CSE 517: Homework 1 Language Models

Lilly Kumari

February 2, 2019

Problem 1. Smoothing

Solution For a tri-gram language model, we want the probability mass across a given context (previous bigrams) to sum up to one i.e. $\sum_{n=1}^{|V|} P(w_n|w_i, w_j) = 1$ Back-off helps to take into account the context distribution (bi-gram distribution) and further if the n-gram (here tri-gram) does not exist in the data i.e. $count(w_{i-2}, w_{i-1}, w_i) = 0$) But while backing off recursively to a lower order model, we add extra probability, hence the total probability mass given a $(n-1)-gram$ context would exceed 1. Thus, those extra probabilities have to get discounted.

The given model could be re-written as:

$$p(w_i|w_{i-2}, w_{i-1}) = p_1(w_i|w_{i-2}, w_{i-1}) + \alpha * p_2(w_i|w_{i-1}) + \beta * p_3(w_i)$$
$$where\, \alpha = 1,\, if\, count(w_{i-2}, w_{i-1}, w_i) = 0,\, else\, 0$$
$$\beta = 1,\, if\, count(w_{i-1}, w_i) = 0,\, else\, 0$$

The given model doesn't have a valid probability distribution since given the back-off probabilities in case of non-existence of some tri-grams will make the probability mass go beyond 1. In order to make it a valid distribution, we discount the ML estimate for the trigrams to save probability mass for bigrams & unigrams.

The modified $p^*_{ML}(w_i|w_{i-2}, w_{i-1})$ uses the discounted tri-gram count function described in Eq 2.

$$p_1(w_i|w_{i-2}, w_{i-1}) = p^*_{ML}(w_i|w_{i-2}, w_{i-1}) = \frac{c^*(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})} \tag{1}$$

$$c^*(w_{i-2}, w_{i-1}, w) = c(w_{i-2}, w_{i-1}, w) - d_1 \quad (for \quad known \quad trigrams) \tag{2}$$

Discounted probability mass from observed tri-grams =

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \epsilon A(w_{i-2}, w_{i-1})} \frac{c^*(w_{i-2}, w_{i-1}, w)}{c(w_{i-2}, w_{i-1})} \tag{3}$$

Similarly, in order to save probability mass for unigrams, the count of bi-grams for observed bigrams is discounted as follows:

$$c^*(w_{i-1}, w) = c(w_{i-1}, w) - d_2 \quad (for \quad known \quad bigrams) \tag{4}$$

Thus the modified $p^*_{ML}(w_i|w_{i-1})$ uses the discounted bi-gram count function described in Eq 4 as follows:

$$p^*_{ML}(w_i|w_{i-1}) = \frac{c^*(w_{i-1}, w_i)}{c(w_{i-1})} \tag{5}$$

$$p_2(w_i|w_{i-1}, w_{i-2}) = \alpha(w_{i-2}, w_{i-1}) * \frac{p^*_{ML}(w_i|w_{i-1})}{\sum_{w \epsilon B(w_{i-2}, w_{i-1})} p^*_{ML}(w|w_{i-1})} \tag{6}$$

Now, discounted probability mass from observed bi-grams =

$$\alpha(w_{i-1}) = 1 - \sum_{w \epsilon A(w_{i-1})} \frac{c^*(w_{i-1}, w)}{c(w_{i-1})} \tag{7}$$

$$p_3(w_i|w_{i-1}, w_{i-2}) = \alpha(w_{i-1}) * \frac{p_{ML}(w_i)}{\sum_{w \epsilon B(w_{i-1})} p_{ML}(w)} \tag{8}$$

Since, as we go in a top-down fashion towards lower order model, we keep on discounting some probability mass for them (d_1 and d_2, chosen at trigram & bigram level respectively) and at each step, we normalize the distribution. Hence, the resultant model has a valid probability distribution.

Problem 4. Language Model Classifier

Solution Let's say we have overall *K* different categories (labels) of a set of classes *C* and the input document *x* is a sequence of tokens $\{x_1, x_2, ..., x_n\}$

The most likely class can be calculated as follows:

$$\begin{aligned} c^* &= \underset{c \epsilon C}{\operatorname{argmax}} P(c \mid x) \\ &= \underset{c \epsilon C}{\operatorname{argmax}} \frac{P(x \mid c)P(c)}{P(d)} \quad (using\ Bayes\ Rule) \\ &= \underset{c \epsilon C}{\operatorname{argmax}} P(x \mid c)P(c) \\ &= \underset{c \epsilon C}{\operatorname{argmax}} P(x_1, x_2, ...x_N \mid c)P(c) \end{aligned}$$

Since for estimating $P(x_1, x_2, ...x_n \mid c)$, we would be needing a very large number of training examples to learn a document size distribution. So, we can assume that the position of the words doesn't matter and the probabilities $P(x_i \mid c_j)$ are independent given the class.

$$\begin{aligned} c^* &= \underset{c \epsilon C}{\operatorname{argmax}} P(x_1, x_2, ...x_n \mid c)P(c) \\ &= \underset{c_j \epsilon C}{\operatorname{argmax}} P(c_j) \prod_{i=1}^{N} P(x_i \mid c_j) \end{aligned}$$

The $P(c_j)$ represents the fraction of documents that belong to class c_j

$$P(c_j) = \frac{doc\,count(c = c_j)}{\#\,of\,documents}$$

The $P(w_i \mid c_j)$ represents the fraction of times word w_i appears among all words in the documents belonging to the class c_j (V is the overall vocabulary across the set of documents)

$$P(w_i \mid c_j) = \frac{c(w_i, c_j)}{\sum_{w \epsilon V} c(w, c_j)}$$

For estimating $P(w_i \mid c_j)$, we can make a mega document by concatenating all documents belonging to the class c_j. Also, we can apply add-k smoothing for determining it as follows:

$$P(w_i \mid c_j) = \frac{c(w_i, c_j) + k}{(\sum_{w \epsilon V} c(w, c_j)) + k * V}$$

Problem 2. Language Models

Solution Design of the Implementation Pipeline and Results:

- Used both START (START1, START2) and STOP boundary tokens, so $V^* = V \cup \{START1, START2, STOP\}$. Changes done after reading the text files only.
- Created vocabulary from the training set and then, used it to determine the UNK cutoff threshold by evaluating the overall corpus coverage against both the training and dev sets. The plots in Figure 1 and 2 show the coverage against different values of cutoff (A value of 3 means that all words occurring less than or equal to 3 times will be replaced with UNK token.) Since at a cutoff of 3, we have around 95% coverage of the dev set and 96% coverage of training set, we set the cutoff to 3.
- After replacing the rare (≤ 3) words with UNK, the vocabulary is updated against the training dataset.
- In the dev and test set, the OOV words (not present in the vocabulary made against training data) are replaced with the UNK token.
- The probabilites are calculated in the log (base 2) space.
- Since there's no smoothing technique used in the bigram and trigram models, we get perplexity = infinity for bigram & trigram model on dev and test data as there are many unobserved bi-grams and tri-grams. For example, ('raymond', 'understand'), ('house', 'the', 'passed').
- Table 1 shows the perplexity scores of the 3 language models on training, dev and test data sets.

Problem 3. Smoothing - Only Trigram Language Models

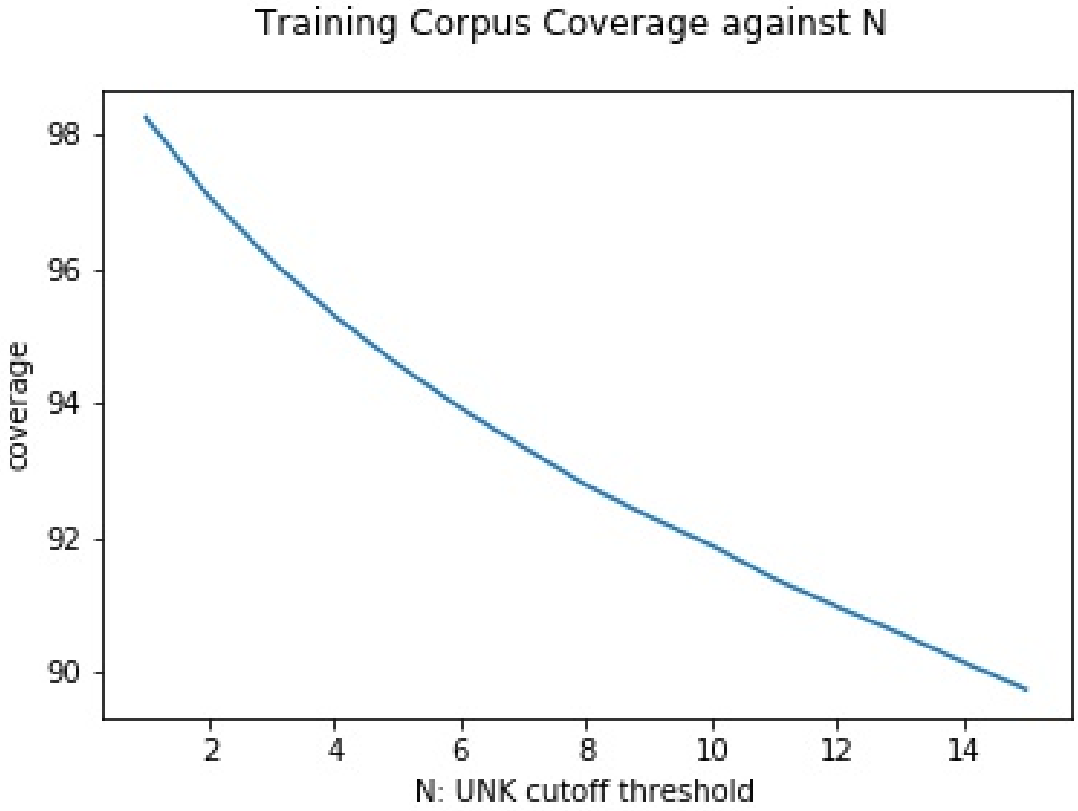

Figure 1:

Table 1: Perplexity scores

LM	Training	Dev	Test
Unigram	491.8704	459.5108	464.4932
Bigram	47.3828	inf	inf
Trigram	6.9278	inf	inf

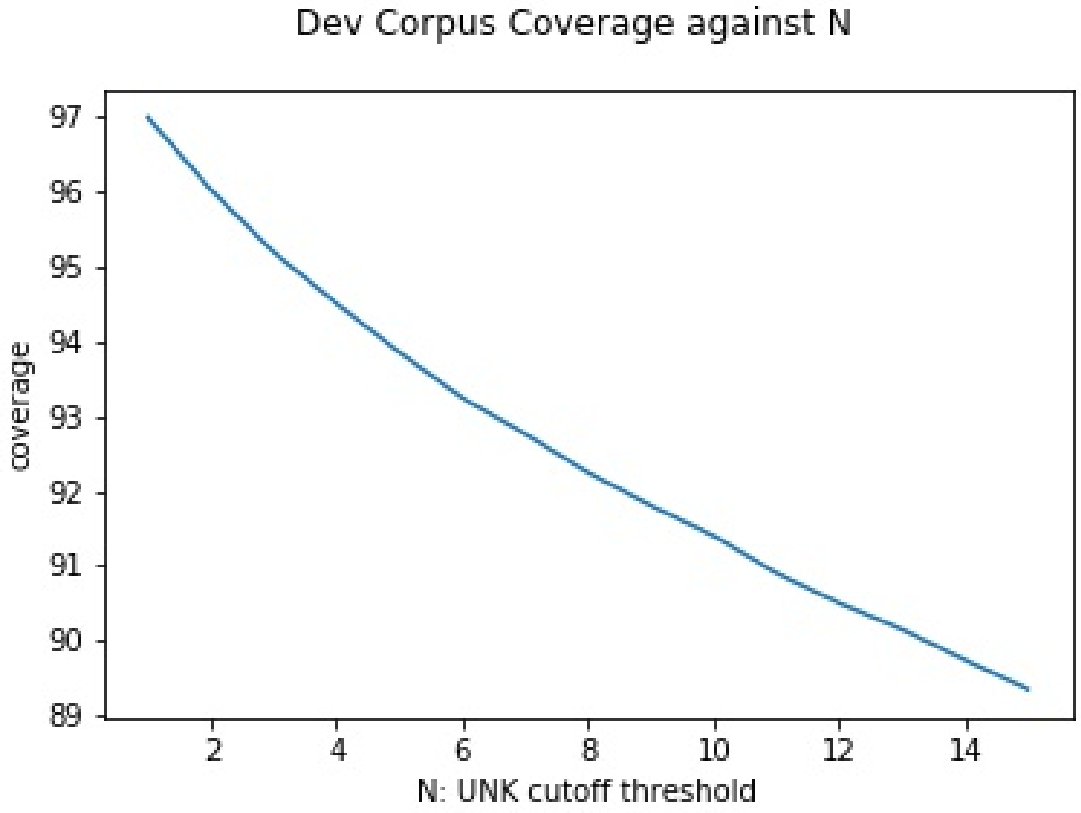

Figure 2:

```
at K = 20 trigram LM perplexity: Training = 4336.981633172256 Dev = 4449.445511238823
at K = 10 trigram LM perplexity: Training = 3787.345116457116 Dev = 3998.677678829385
at K = 1 trigram LM perplexity: Training = 1623.8177966338667 Dev = 2435.217876547526
at K = 0.75 trigram LM perplexity: Training = 1382.7509163895927 Dev = 2260.6309484443714
at K = 0.5 trigram LM perplexity: Training = 1081.2087818329242 Dev = 2031.1072792899952
at K = 0.25 trigram LM perplexity: Training = 680.8987301632332 Dev = 1689.4273977386897
at K = 0.1 trigram LM perplexity: Training = 353.1446988337505 Dev = 1339.2183942588135
at K = 0.05 trigram LM perplexity: Training = 213.4262855893201 Dev = 1144.0046468446683
at K = 0.01 trigram LM perplexity: Training = 70.25254297009593 Dev = 870.1702387007215
at K = 0.001 trigram LM perplexity: Training = 19.36673086136386 Dev = 798.3337686955196
at K = 0.0001 trigram LM perplexity: Training = 9.093358393415667 Dev = 1126.9141340166778
at K = 1e-05 trigram LM perplexity: Training = 7.194151119342178 Dev = 2208.5083318901375
at K = 1e-06 trigram LM perplexity: Training = 6.955286532484236 Dev = 4813.314118160038
at K = 1e-08 trigram LM perplexity: Training = 6.92808087726366 Dev = 23604.40807965636
```

Figure 3: Add-K Smoothing- perplexity scores

```
at x1 = 0.100000, x2 = 0.100000, x3 = 0.800000, Trigram LM perplexity: Training = 33.197002, Dev =179.911308
at x1 = 0.200000, x2 = 0.100000, x3 = 0.700000, Trigram LM perplexity: Training = 21.794936, Dev =168.265189
at x1 = 0.300000, x2 = 0.100000, x3 = 0.600000, Trigram LM perplexity: Training = 16.591781, Dev =163.989104
at x1 = 0.300000, x2 = 0.200000, x3 = 0.500000, Trigram LM perplexity: Training = 15.715659, Dev =148.273939
at x1 = 0.250000, x2 = 0.250000, x3 = 0.500000, Trigram LM perplexity: Training = 17.211762, Dev =143.690876
at x1 = 0.300000, x2 = 0.200000, x3 = 0.500000, Trigram LM perplexity: Training = 15.715659, Dev =148.273939
at x1 = 0.300000, x2 = 0.250000, x3 = 0.450000, Trigram LM perplexity: Training = 15.333706, Dev =143.437207
at x1 = 0.300000, x2 = 0.300000, x3 = 0.400000, Trigram LM perplexity: Training = 14.981830, Dev =139.924257
at x1 = 0.350000, x2 = 0.350000, x3 = 0.300000, Trigram LM perplexity: Training = 13.301870, Dev =139.284559
at x1 = 0.400000, x2 = 0.300000, x3 = 0.300000, Trigram LM perplexity: Training = 12.430723, Dev =143.253759
at x1 = 0.450000, x2 = 0.350000, x3 = 0.200000, Trigram LM perplexity: Training = 11.288246, Dev =146.791379
at x1 = 0.500000, x2 = 0.350000, x3 = 0.150000, Trigram LM perplexity: Training = 10.515533, Dev =153.729161
at x1 = 0.600000, x2 = 0.300000, x3 = 0.100000, Trigram LM perplexity: Training = 9.408671, Dev =171.465657
at x1 = 0.700000, x2 = 0.200000, x3 = 0.100000, Trigram LM perplexity: Training = 8.657923, Dev =190.445439
at x1 = 0.800000, x2 = 0.150000, x3 = 0.050000, Trigram LM perplexity: Training = 7.939376, Dev =237.446596
at x1 = 0.850000, x2 = 0.100000, x3 = 0.050000, Trigram LM perplexity: Training = 7.677661, Dev =266.267470
at x1 = 0.900000, x2 = 0.050000, x3 = 0.050000, Trigram LM perplexity: Training = 7.437604, Dev =320.642928
```

Figure 4: Linear Interpolation - perplexity scores

Solution 3.1.a Figure 3 shows the perplexity scores of trigram model using add-k smoothing for different values of *k*

3.1.b Figure 4 shows the perplexity scores for different values of $\lambda 1$, $\lambda 2$ and $\lambda 3$ (in the figure, x is λ)

3.1.c Hyperparameters:

- Add-K smoothing: K = 0.001 since the dev set perplexity is lowest for that value. The test set perplexity score using that k is 799.0265.
- Linear Interpolation: $\lambda 1 = 0.35, \lambda 2 = 0.35, \lambda 3 = 0.3$ (dev set perplexity score lowest for these values). Test set perplexity score = 140.6197

3.2 If the training data is reduced to half, the vocabulary size would get reduced undoubtedly. But, the occurrences of many bi-grams and tri-grams would become less too. It's highly likely that a good number of such n-grams would belong to unobserved category in the dev and test data. Thus, the perplexity of a model trained on half of the training dataset would increase on previously unseen dataset.

3.3 If the cutoff threshold for UNK is high (here 5), then a lot of words from the vocabulary will get replaced by UNK, thus reducing the vocabulary size. A good number of bi-grams from the previously unseen data could be already observed (tailing or starting with UNK). Same could be possible for tri-grams (but in lower magnitude). While when the UNK cutoff threshold is small (for example 1), the vocabulary coverage will not be reduced much and the model will have many different choices for prediction (next word) i.e. more flexibility in choices. So, the perplexity in case of choosing a higher UNK cutoff (5) would decrease as compared to the perplexity score corresponding to choosing a lower UNK cutoff (1).