_first_major_major_medal_medal_british_british_hurdler_hurdler_
 sarah_sarah_claxton_claxton_is_is_confident_confident_she_she_can_can_win_win_
her_her_first_first_major_major_medal_medal_at_at_next_next_
month_month_s_s_european_european_indoor_indoor_championships_championships_in_in_
 madrid_madrid_the_the_year_year_old_old_has_has_already_already_smashed_smashed_
...
```

5. Move each bigram to a separate line using sed to replace each (word1)_(word2)_ with word1,word2\n.

```
$ sed 's/\([a-z]*\)_\([a-z]*\)_/\1,\2\n/g' < middle.txt > bigrams.txt
```

No console output; first 5 lines of bigrams.txt:

```
claxton,hunting
hunting,first
first,major
major,medal
medal,british
```

6. Sort to gather duplicates, count duplicates (uniq with c flag), and sort by count (n flag for numeric sort; r flag puts the most frequent at the top).

```
$ sort bigrams.txt | uniq -c | sort -nr > counts.txt
```

First 5 lines of counts.txt:

```
 9 in,the
 6 for,the
 5 o,sullivan
 5 i,was
 4 the,race
```

7. Remove counts, display only top 100 bigrams. Remove counts by using awk to print second column, and head command to only print the first 100 lines.

```
$ awk '{print $2}' counts.txt | head -100
```

First 5 lines of output:

```
in,the
for,the
o,sullivan
i,was
the,race
```

3 N-gram: the longer the better?

We cannot use 10-gram or even longer n-grams to effectively model language because there are too many possible sequences of 10 words. As such, any practical corpus is not long enough to contain the all 10-grams, much less approximately represent the relative frequency of their true distribution.

4 Language Modeling (LM)

Corpus: AAABANBABBBNNANBNN

4.1 Unigram Probabilities

Counts: A: 6, B: 6, N: 6, Total = 18
Unigram probabilities:
A: 6/18 = 1/3
B: 6/18 = 1/3
N: 6/18 = 1/3

4.2 Bigram Probabilities

Counts:
{(A, A): 2, (A, B): 2, (B, A): 2, (A, N): 2, (N, B): 2,
(B, B): 2, (B, N): 2, (N, N): 2, (N, A): 1, (N, #): 1,}
Bigrams probabilities: Using MLE, $P(w_2 \mid w_1) = \text{count}(w_1 w_2)/\text{count}(w_1)$
$P(A \mid A) = \text{count(AA)}/\text{count(A)} = 2/6 = 1/3$
$P(B \mid A) = \text{count(AB)}/\text{count(A)} = 2/6 = 1/3$
$P(A \mid B) = \text{count(BA)}/\text{count(B)} = 2/6 = 1/3$
$P(N \mid A) = \text{count(AN)}/\text{count(A)} = 2/6 = 1/3$
$P(B \mid N) = \text{count(NB)}/\text{count(N)} = 2/6 = 1/3$
$P(B \mid B) = \text{count(BB)}/\text{count(B)} = 2/6 = 1/3$
$P(N \mid B) = \text{count(BN)}/\text{count(B)} = 2/6 = 1/3$
$P(N \mid N) = \text{count(NN)}/\text{count(N)} = 2/6 = 1/3$
$P(A \mid N) = \text{count(NA)}/\text{count(N)} = 1/6$
$P(\# \mid N) = \text{count(N\#)}/\text{count(N)} = 1/6$

4.3 Testing and Perplexity

Test data: ABANABB

1. Find perplexity of the unigram LM

$$PP(W) = P(w_1 w_2 ... w_N)^{-1/N}$$

For unigram model...

$$P(w_1 w_2 ... w_N) = P(w_1)P(w_2)...P(w_N)$$

Our trained model assigns 1/3 to each word

$$P(w_1 w_2 ... w_N) = \frac{1}{3} \cdot \frac{1}{3} \cdot ... \cdot \frac{1}{3} = \frac{1}{3}^N = \frac{1}{3}^7$$
$$PP(W) = P(w_1 w_2 ... w_N)^{-1/N} = \left(\frac{1}{3}^7\right)^{-1/7} = \frac{1}{3}^{-1} = 3$$

2. Find perplexity of the bigram LM

$$PP(W) = P(w_1 w_2 ... w_N)^{-1/N}$$

For bigram model...

$$\begin{aligned}
P(w_1 w_2 ... w_N) &= \prod_{i=2}^{N} P(w_i \mid w_{i-1}) \\
&= P(B \mid A)P(A \mid B)P(N \mid A)P(A \mid N)P(B \mid A)P(B \mid B)P(\# \mid B) \\
&= \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{6} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{0}{6} = 0 \\
PP(W) &= P(w_1 w_2 ... w_N)^{-1/N} = 0^{-1/8} = \text{undefined}
\end{aligned}$$

4.4 Add-one Smoothing

Report perplexities after using add-one smoothing in training
Add 1 Smoothing: $P(w_2 \mid w_1) = \frac{\text{count}(w_1 w_2)+1}{\text{count}(w_1)+V}$

Unigrams

$$P(w_1 w_2 ... w_N) = \left(\frac{6+1}{18+3}\right)^7 = \frac{1}{3}^7$$

$$PP(W) = P(w_1 w_2 ... w_N)^{-1/N} = \left(\frac{1}{3}^7\right)^{-1/7} = \frac{1}{3}^{-1} = 3$$

Bigrams

$$\begin{aligned}
P(w_1 w_2 ... w_N) &= P(B \mid A)P(A \mid B)P(N \mid A)P(A \mid N)P(B \mid A)P(B \mid B)P(\# \mid B) \\
&= \frac{2+1}{6+4} \cdot \frac{2+1}{6+4} \cdot \frac{2+1}{6+4} \cdot \frac{1+1}{6+4} \cdot \frac{2+1}{6+4} \cdot \frac{2+1}{6+4} \cdot \frac{0+1}{6+4} \\
&= \frac{3}{10} \cdot \frac{3}{10} \cdot \frac{3}{10} \cdot \frac{2}{10} \cdot \frac{3}{10} \cdot \frac{3}{10} \cdot \frac{1}{10} = \frac{(1)(2)(3)^5}{(10)^7} \\
PP(W) &= P(w_1 w_2 ... w_N)^{-1/N} = \frac{(2)(3)^5}{(10)^7}^{-1/8} = 3.46
\end{aligned}$$

5 Linear Interpolation

If we tuned the lambda hyperparameters on the entire training set, then the model may be overfitted for the training data. The model should ideally be able to generalize well to unseen data, but this setup optimizes for this specific subset. A better method would involve partitioning the training set into a training and held-out validation set. The validation set can then be used to tune hyperparameters to help the model's generalizability.

6 Out-of-Vocabulary Words

Using a standard bigram LM without smoothing underestimates the probabilities for OOV words, since it assigns zero probability to bigrams that actually do occur, just not in the training set. One way to deal with OOV words is to train with an unknown word token. For this method, we would need to decide on a fixed vocabulary ahead of time that excludes some of the less common words in the training set. Then we would replace OOV words in the training and test sets in the pre-processing stage. This way, the LM trains with and therefore has some knowledge about the probabilities of bigrams involving the unknown token, potentially performing better than blindly assigning some low probability to the unknown token.