COMPGI19 Assignment 1 - Language Models

Weijie HUANG
MSc Web Science and Big Data Analytics
Department of Computer Science
UCL
London
weijie.huang@ucl.ac.uk

Abstract

In this assignment, the design and implement of the language models are used for processing the provided rap corpus.

1 Problem 1.

1.1 Tokenization (10pts)

To solve this problem, the regular expressions and lookahead & lookbehind Zero-Length Assertions are being used here.

First of all, the tokenizer need to match all of the [BAR], [/BAR], [VERSE #Number#]. Like I listed below, the value "BAR" is included the label "[" and "]".

The value "ruleForBAR" is the rule that if element is surrounded by the [ ], that it should be separated, except the "BAR]" And the value "ruleForBAR2" is almost the same, when element surrounded by [ ], that should be separated except the "[BAR", "[/".

When the tokenizer executed the rules above, the [VERSE 1] would be separated as "[", "VERSE", "1", "]", but the [BAR] and [/BAR] will remain the same.

```
// "[  ]" [BAR] [/BAR] [VERSE 1,2,3...]
val BAR = "[\\]\\[]"
val ruleForBAR = s"(?<!BAR)(?=$BAR)"
val ruleForBAR2 = s"(?<=$BAR)(?!BAR)(?!\\/)"
```

Fig.1 Rules For BAR

Secondly, the tokenizer need to match the symbol, single quotation mark " ‘ " by using the "ruleForQM". Normally, the word that followed with " ’ " need to be separated, such as "it ’s", "shinin' " except the letter "n" that followed with " ‘ ". And when it comes into "n’t", that should be treated as a union, such as "can’t" separate into "ca", "n’t".

```
// " ' "  n't
val quotationMark = "\\'"
val ruleForQM = s"(?<!n)(?=$quotationMark)|(?=n\\'t)"
```

Fig.2 Rules For Single quotation mark

Thirdly, the tokenizer need to match the symbol, full stop ".". Usually, the tokenizer can separate it directly, but in this case, there are some other words such as "Dr.", "Mr.", so in

values ruleforDr and ruleforDr2, if the tokenizer meets “a sentence.”, it will separate it into ”a”, “sentence”, “.”, but if the word before “.” Is “Mr” or “Dr”, it will treat them as a union and ignore the blank between “r” and “.”.

```
// Dr.Mr.
val fullstop = "[\\.]"
val ruleforDr = s"(?<!Dr|Mr)(?=$fullstop)"
val ruleforDr2 = s"(?<=$fullstop)"
```

Fig.3 Rules for Dr and Mr

Finally, for all other punctuations, the tokenizer can easily just separate them individually. The list of all the rest of the punctuations are listed below.

```
// Punctuations
val symbol = "[\\,\"\\?\\!\\(\\)\\*\\{\\}\\;\\:\\+\\_]"
val RuleForPunctuations = s"(?=$symbol)"
val RuleForPunctuations2 = s"(?<=$symbol)"
```

Fig.4 Rule for the rest punctuations

One more thing is that the tokenizer should contain the “\\s” into the final procedure, so that the word could be separated by the original blank.

```
//Final Procession
val tokenizer = Tokenizer.fromRegEx(s"(\\s|$ruleForBAR|$ruleForBAR2|$ruleForQM" +
  s"|$RuleForPunctuations|$RuleForPunctuations2|$ruleforDr|$ruleforDr2)")
```

Fig.5 tokenizer

2 Bar-Aware Language Model (total 55pts)

2.1 Data Analysis (5pts)

Traverse the whole List[String] which is created by “loadWords” function. The initial value of “counts” equals to zero, when it encounters the words except [/BAR], the counts should add one, else it should be store in a ListBuffer[Int]. After this procedure, we can use the function “groupBy” to count what exactly times that every distance has shown, such as ”0 -> 439, 5 -> 260”. We can create a histogram after all of the data are being input in Excel. The graph is shown below.

In this graph, there are about 5611 elements, which means there are 5611 pairs of [BAR] [/BAR]. The counts are increased as the distance increased from one to ten, peak at 659, and then decreased smoothly from 11 to 29. One more thing, when the distance equals to 28, the count of this bar is 0.

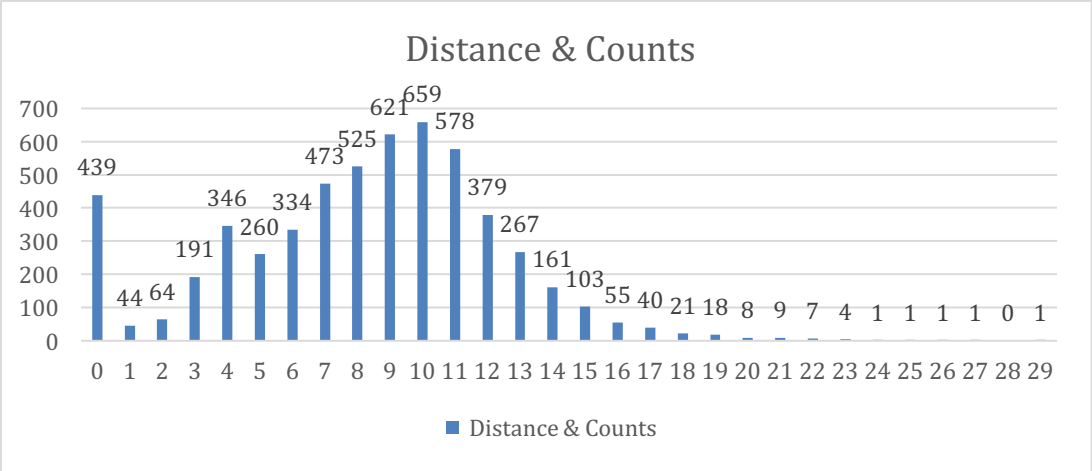

Fig.6 histogram of distance and counts

2.2 Linearly Bar-Aware: Model (10pts)

As the requirement presents, the probability of seeing [/BAR] is linearly increasing with the distance from the last [BAR]. And also, the other words' probability is followed the uniform language model, which means they should decrease as the probability of seeing [/BAR] increases. We assume that:

$$\mathbf{Y = AX + B}$$

In this equation, the variable X represent the distance from the last [BAR] and the variable Y represents the probability of seeing [/BAR] in the next word. Both of the A and B are parameters there, and it can lead to the best perplexity, which means the minimum perplexity can be used to predict parameters A and B.

$$\boldsymbol{Y' = \frac{1 - (AX + B)}{|V| - 1}}$$

To meet the requirement, this equation is using unigram language model for calculating the probability of next word which is not the [/BAR]. Both A and B are parameters that we may calculate later and |V| represent the size of the vocabulary set in the document. The "1" in numerator is represent the probability 1, and the other "1" is the count that we should decrease the size of element [/BAR] in the size of vocabulary size.

2.3 Linearly Bar-Aware: Implementation (10pts)

The parameters A and B's range can be calculated through data in question 2.1 and equation in 2.2. As the figure.6 shows, the [/BAR] appears 439 times when the distance is 0, so the parameter B should be about

$$\mathbf{B = Y = \frac{Count(distance = 0)}{Count(BAR\ pairs)} = \frac{439}{5611} = 0.0782}$$

What's more, the order in language model is 20, which means that the X in equation should be no more than 20, and the range of parameter A is about

$$\mathbf{A = \frac{1 - B}{X} = \frac{1 - B}{20} = 0.046}$$

But I suggest that we should use Maximum Likelihood Estimate(MLE) to calculate the probability of each distance and then use linear regression to calculate the exact parameters of A and B. Anyway the parameter B here is 0.0782 as a constant, trying to find the best parameter A via calculating the perplexity. The Fig.7 I listed below show the relationship between parameter A and perplexity. The perplexity of using this linearly Bar-aware language model is 4285. And the only difference between linearly Bar-aware language model

and uniform language model is that when the next word is [/BAR], the previous one language model would return

$$\mathbf{P(distance) = 0.00649|distance| + 0.0782}$$

, and the uniform language model will return $\frac{1}{|V|}$ instead. The perplexity of uniform language model is 7313, which is much lager than the linearly one.

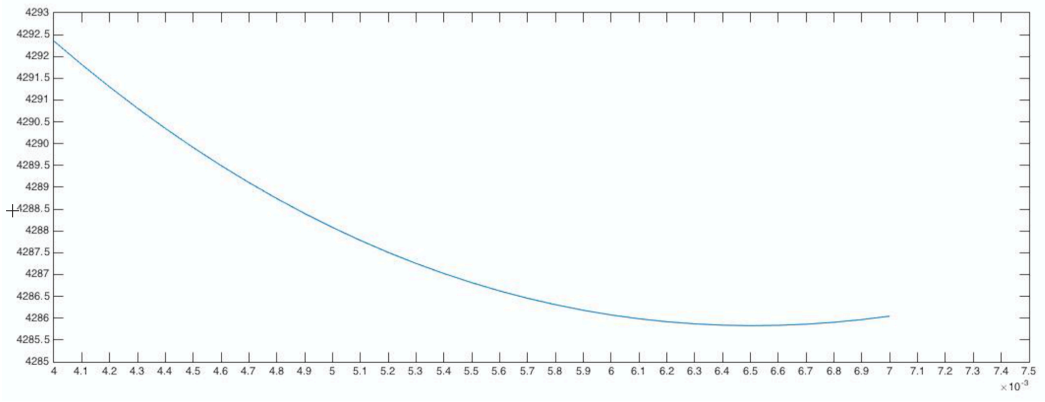

Fig.7 relation between parameter A and perplexity

2.4 Per-Distance Bar-Aware: Model (10pts)

For k times success in n times trials, the probability can be calculated as

$$\mathbf{P(k)} = \binom{n}{k} p^k (1-p) n - k$$

As the requirement indicates, the Maximum Likelihood Estimate(MLE) should be calculated, and the MLE for probability should be

$$\mathbf{maxP(distance)} = \binom{n}{k} p^k (1-p) n - k$$

, generally probability for MLE is calculated as $\frac{k}{n}$.

So in this section, the parameters for each distance can be calculated as

$$\mathbf{P(distance)} = \frac{\boldsymbol{counts(Distance)}}{\boldsymbol{count(BAR\ pairs)} - \sum_{i=0}^{Distance-1} \mathbf{counts(i)}}$$

and a Map<Key, Value> structure can be used to store both of the distance and probability. And I have set the order equals to 20, so that the maximum of distance should be 19. If the distance is more than 19, which means the [/BAR] is not in the history, the probability at this time should be return P(distance = 19).

For instance, in this language model, if the next word is "[/BAR]", the probability depends on the distance, which can be checked in the Map< Distance, Probability >, and as the strategy that showed before, if the distance is between 0 to 19, it should return the corresponding probability in Map, and if the distance is lager than 19, it should return the P(Distance =19). If the next word is the others and distance equals to L, it should follow the rule of uniform language model, which mean it should return $\frac{1-P(Distance=L)}{|V|}$.

The perplexity of this Per-Distance Bar aware model is 4215.

2.5 Per-Distance Bar-Aware: Implementation (10pts)

(a)As question 2.4 stated, by calculating the MLE, simply using the

$$\textbf{Maximum Likelihood Estimate} = \frac{k}{n} = \frac{\boldsymbol{counts(Distance)}}{\boldsymbol{count(BAR\ pairs)} - \sum_{i=0}^{Distance-1} \mathbf{counts(i)}}$$

(b)For optimizing perplexity in the previous language model, we can add a parameter alpha

during the processing of calculating the MLE:

$$\textbf{Maximum Likelihood Estimate} = \frac{(k+\alpha)}{n+\alpha|V|} = \frac{counts(Distance)+\alpha}{count(BAR\ pairs)-\sum_{i=0}^{Distance-1} counts(\mathrm{i})+\alpha*|V|}$$

In this equation, |V| represent the times we counted the probability. Because when the alpha being counted every single times, the denominator should be added into the same amount alpha, and if the probability being counted for |V| times, the alpha*|V| should be added into the denominator. Because the order equals to 20, the times that being calculated should be 20 too, which means |V| = 20 in this language model. After adding the suitable parameter alpha into the equation, which has been tested for more than 50 times, the perplexity is 4198, which is a little bit lower than the previous one.

2.6 Interpolation and Start of Bars (10pts)

By adding a new condition that seeing a [/BAR] when [BAR] follows in the probability function, the perplexity decreased rapidly to 1796.

Also, interpolated language model can be used there(the original one with unigram model). After testing the parameter alpha in union language model, the alpha should be 0.3, and at this moment, the perplexity is 412.

$$P(distance) = 0.3 * \left(\frac{(k+\alpha)}{n+\alpha|V|}\right) + 0.7 * \frac{counts(word)}{counts(words)}$$

$$= 0.3 * \left(\frac{counts(Distance)+\alpha}{count(BAR\ pairs)-\sum_{i=0}^{Distance-1} counts(\mathrm{i})+\alpha*|V|}\right) + 0.7 * \frac{counts(\mathrm{word})}{counts(words)}$$

In this equation, counts(word) represent the times that a word occurs. Counts(words) represent the whole size of the training document, which is equal to 7314. It is clearly that in unigram language model, there is no difference between the [/BAR] and other words. As the question requested, the bar length distribution on these 1000 words are listed below. And also I have tested both the 5000 samples bar length distribution and the 50000 one. Because there are not enough [BAR] pairs in the 1000 samples, the tendency is not clearly enough. But as the samples numbers increased, the tendency is much clear than the previous one, and it is really close to the graph in Q2.1. The data are being trained in the language model, and it became much more smoothing. The counts keep increasing when the distance is between 1 to 9, but the trend is quite smooth. Also when the distance is larger than 10, the number decreased but in a smoothing way. There are still any existences when the distance is lager than 30 due to the smoothing procedure.

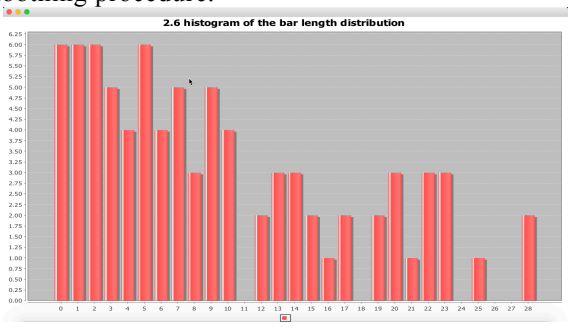

Fig.8 bar length distribution of 1000 words

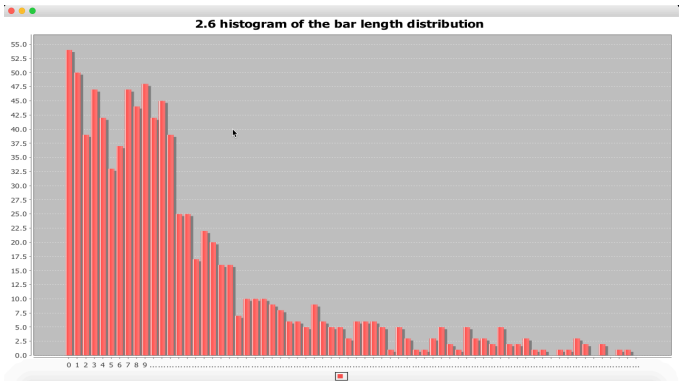

Fig.9 bar length distribution of 5000 words

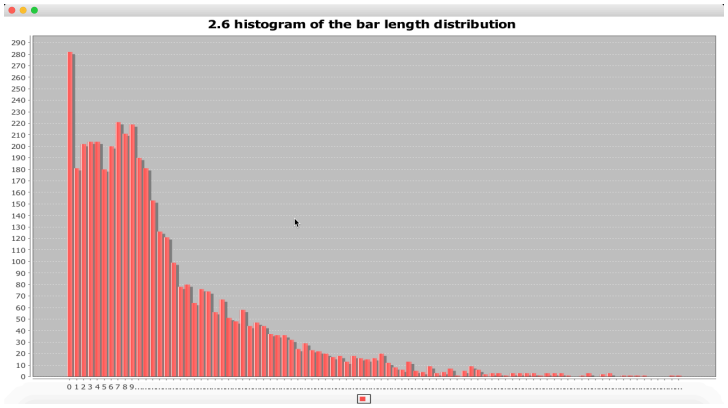

fig.10 bar length distribution of 50000 words

3 Language Models for Document Classification (25pts)

3.1 Model (10pts)

Usually, X represent the data and Y represent the label in document classification. And in this question, the aim is to solve the predicting problem, which means calculation of the P(Y|X) for each label is required. And the definition of the noisy channel model should be:

$$\hat{y}(x) = \underset{y \epsilon Y}{\mathbf{argmax}}(Y = y | X = x)$$

According to the background, the relationship between author(Y) and corresponding lyrics(X) is required.

Because of the Bayes rules, the noisy channel model can be inverted to:

$$\mathbf{P(Y|X)} = \frac{P(X|Y)P(Y)}{P(X)}$$

and because of our goal is to calculate the $\hat{y}(x)$, the P(X) can be ignored since it is a constant. Also, we assume that the position of the words in the lyrics does not matter and each of the word in lyrics are independent to the given author. So P(Y|X) is listed as below:

$$\mathbf{P(Y|X)} = \arg\max_{y} P(X1, X2, X3..Xn|Y)\mathbf{P(Y)} = \prod_{k=1}^{n} P(\mathbf{X^k}|Y)\mathbf{P(Y)}$$

At this moment, the P(Y) in the equation is the language model, it can be trained with the existing data, then it could return the probability of the corresponding author Y. The language model should be unigram model there, so the order in this model should be 1. It is because during the calculating of probability of the corresponding author Y, the language model do not require the history information, all it need are the counting numbers of the author's name.

The best way to improve the language model for this task is to consider both the relation and positon of all the words that shows in lyrics.

3.2 Implementation (15pts)

The language model above used the "p3_train.txt" as training set, and used both "p3_dev.txt" and "p3_test.txt" as testing set. The classification accuracy is 0.9 and 0.9666666666666667 respectively.

First of all, it is clearly shown in 3.1 that $\mathrm{P}(\mathrm{Y}|\mathrm{X})$ should be calculated as below:

$$\mathbf{P(Y|X)} = \arg\max_{y} \boldsymbol{P(X1, X2, X3..Xn|Y)P(Y)} = \prod_{\mathbf{k=1}}^{\mathbf{n}} \boldsymbol{P}(\mathrm{X}^{\mathbf{k}}|\mathbf{Y})\mathbf{P(Y)}$$

and make a comparison for each of the author. The language model provides the priors, which is the P(Y) in the equation. And for the $P(\mathrm{X}^{\mathrm{k}}|\mathrm{Y})$, we can calculate via

$$\boldsymbol{P}(\mathrm{X}^{\mathbf{k}}|\mathbf{Y}) = \frac{\boldsymbol{count}(\mathrm{X}^{\mathbf{k}}, \mathbf{Y}) + \boldsymbol{\alpha}}{\boldsymbol{count(Y)} + \boldsymbol{\alpha}|\boldsymbol{V}|}$$

In this equation, $count(\mathrm{X}^{\mathrm{k}}, \mathrm{Y})$ is the times that word X^{k} appear in lyrics which is written by author Y. Alpha is one of the parameters we defined in 3.1. count(Y) is the total length of all the class that belong to the author Y. Finally, the |V| is the count of all the words that once appear in all of the class.

4 Ignorant Machine Translation (10pts, no coding required)

First of all, we need to translate the rap song in German into English one. According to the noise channel model, that is finding:

$$\arg\max_{t} \mathbf{p(English|German)} = \arg\max_{t} \mathbf{p(English)\ p(German|English)}$$

In this equation, p(English) is the unigram language model, and p(German|English) is equal to p′(German|English(length)). The order in unigram language model is 1, that means this language model does not need to reference the history, and the return value which is the probability of this language model should be

$$\mathbf{Probability(Length = L)} = \frac{\boldsymbol{Count}(\mathbf{Length = L})}{\boldsymbol{Count}(\mathbf{Length})}$$

So the $\arg\max_{t} \mathrm{p(English|German)}$ of different English songs should be $(\mathrm{P}'(g|length(English))\frac{\boldsymbol{Count}(\boldsymbol{length=L})}{\boldsymbol{Count}(\mathbf{length})})$ For example, there are two English songs. The first one's length is 50, and another one is 40. Also, there are 10 songs' length is 50, 20 songs' length is 40, and the number of all the English songs are 100. The $\arg\max_{t} \mathrm{p(English|German)}$ of previous one is $(\frac{10}{100}(\mathrm{P}'(g|length = 50))$ while the other one is $(\frac{20}{100}(\mathrm{P}'(g|length = 40))$. The larger one should be the suitable choice for the correct translation.

The order in bigram language model is 2, which means the probability should reference the previous song. When calculating the probability of length = L, the equation should be

$$\mathbf{Probability(Length = L)} = \frac{\boldsymbol{Count}(\boldsymbol{L}\mathbf{ength = L\ \&\ Previous\ length = L'})}{\boldsymbol{Count}(\mathbf{Length = L'})}$$

For example, when calculating the probability of Length = 40, we should not only count the number while Length = L, also need to count the previous one. If the Count(Length = 40 & Previous Length=30) equals to 50, and Count(Length=30) equals to 100. The $\arg\max_{t} \mathrm{p(English|German)}$ should be is $(\frac{50}{100}(\mathrm{P}'(g|length = 40))$ The rest are all the same comparing to the unigram model.

We can use Interpolated language model to improve the performances of there two models. After calculating the probability of both the unigram model and bigram model, we can use the parameter alpha to combine both of the result,

$$\mathbf{p}(\mathbf{English}) = \left(\frac{\boldsymbol{Count}(\mathbf{Length = L})}{\boldsymbol{Count}(\mathbf{Length})} * \boldsymbol{alpha}\right) + \left(\frac{\boldsymbol{Count}(\boldsymbol{L}\mathbf{ength = L\ \&\ Previous\ length = L'})}{\boldsymbol{Count}(\mathbf{Length = L'})} * (\mathbf{1} - \boldsymbol{alpha})\right)$$

Interpolated language model can obviously increase the accuracy and efficiency.

5 pen-Ended Language Model Development (Optional, 10pts)

The language model in question 5 is based on the one in question 2.6. In the previous language model, the words which are not [BAR] or [/BAR] should be applied to the uniform model. But in this section, we can use the unigram model instead.
The perplexity is 336 after renewing the language model. The perplexity could be lower if bigram model is used in calculating the [/BAR]'s probability.

Acknowledgments

Thanks for the all the tutors and classmates that answer my question in the discussion forum.

References

[1] University of Massachusetts Amherst, 2009, Introduction to Natural Language Processing. [pdf] Massachusetts, Available at:< https://people.cs.umass.edu/~dasmith/inlp2009/lect5-cs585.pdf> [Accessed 9 November, 2015]

[2] Macquarie University, 2014, The Noisy Channel Model and Markov Models. [pdf] Sydney, Available at:<http://web.science.mq.edu.au/~mjohnson/papers/Johnson14-04LM-talk.pdf> [Accessed 9 November, 2015]