ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Assignment 1

Text Analytics

Submitted by:

- Panagiotis Antoniozas
- Spyros Mastrodimitris Gounaropoulos
- Panagiota Tavoularea

➢ **You can access our Our Google Colab link here**

Introduction

For this assignment we chose the “Genesis” corpus that the “NLTK” library provides. The Book of Genesis is the first book of the Bible and tells the story of the creation of the world and the origins of humanity.

Question (i)

After loading the corpus text into a python list, we performed the data preprocessing. More specifically, we converted the letters to lowercase, so as to help us create a more efficient spelling corrector.

Next, we proceed to the tokenization at sentence and word level (using TweetTokenizer), so as each sentence to be formatted as the examples below.

Example 1	['in', 'the', 'beginning', 'god', 'created', 'the', 'heaven', 'and', 'the', 'earth', '.']
Example 2	['and', 'god', 'called', 'the', 'firmament', 'heaven', '.']

Following the tokenization, we created the vocabulary. The basic idea for the vocabulary creation was to keep only the words that appear more than 10 times in the corpus, in order to deal with rare words that may mislead our model.

Afterwards, we created a function that takes as input a list of sentences having the above format, and replaces the words that are not included in the vocabulary with the special token “*UNK*” (Unknown). Below, we present some useful metrics for better understanding of our text.

Total Text Words	***Total Non-Vocabulary Words (“*UNK*”)***	***Total Vocabulary Words***
44256	5632	411

As we can see, there are many unknown words and also the words included in the vocabulary are few, which means that many of the words are repeated. We should have in mind that they might have an impact on the overall performance of our language models.

In the next step, we split the data to train, test, dev. Firstly, we split the data to train and test using the 80/20 rule. Afterwards, on the already created train set, we also utilized the 80/20 rule

in order to create our final train and dev set. We will use the train set to train our models and the test and dev sets for model evaluation and hyper-parameter tuning respectively. In the table below we can see the number of sentences that each set contains.

Train Set	***Test Set***	***Dev Set***
1173	235	234

Consequently, we created the unigrams, bigrams and trigrams (the unigrams will help us to calculate the bigrams and trigram's log-probabilities).

In particular, for the unigrams we added at the beginning of each sentence the token <s> that indicates the start of the sentence. For the bigrams, we added at the beginning of each sentence the token <s> and at the end, the token <e>, that indicates the start and the end of the sentences respectively. For the trigrams, we added at the beginning of each sentence the tokens <s>, <s> and at the end, the token <e>, <e>, that indicates the start and the end of the sentences respectively.

But, at the log-probabilities calculation step, we faced a problem. We were not able to correctly calculate the following probability: (we do not use Laplace smoothing in order to be more clear in our example).

$$P(and \mid < s >, < s >) = \frac{c(< s >, < s >, and)}{c(< s >, < s >)} \quad [1]$$

It is clear that none of the bigrams have $(< s >, < s >)$ at the beginning of the sentence, so $c(< s >, < s >)$ will be equal to zero. To deal with this problem, we implemented an if statement in the for loop for the log-probabilities calculations. Basically, if we came across a trigram that starts with <s>, <s>, then we calculate its probability using the corresponding unigram counter like in the following example.

$$P(and \mid < s >, < s >) = \frac{c(< s >, < s >, and)}{c(< s >)} \quad [2]$$

Then, in order to create the bigram and trigram language models, we computed the sum of the logarithms for each sequence of the n-gram probabilities instead of their product. Probabilities are likely to be very close to zero and that can cause numerical stability issues and lead to loss of precision in computation. Also, we have to note that we used Laplace smoothing (a=1) for the probabilities calculation. The final log-probabilities are presented in the following table.

Sum of Bigrams Log-Probabilities	-59527.324
Sum of Trigrams Log-Probabilities	-153734.954

Question (ii)

For the Cross Entropy & Perplexity Calculation we did not include the probabilities of the form $P(*\,start\,*|\ldots)$ or $P(*\,start1\,*|\ldots)$, $P(*\,start2\,*|\ldots)$, since we are not predicting the start pseudo-tokens, but we included probabilities of the form $P(*\,end\,*|\ldots)$, since we do want to be able to predict if a word will be the last one of a sentence.

So, in order to avoid (<s>, first_word) bigrams for the bigram language model we simply change the for loop (that passes through words), to start from the 2nd word. Similarly, to avoid (<s>,<s>,firstword) and (final_word,<e>,<e>) trigrams for trigram language model we simply change the for loop (that passes through words), to start from the 3rd word and finishes to len(sentence) -1.

The **cross entropy** and **perplexity** for each model is presented in the following table:

	Cross Entropy	***Perplexity***
Bigram	5.326	40.120
Trigram	6.307	79.184

It seems that the bigram model performs better than the trigram on the test set (because it has lower cross entropy and perplexity score). Receiving better results on a bigram model rather than a trigram model is something unusual.

But, as we can see from the second table, we have a lot of unknown words in our text. Also the Book of Genesis, as the story proceeds and passes from one fact to another the corresponding topic vocabulary changes too. All the above, in combination with its small size, lead to miss words which are not so rare in other texts. For instance, next we present some examples of rare words by implementing the (count<10) rule:

Sample of Unknown Considered Words:

- part
- reason
- plenty
- none
- still
- never
- peace
- answer
- understand
- remember
- sad

Unknown words, can be any part of speech, therefore can exist after or before any word.This means that trigrams or bigrams containing "*UNK*", practically do not give extra information concerning the next words. Below, we can see the percentages of bigrams and trigrams containing unknown words.

*Percentage of Bigrams Contain "*UNK*"*	*Percentage of Trigrams Contain "*UNK*"*
7.37 %	26.59 %

It is obvious that trigrams have three times as many "*UNK*" as the corresponding bigrams. Thus, it is more difficult to predict the next word using a trigram model rather than a bigram language model and this fact explains to a certain extent why bigram language model has better performance.

Question (iii)

In order to create the context aware spelling corrector we have to incorporate both the Levenshtein distance and the bigram (or trigram) language model probabilities. An exhaustive search is not by far the most efficient option. For this exact reason, we use a heuristic search algorithm and specifically Beam search.

Our input is a list of strings (that being the sentence), the bigram or the trigram model, the vocabulary and the Beam search width set at 2. The first step is to initiate the sentences with the node <s>, so we are able to calculate the bigram of the first word (or <s><s> for the trigram). For the implementation of the Beam search, we use pseudo-tree so each word is expressed as a tree node and the tree's root is the <s> node. For each new word we calculate the Levenshtein distance from all members of the vocabulary. The 10 words with the shortest distances are selected as candidates. In order for a candidate to be selected as the best fit we need to maximize the following formula.

$$L_k = \lambda_1 \, log \, P\left(t_1^k\right) + \lambda_2 \, log \, P\left(w_1^k \middle| t_1^k\right)$$

The first part of the formula is produced from the language model (probability of occurrence between the candidates and his parent node in the corpus) and the second part is based on the inverse Levenshtein distance as mentioned above. The top candidates are considered the parent nodes for the next iteration. The aforementioned procedure is repeated for every word in the sentence. As an output is given the inverted leaf to the root path as the closest fit.

It is important to mention that the inverse Levenshtein distance was approached in two different ways. The first was a normalised ratio and the second was adding an offset to the denominator. Among experimental tests, it was observed that the second method produced in general had better results.

Concerning hyper parameter tuning, it was done in two stages. After the development of the Beam search algorithm, multiple tests were done in order to balance the effects of bigram-trigram models and word distance ($\lambda 1, \lambda 2$) . The second stage was about Beam width and the number of candidates and was completed after question (v), using the dev test, the question

(iv) randomizer and the question (v) CER, WER functions. An even more extensive presentation is given at the last page of the report.

As the beam searches return the two best sentences accompanied with their probabilities, two caller functions were created to keep only the first sentence and drop the <s> elements.

Input	This is a simple sentence	and lrd saild
Bigram output	['<s>', 'this', 'is', 'a', 'time', 'thence']	['<s>', 'and', 'lord', 'said']
Trigram output	['<s>', '<s>', 'this', 'is', 'a', 'time', 'thence']	['<s>', '<s>', 'and', '.', 'lord', 'said']

Input	and gd seperated the ligth from the darkness	. and
Bigram output	['<s>', 'and', 'god', 'departed', 'the', 'light', 'from', 'the', 'dukes']	['<s>', '.', 'and']
Trigram output	['<s>', '<s>', 'and', 'god', 'departed', 'the', 'light', 'from', 'the', 'dukes']	['<s>','<s>', '.','and']

It is apparent that the model fails to recognize existing words as correct. The reason behind the problem is the very limited vocabulary range, meaning that each word absent from the vocabulary is treated as misspelt. The words that are indeed misspelt are successfully corrected. The bigram (, , and) has the highest probability of appearance, also the distance between (, .) is 1. So, a lesser corrector might misinterpret the one for the other if the language model is dominant and our algorithm seems unaffected.

Question (iv)

For the evaluation of the spelling corrector it was necessary to provide the test set, the ground truth and the modified test set to be corrected. This modification was made possible using our hard coded randomizer, using 3 for loops to scan every character of every word of every sentence inside the test set and with a possibility of 25% to alter it to random lower or uppercase character.

After the randomised test set is ready, we test the first 10 sentences of the test set showing the originals, the randomised and the corrected results for the bigram and the trigram spelling correctors. Some of the sorter sentences are presented below.

Bigram spelling corrector output

Original	Randomised	Corrected
'and', 'the', 'sons', 'of', '*UNK*', ';', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', '.'	'USu', 'tue', 'aoqs', 'oU', '*UNK*', ';', '*UNK*', ',', 'ane', '*UNK*', ',', 'dnd', '*UNK*', ',', 'and', '*UNK*', ',', 'WnT', '*UNK*', '.'	'and', 'the', 'sons', 'of', 'the', ';', 'and', ',', 'and', 'the', ',', 'and', 'the', ',', 'and', 'the', '.'
'what', 'shall', 'we', 'speak', '?'	'whAt', 'shall', 'we', 'sphrk', '?'	'what', 'shall', 'we', 'speak', '?'
'and', 'they', 'made', 'their', 'father', 'drink', 'wine', 'that', 'night', 'also', ':', 'and', 'the', 'younger', '*UNK*', ',', 'and', 'lay', 'with', 'him', ';', 'and', 'he', '*UNK*', 'not', 'when', 'she', 'lay', 'down', ',', '*UNK*', 'when', 'she', '*UNK*', '.'	'and', 'they', 'madN', 'theil', 'fatheD', 'drBnw', 'winb', 'that', 'gight', 'also', ':', 'anS', 'tha', 'yoYnper', '*UNK*', ',', 'aFs', 'EaL', 'wfch', 'him', ';', 'Qnd', 'he', '*UNK*', 'Uot', 'wRen', 'sge', 'lac', 'dowU', ',', '*UNK*', 'Xhen', 'zhe', '*UNK*', '.'	'and', 'they', 'made', 'their', 'father', 'drink', 'wine', 'that', 'night', 'also', ':', 'and', 'the', 'younger', ',', ',', 'and', 'was', 'with', 'him', ';', 'and', 'he', 'and', 'not', 'when', 'she', 'lay', 'down', ',', 'and', 'when', 'he', 'and', '.'

The spelling corrector suffers heavily from the overabundance of unknown words that tend to be the majority of a few sentences. Also since the term *UNK* is not in the vocabulary the model tries to find the closest matching words. Those tend to be the most common three character words in the train set(and, the and now). Beyond the problem of failing to guess a word that it does know, sometimes this wrong assumption creates a cascading effect on the prediction of the next word as it takes into consideration different bigram probabilities.

On the other hand the model is capable of returning very accurate predictions if the number of unknowns is small, even when more than one letters were altered in the same word.

Trigram spelling corrector output

Original	Randomised	Corrected
'and', 'the', 'sons', 'of', '*UNK*', ';', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', ',', 'and', '*UNK*', '.'	'USu', 'tue', 'aoqs', 'oU', '*UNK*', ';', '*UNK*', ',', 'ane', '*UNK*', ',', 'dnd', '*UNK*', ',', 'and', '*UNK*', ',', 'WnT', '*UNK*', '.'	'and', 'the', 'sons', 'of', ',', ';', ',', ',', 'and', ',', ',', 'and', ',', ',', 'and', ',', '.'
'what', 'shall', 'we',	'whAt', 'shall', 'we',	'what', 'shall', 'we',

'speak', '?'	'sphrk', '?'	'speak', '?'
'and', 'they', 'made', 'their', 'father', 'drink', 'wine', 'that', 'night', 'also', ':', 'and', 'the', 'younger', '*UNK*', ',', 'and', 'lay', 'with', 'him', ';', 'and', 'he', '*UNK*', 'not', 'when', 'she', 'lay', 'down', ',', '*UNK*', 'when', 'she', '*UNK*', '.'	'and', 'they', 'madN', 'theil', 'fatheD', 'drBnw', 'winb', 'that', 'gight', 'also', ':', 'anS', 'tha', 'yoYnper', '*UNK*', ',', 'aFs', 'EaL', 'wfch', 'him', ';', 'Qnd', 'he', '*UNK*', 'Uot', 'wRen', 'sge', 'lac', 'dowU', ',', '*UNK*', 'Xhen', 'zhe', '*UNK*', '.'	'and', 'they', 'made', 'their', 'father', 'drink', 'wine', 'that', 'night', 'also', ':', 'and', 'the', 'younger', ',', ',', 'as', 'a', 'with', 'him', ';', 'and', 'he', ',', 'not', 'when', 'she', 'lad', 'down', ',', ',', 'when', 'the', ',', '.']

Through a short observation it is clear that the trigram model failed to surpass the accuracy prediction of the bigram. In fact, it is less effective by making mistakes not seen in the bigram model. As the main factor we believe is the highest concentration of unknown words and their highest probability appearing in a word triplet, turning the idea of a trigram from a strength to a weakness. Testing showed that by dropping the requirement from 10 to 2 to include a word in the vocabulary drastically improved its accuracy. Of course that massively worsened the cross entropy and the perplexity so it was not pursued further. Although we considered it important enough to mention it, as it provides a deeper understanding to the causes of the trigrams spelling corrector problems.

Question (v)

For the estimation of the metrics of word error rate(WER) and character error rate(CER) we used the provided examples using the existing library functions. As the ground truth we chose the test set and the Beam search took as an input the randomised set created in question (iv). From this procedure we had the following findings.

	WER	***CER***
Bigram	0.18	0.21
Trigram	0.23	0.24

The produced results are not that surprising, the question (iv) examples had already revealed that the bigram spelling corrector had superior results. Those assumptions based on independent examples are confirmed through a general and extensive WER, CER calculation.

The majority of the errors are produced because of the large number of unknowns. It was calculated that when the spelling corrector was provided with the ground truth returned a WER of 0.13. That means that only 27.8% of the 0.18 WER were caused by words that the model had in its vocabulary and failed to predict correctly.

Hyperparameter Tuning

Bigram spelling corrector

Width	***1***	***2***	***3***	***4***	***5***	***6***	***10***
WER	0.235	0.233	0.239	0.240	0.2432	0.245	0.249
CER	0.2512	0.2513	0.2551	0.2565	0.2573	0.2591	0.262

As the width increases the WER and CER increase with the single exception being the width of 2 that achieves better WER and similar CER. Those results are confirmed by multiple tests and different randomised validation sets. For that reason width 2 was selected.

In addition during the width test we also experimented with the number of candidates. The results showed a drop in performance, in terms of time and results, when the number of candidates increased to more than 10.

Trigram spelling corrector

Width	***1***	***2***	***10***
WER	0.2554	0.2554	0.2554
CER	0.2554	0.2554	0.2554

Even though extensive tests took place the trigram spelling corrector proved very stable and any hyperparameter tuning showed little to no effect on the results.