Assignment 1

Manasa Bollavaram

March 15, 2017

1 [Written] Smoothing (Back-off)

In a language model, if the probabilities/parameters of the model are computed by maximumlikelihood there are some disadvantages. For example, the trigrams which are never seen in the training data have their probability assigned zero and this is not at all useful for preplexity calculations because perplexity goes to infinity. Practically we expect to see trigrams previously unseen in the training data.

To handle such cases, various smoothing techniques have been developed such as laplace smoothing, good-turing smoothing, katz back-off model etc.

The probability distribution given in the problem is developed to handle such cases where trigrams are unseen in the training data and instead of assigning zero probability, they are given p_2 if bigrams are seen in the training data and p_1 if bigrams are also not seen in the training data. Assigning some value instead of zero is a better way to handle the issues of zero as probability. But, fundamentally for a distribution to classify as a probability distribution,the sum of all probabilities should add to one. The probability distribution given in the problem set adds extra probabilities to the trigrams never seen, keeping the probabilities of trigrams seen the same and hence making the sum of all the probabilities greater than one and thus not making this a valid probability distribution.

We can make a valid probability distribution by using smoothing techniques where discounting is done and the discounted mass is distributed among the unseen trigrams.

We can modify the counts and discount them by k.

$$c^*(w_{i-2}, w_{i-1}, w_i) = c(w_{i-2}, w_{i-1}, w_i) - k \tag{1}$$

$$c^*(w_{i-1}, w_i) = c(w_{i-1}, w_i) - k \tag{2}$$

We then get the probability mass which can be distributed among the trigrams never seen in the training data.

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

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{c^*(w_{i-1}, w_i}{c(w_{i-1})}) \tag{4}$$

The probability distribution can be modified as following

$$p_1(w_i|w_{i-1}, w_{i-2}) = \frac{c^*(w_{i-2}, w_{i-1}, w_i}{c(w_{i-2}, w_{i-1})}) \tag{5}$$

$$p_2(w_i|w_{i-1}, w_{i-2}) = \alpha(w_{i-2}, w_{i-1})\frac{num}{den} \tag{6}$$

where num and den are as following

$$num = \frac{c^*(w_{i-1}, w_i}{c(w_{i-1})}) \tag{7}$$

$$den = \sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} \frac{c^*(w_{i-1}, w_i}{c(w_{i-1})}) \tag{8}$$

$$p_3(w_i|w_{i-1}, w_{i-2}) = \alpha(w_{i-2}, w_{i-1})\alpha(w_{i-1})\frac{num1}{den1} \tag{9}$$

where $num1$ and $den1$ are as follows

$$num1 = \frac{p_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} p_{ML}(w_i)} \tag{10}$$

$$den1 = \sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} \frac{p_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} p_{ML}(w_i)} \tag{11}$$

2 Using language models for text categorization

The problem of categorizing text based on language model can be viewed as analogous to POS tagging. In POS tagging we label/tag a given word with its appropriate POS based on the probabilities computed. In the case of categorizing text,given a document x, we compute its probability by running he language models for all the categories available and choosing the category with the highest probability as the label to the document.

$$y = \arg\max_{1...k} p_{LM_i}(x) \tag{12}$$

where LM_i represents a language model for the ith category and $p_{LM_i(x)}$ represents the probability that a given document x belongs to the category i.

3 [Programming] Building a language model (No smoothing)

Design choices:

The sentences are tokenized based on white spaces. START and STOP symbol have been added to each sentence.

A fraction of the words with a frequency of 2 and lower are considered for UNKing. We cannot UNK all of them because it would lead to more than half of the corpus to UNK and which might reduce perplexity on unseen data but it is not the authentic perplexity. Fraction of the words tagged as UNK is varied. Following are the perplexity values with various fractions.
Perplexity scores of the unigram, bigram and trigram models implemented for *training*, *dev* and *test* sets are reported.

Trainingset
Design choices: UNK threshold frequency of 2, 50 percent of the words below the threshold frequency are tagged as UNK leading to 31,546 unique words in the vocabulary. Perplexities for such design choices are reported below.

N-Gram Perplexities without Smoothing
Perplexity 1-gram Language Model = 715.91
Perplexity 2-gram Language Model = 56.44
Perplexity 3-gram Language Model = 5.23

Design choices: UNK threshold frequency of 2, 15 percent of the words below the threshold frequency are tagged as UNK leading to 40,950 unique words in the vocabulary. Perplexities for such design choices are reported below.

N-Gram Perplexities without Smoothing
Perplexity 1-gram Language Model = 803.52
Perplexity 2-gram Language Model = 55.70
Perplexity 3-gram Language Model = 4.94

Percentage to be UNKed is selected as 15. Perplexity values are higher in the case of unigram model for 15 percent compared to 50 percent. Perplexity values of bigram and trigram models are barely affected by the percentage variation of UNK words. To avoid the artificial low perplexity on test corpus, 15 percent is selected for development and test sets instead of 50 percent.

DevelopmentSet

Without smoothing, bigram and trigram models produce invalid perplexity scores.
Unigram Perplexity without Smoothing
Perplexity 1-gram Language Model = 702.7547849725788

TestSet
As in the case of development set, without smoothing, bigram and trigram models produce invalid perplexity scores.

Unigram Perplexity without Smoothing
Perplexity 1-gram Language Model = 703.5256870123658

4 [Programming] Smoothing techniques

4.1 Add-K smoothing and linear interpolation

Add-K Smoothing

Perplexity scores of trigram models on *training*, *dev* and *test* sets for various values of K are reported below.
Training
K=1 Perplexity = 5348.27
K=0.01 Perplexity=162.47
K=0.001 Perplexity=31.41
As the perplexity reduces with the decrease in order of magnitude of K values, K=0.001 is selected for further smoothing.
Dev
K=0.001 Perplexity=2209.19
Test
K=0.001 Perplexity=2193.67

Linear Interpolation

A grid search is performed on the *dev* set to find the optimum *lambda* values for the trigram model implemented. The optimum values found are *lambda*1=0.3, *lambda*2=0.6 and *lamda*3=0.1.

Perplexity values for various *lambda* values are reported below
Training and *Dev*
*lambda*1=0.3; *lambda*2=0.6; *lambda*3=0.1; Perplexity for *training*=28.32; Perplexity for *dev* = 289.29
*lambda*1=0.1; *lambda*2=0.2; *lambda*3=0.7; Perplexity for *training*=10.84; Perplexity for *dev* = 510.67
*lambda*1=0.7; *lambda*2=0.1; *lambda*3=0.2; Perplexity for *training*=26.07; Perplexity for *dev* = 351.77
*lambda*1=0.4; *lambda*2=0.4; *lambda*3=0.2; Perplexity for *training*=21.74; Perplexity for *dev* = 292.02

Test
*lambda*1=0.3; *lambda*2=0.6; *lambda*3=0.1; Perplexity = 289.38

4.2

If only half the training data is used, UNK words in the test corpus can be high and this might artificially lower the perplexity. In a different case, using a larger training corpus assigns higher probabilities to the sentences in the test corpus if test corpus is drawn from the same distribution as training. Higher probabilities would drive down the perplexity but reducing the training data would assign lower probabilities and hence increasing the perplexity.

4.3

In the given training corpus, the words that appeared below the frequency of 5 account to more than 70 percent of the vocabulary. If all of these are UNKed then it would lead to artificially lowering the perplexity on the previously unseen data and would not be of much use. If less words are UNKed, we can extract more information from the training and assign accurate but low probabilities to the previously unseen data driving up the perplexity.