1st Assignment: N-gram Language Models in the context of "Text Analytics"

CONTRIBUTORS: DIMITRA TSAKIRI (f3352123), PANTELEIMON SFAKIANAKIS(f3352121), DIMITRIOS GKAVERAS(f3352104), ALEXANDROS SKONDRAS (f3352119)

The link for our Google Colab notebook can be found here.

Question 3

For the causes of this assignment, we chose to work with a corpus offered by the NLTK library named *"Australian Broadcasting Commission 2006"*.

Question 3 (i)

In this question our main purpose is to implement a bigram and a trigram model based on the corpus that we downloaded. We start by tokenizing the text to be formed as a list that contains lists of each sentence per word so that it is properly formed for use. Then, we split our tokenized text in three sets (train, test and dev) that have size as shown in the following table. At this point we should add that the size of every set contains the number of the sentences and that the three sets are split randomly.

Train Set	Test Set	Dev Set
10399	**2580**	**20**

Then we proceed by searching for the words that are used in our corpus at least ten times to establish our vocabulary. The words that are not included in the vocabulary which practically means that they are used less than ten times, were replaced in every set of the corpus by *UNK*, a string that symbolizes an unknown word. The vocabulary has 2802 words in total. Then, only for the text in our train set, we calculate the unigrams, bigrams and trigrams based on the frequencies of the words. For the bigram we add a pseudo-token <s> and a pseudo-token <e>, so that we can calculate the probability of a word to be at the start or at the end of a sentence, respectively. We use the same pseudo-tokens for the trigram too, but instead of <s> and <e>, we placed (<s>, <s>) at the start of each sentence and similarly at the end, (<e>, <e>). After the calculation of the bigrams and the trigrams, we implement Laplace smoothing and in order to answer this question fully we proceed by computing the sum of the logarithms of the n-gram probabilities since their product would be a very small number that would possibly cause an exception problem in our code. Thus, we take the logarithms of their products to, eventually, transform the bigram and trigram probabilities into numbers of a larger scale. This turns the computation from the form of products into a form of sums, which is a notably more efficient calculation.

The following table shows the sum of the logarithms of the n-gram probabilities for both models:

Bigram model	Trigram model
725019.38	1709049.55

Question 3 (ii)

For the causes of this section, we calculated the cross entropy for both models with and without the use of libraries. Their values are shown in the following table:

Cross Entropy	Without Library	Using Nltk
Bigram model	7.92	11.45
Trigram model	9.88	11.43

It is clear that there are some significant differences between the values of the above table, depending on the method used for the bigram and trigram models. In the case we did not use any library for the calculation of entropy, the bigram model is more accurate, but in the case we use libraries for our calculations, the trigram is more efficient. This can be explained by the fact that in both methods the vocabulary we used was slightly different, the vocabulary that the library creates contains by default <UNK>, <s> and </s>. It is also possible that calculation errors affected the outcome, due to numerical approximations that might have been applied in both library and non-library methods.

We followed the same process for the calculation of perplexity and the outcome of the calculations are shown below.

Perplexity	Without Library	Using Nltk
Bigram model	241.42	2805.99
Trigram model	939.53	2773.08

It is obvious that the same differences of the values of entropy are shown in terms of perplexity, too. The reasons that explain these differences are the same as above. The differences between the two vocabularies result in different calculations.

In general, it is usually expected for the bigram model to be less accurate than the trigram model because the latter includes the bigram's information, but we should underline that this is not guaranteed. The main reason behind a more accurate bigram model than a trigram one can be the content of the train set. In our case the corpus is full of the words: he, she, said, says, ", so we have a lot of unseen trigrams that cause issues when we calculate the entropy or the perplexity of our test set.

Question 3 (iii)

In this question, we created a context-aware spelling corrector using Beam Search Decoder and the bigram language model we made in Question 3(i). We split the implementation of this endeavor into 3 steps. First, we tokenize the chosen sentence about to be analyzed and add the tokens <s> and <e> at the start and the end, respectively. After that, the tokenized sentence turns into an array and we proceed to the creation of a new array (named sentence_list) that contains a dictionary for every observed token-word in the sentence of interest. Each dictionary contains the probabilities of a specific token-word derived from its Levenshtein distance from every possible word in the vocabulary we established in the creation of the bigram language model. In Step 3 of the Beam Search Decoder implementation, the Levenshtein probabilities are turned into log(probabilities) for the reasons we explained in Question 3(i) and the Beam Search Algorithm is applied, according to which for every observed word of the sentence we are analyzing, we keep the 3 candidate words of the vocabulary that minimize:

$$-\lambda_1 \sum_{i=1}^{k} \log\left(P(t_i|t_{i-1})\right) - \lambda_2 \sum_{i=1}^{k} \log\left(P(w_i|t_i)\right)$$

$$where \quad t_i: candidate\ word \quad and \quad w_i: observed\ word \qquad (*)$$

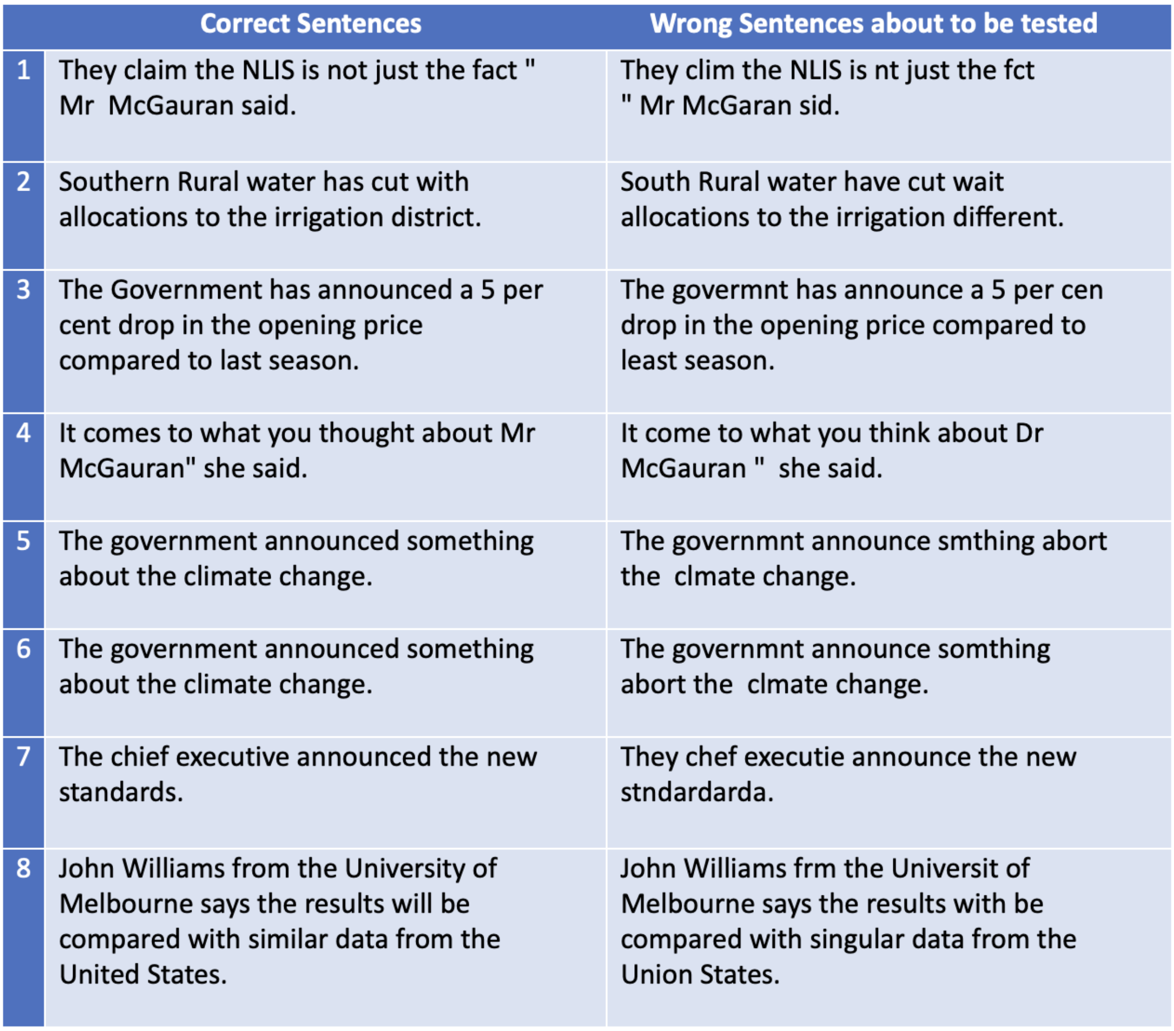
After the last iteration ('<e>'), we choose the best route (with the minimum aforementioned (*)) by following it reversely, choosing the candidate word per observed word that reduces the (*) sum the most, in order to find the full sequence, which is the final prediction.

The number of candidate words the algorithm keeps in every iteration can, of course, vary, hence we chose 3 arbitrarily. This number selection may as well differentiate significantly the prediction of the algorithm, potentially improving or deteriorating its performance.

It is particularly worth-mentioning that at first, lists were used for the creation of the spelling corrector. However, in order to optimize its efficiency and speed, we proceeded to switching from the use of lists to the use of arrays instead. This led in an impressive decrease in the time needed for the spelling corrector to make predictions from approximately 4 minutes to not even a single second. Along with that, small numerical differences in the sum(log(probabilities)) values were also noticed, that did not have, however, any important effect on the corrector's total performance.

To maximize the performance of the spelling corrector, we used a development set of 20 sentences, from which we randomly selected 7, twisted them by replacing specific words with optically similar ones that both may (Type II error) or may not be (Type I error) in the vocabulary we have established in Question 1, using the train_set. We also erased random letters from some words causing spelling errors (Type I error) to make the development set sentences even more wrong. Through the aforementioned procedure while checking the prediction for each flawed sentence for many different λ values, we were able to hyper-tune the λ_1, λ_2 parameters, choosing in the end 1 and 5 as their values respectively.

Indicatively, below some of these tested flawed sentences will be presented and further delved into. Note that some of the non-flawed sentences may not make clear sense, since they are twisted and also most of the times derive from the context of a dialogue. However, they are grammatically correct.



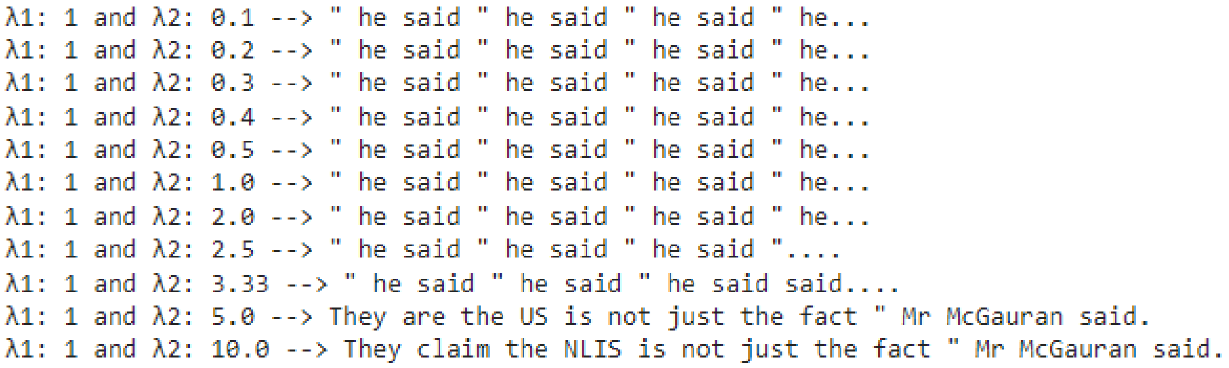
	Correct Sentences	Wrong Sentences about to be tested
1	They claim the NLIS is not just the fact " Mr McGauran said.	They clim the NLIS is nt just the fct " Mr McGaran sid.
2	Southern Rural water has cut with allocations to the irrigation district.	South Rural water have cut wait allocations to the irrigation different.
3	The Government has announced a 5 per cent drop in the opening price compared to last season.	The govermnt has announce a 5 per cen drop in the opening price compared to least season.
4	It comes to what you thought about Mr McGauran" she said.	It come to what you think about Dr McGauran " she said.
5	The government announced something about the climate change.	The governmnt announce smthing abort the clmate change.
6	The government announced something about the climate change.	The governmnt announce somthing abort the clmate change.
7	The chief executive announced the new standards.	They chef executie announce the new stndardarda.
8	John Williams from the University of Melbourne says the results will be compared with similar data from the United States.	John Williams frm the Universit of Melbourne says the results with be compared with singular data from the Union States.

The following outputs include predictions for λ_1=1 and λ_2 with values in [0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 2.5, 3.33, 5, 10].

(1) The 1st sentence contains the following mistakes:

clim <-- claim, nt <-- not, fct <-- fact, McGaran <-- McGauran, sid <-- said

All the mistakes are of Type I, they are misspelled words, which means that they do not belong to the vocabulary.



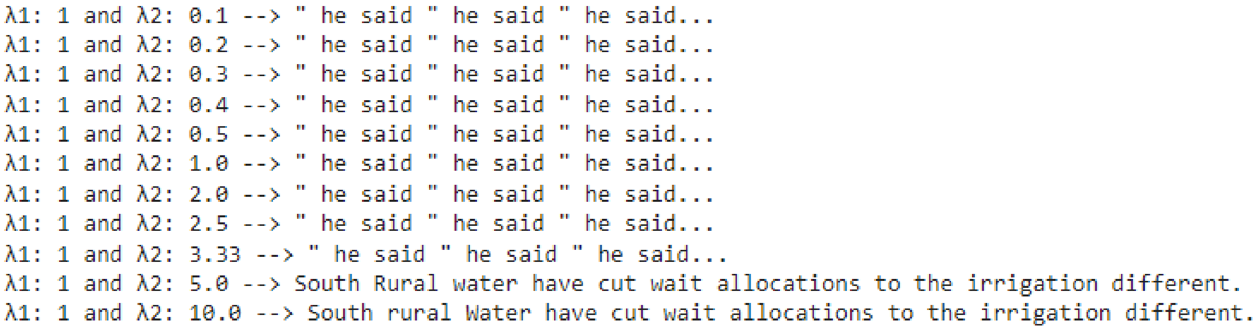
```
λ1: 1 and λ2: 0.1 --> " he said " he said " he said " he...
λ1: 1 and λ2: 0.2 --> " he said " he said " he said " he...
λ1: 1 and λ2: 0.3 --> " he said " he said " he said " he...
λ1: 1 and λ2: 0.4 --> " he said " he said " he said " he...
λ1: 1 and λ2: 0.5 --> " he said " he said " he said " he...
λ1: 1 and λ2: 1.0 --> " he said " he said " he said " he...
λ1: 1 and λ2: 2.0 --> " he said " he said " he said " he...
λ1: 1 and λ2: 2.5 --> " he said " he said " he said "....
λ1: 1 and λ2: 3.33 --> " he said " he said " he said said....
λ1: 1 and λ2: 5.0 --> They are the US is not just the fact " Mr McGauran said.
λ1: 1 and λ2: 10.0 --> They claim the NLIS is not just the fact " Mr McGauran said.
```

It is clear that values λ_1=1 and λ_2=10 predict perfectly the given sentence. However, the combination λ_1=1, λ_2=5 corrects 4 out of the 5 initial mistakes and at the same time replaces NLIS which is a correct abbreviation with a more common one, US.

(2) The 2nd sentence contains only mistakes of Type II, which means that the wrong words are such that also belong to the established vocabulary.

More specifically:

South <-- Southern , have <-- has , wait <-- with , different <-- district

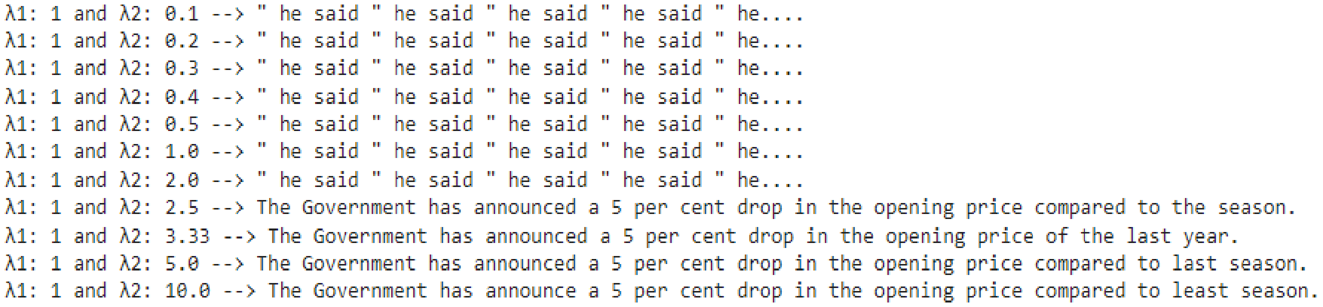


```
λ1: 1 and λ2: 0.1 --> " he said " he said " he said...
λ1: 1 and λ2: 0.2 --> " he said " he said " he said...
λ1: 1 and λ2: 0.3 --> " he said " he said " he said...
λ1: 1 and λ2: 0.4 --> " he said " he said " he said...
λ1: 1 and λ2: 0.5 --> " he said " he said " he said...
λ1: 1 and λ2: 1.0 --> " he said " he said " he said...
λ1: 1 and λ2: 2.0 --> " he said " he said " he said...
λ1: 1 and λ2: 2.5 --> " he said " he said " he said...
λ1: 1 and λ2: 3.33 --> " he said " he said " he said...
λ1: 1 and λ2: 5.0 --> South Rural water have cut wait allocations to the irrigation different.
λ1: 1 and λ2: 10.0 --> South rural Water have cut wait allocations to the irrigation different.
```

As it is obvious, none of the mistakes were corrected in the predictions, no matter whether the bigram or the Levenshtein log(probabilities) have a significantly larger weight. This practically leads us to think that the corrector we designed does not work well for the errors of type II. This might also have to do with the corpus that we have chosen to create our vocabulary with.

(3) The 3rd sentence contains both errors of Type I and Type II.

govermnt <-- government , announce <-- announced , cen <-- cent , least <-- last



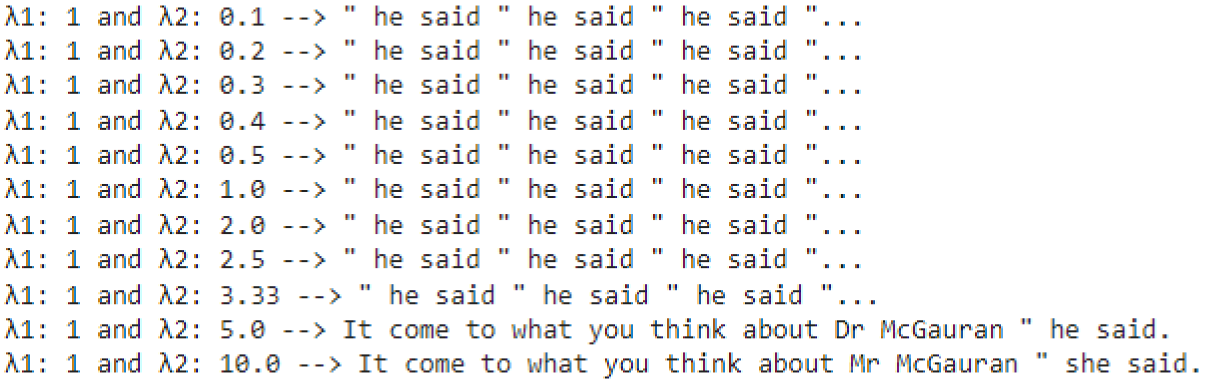
```
λ1: 1 and λ2: 0.1 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 0.2 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 0.3 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 0.4 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 0.5 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 1.0 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 2.0 --> " he said " he said " he said " he said " he....
λ1: 1 and λ2: 2.5 --> The Government has announced a 5 per cent drop in the opening price compared to the season.
λ1: 1 and λ2: 3.33 --> The Government has announced a 5 per cent drop in the opening price of the last year.
λ1: 1 and λ2: 5.0 --> The Government has announced a 5 per cent drop in the opening price compared to last season.
λ1: 1 and λ2: 10.0 --> The Government has announce a 5 per cent drop in the opening price compared to least season.
```

It is apparent that the combination λ_1=1, λ_2=5 predicts the sentence flawlessly. On the contrary, combination λ_1=1, λ_2=10 only corrects 2 out of the 4 existing errors, which also happen to be errors of Type I. This is another confirmation that the spelling corrector we created only works well enough for this type of errors and unfortunately ignores those of Type II.

(4) The 4th sentence also contains both types of errors.

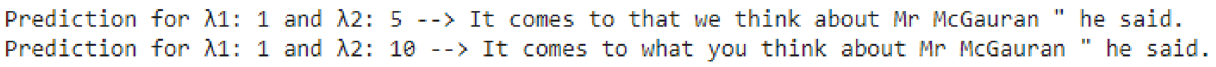
More particularly:

come <-- comes , think <-- thought , Dr <-- Mr



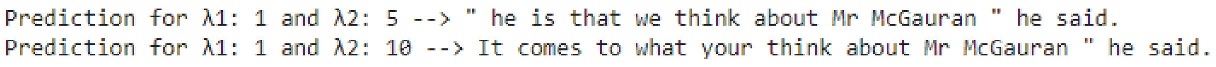
```
λ1: 1 and λ2: 0.1 --> " he said " he said " he said "...
λ1: 1 and λ2: 0.2 --> " he said " he said " he said "...
λ1: 1 and λ2: 0.3 --> " he said " he said " he said "...
λ1: 1 and λ2: 0.4 --> " he said " he said " he said "...
λ1: 1 and λ2: 0.5 --> " he said " he said " he said "...
λ1: 1 and λ2: 1.0 --> " he said " he said " he said "...
λ1: 1 and λ2: 2.0 --> " he said " he said " he said "...
λ1: 1 and λ2: 2.5 --> " he said " he said " he said "...
λ1: 1 and λ2: 3.33 --> " he said " he said " he said "...
λ1: 1 and λ2: 5.0 --> It come to what you think about Dr McGauran " he said.
λ1: 1 and λ2: 10.0 --> It come to what you think about Mr McGauran " she said.
```

In this case, it is evident that the mistakes we planted in the tested sentence are not as important as in the previous examples. Thus, we notice that only 1 of 3 are corrected in the combination λ_1=1, λ_2=5. Notice though, that “she” has also turned to “he”, which was not needed, perhaps because the bigram (he, said) is more common than (she, said) (notice that all other combinations return predictions containing only “ he said). However, it is a change that does not really affect the meaning of the sentence. At this point, it is important to mention that difficult sentence-cases like this one may stand a better chance at being predicted more correctly, if we choose to keep more than 3 candidate words per iteration. Indicatively, if we choose 15 candidate words per iteration, combination λ_1=1, λ_2=10 predicts the sentence almost perfectly and it only changes again “she” to “he”:



```
Prediction for λ1: 1 and λ2: 5 --> It comes to that we think about Mr McGauran " he said.
Prediction for λ1: 1 and λ2: 10 --> It comes to what you think about Mr McGauran " he said.
```

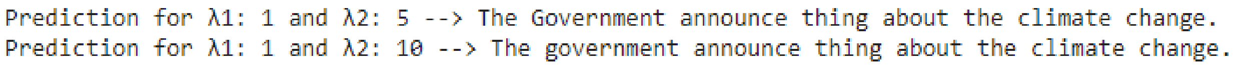
If we increase the number of candidate words even more though, for example to 20, “you” turns falsely to “your”. So, increasing the number a lot does not always provide us with the best prediction possible.



```
Prediction for λ1: 1 and λ2: 5 --> " he is that we think about Mr McGauran " he said.
Prediction for λ1: 1 and λ2: 10 --> It comes to what your think about Mr McGauran " he said.
```

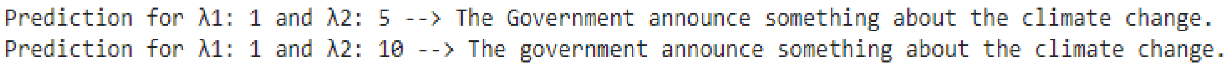
Continuing with the number of candidate words now to 3 again, in order to support even more the argument that the corrector works way better for Type I errors and more specifically, the Levenshtein distance of an observed word plays a more crucial role to the prediction than the frequency (probability) of a bigram, always in accordance to the content of the corpus we have chosen to work with, the following example is sufficiently explanatory.

In the tested sentences (5) and (6) the only difference is that in (5) there is “smthing” instead of “somthing”. Note that the Levenshtein distance of “smthing” from “something” is 2 and from “thing” is also 2. Therefore, “something” is not predicted.



```
Prediction for λ1: 1 and λ2: 5 --> The Government announce thing about the climate change.
Prediction for λ1: 1 and λ2: 10 --> The government announce thing about the climate change.
```

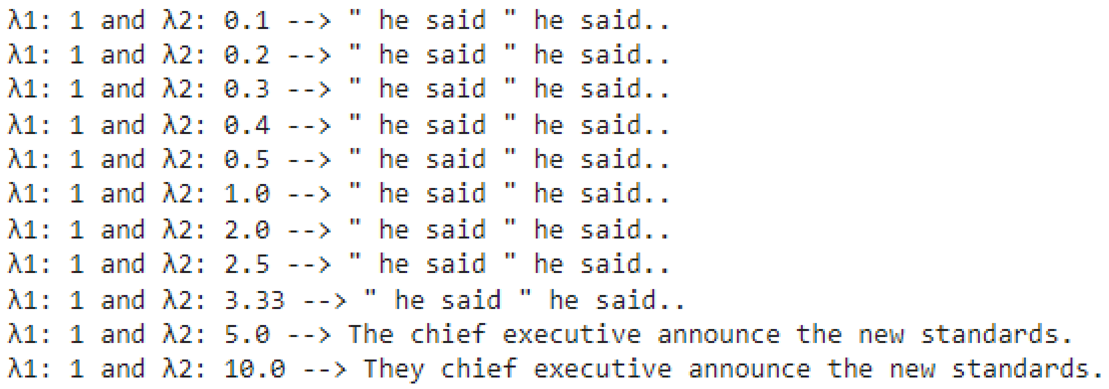
On the contrary, ”somthing”, which has Levenshtein distance from ”something” equal to 1 and from “thing“ equal to 3, is predicted flawlessly as ”something”, since the Levenshtein distance is smaller.



```
Prediction for λ1: 1 and λ2: 5 --> The Government announce something about the climate change.
Prediction for λ1: 1 and λ2: 10 --> The government announce something about the climate change.
```

(7) In the 7th sentence, there at least two errors for both types. Actually:

They <-- The, chef <-- chief, executie <-- executive, announce <-- announced, stndardarda <-- standards

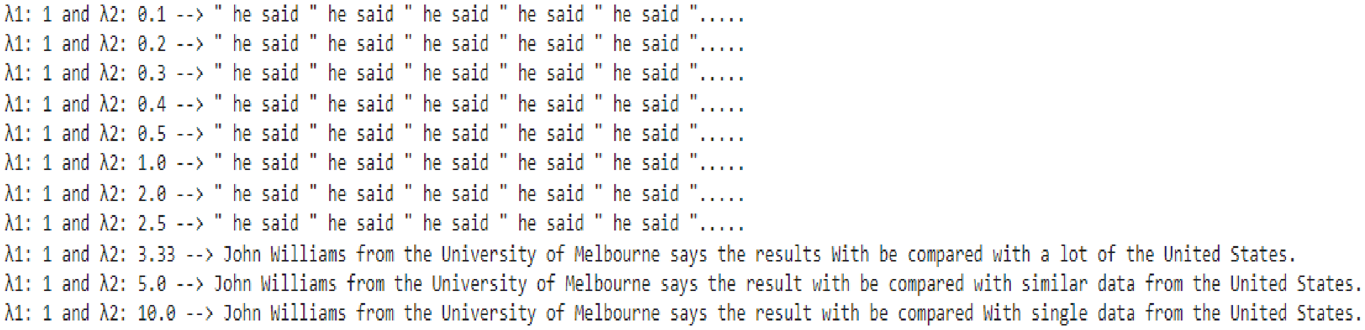


```
λ1: 1 and λ2: 0.1 --> " he said " he said..
λ1: 1 and λ2: 0.2 --> " he said " he said..
λ1: 1 and λ2: 0.3 --> " he said " he said..
λ1: 1 and λ2: 0.4 --> " he said " he said..
λ1: 1 and λ2: 0.5 --> " he said " he said..
λ1: 1 and λ2: 1.0 --> " he said " he said..
λ1: 1 and λ2: 2.0 --> " he said " he said..
λ1: 1 and λ2: 2.5 --> " he said " he said..
λ1: 1 and λ2: 3.33 --> " he said " he said..
λ1: 1 and λ2: 5.0 --> The chief executive announce the new standards.
λ1: 1 and λ2: 10.0 --> They chief executive announce the new standards.
```

In this case, the combination λ_1=1, λ_2=5 gives an almost perfect prediction, with the only persistent error being that it kept “announce” as it is. The combination λ_1=1, λ_2=10 also does not correct “They”, hence it gives a worse prediction.

(8) Lastly, in the 8th sentence, we chose again to put both types of errors. These errors are:

Frm <-- from, Universit <-- University, with <-- will, singular <-- similar, Union <-- United

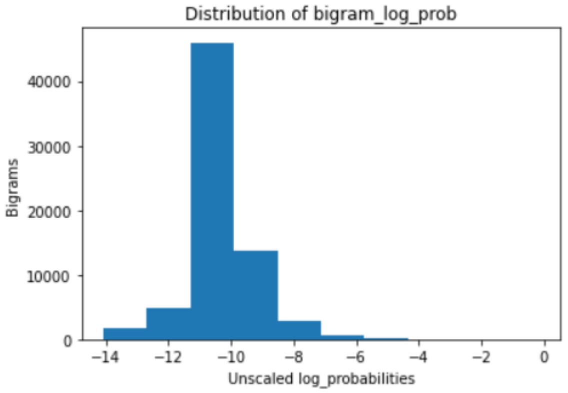


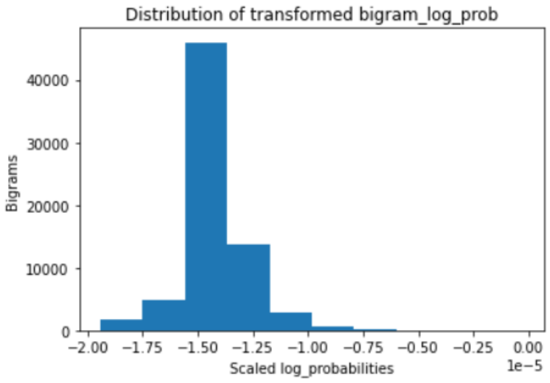
```
λ1: 1 and λ2: 0.1 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 0.2 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 0.3 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 0.4 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 0.5 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 1.0 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 2.0 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 2.5 --> " he said " he said " he said " he said " he said ".....
λ1: 1 and λ2: 3.33 --> John Williams from the University of Melbourne says the results With be compared with a lot of the United States.
λ1: 1 and λ2: 5.0 --> John Williams from the University of Melbourne says the result with be compared with similar data from the United States.
λ1: 1 and λ2: 10.0 --> John Williams from the University of Melbourne says the result with be compared With single data from the United States.
```

Here, the best combination is again λ_1=1, λ_2=5, since it gives a prediction with only one mistake, “with” instead of “will”. The combination λ_1=1, λ_2=10 has the same error, but it also replaces “singular” with “single”, which is incorrect.

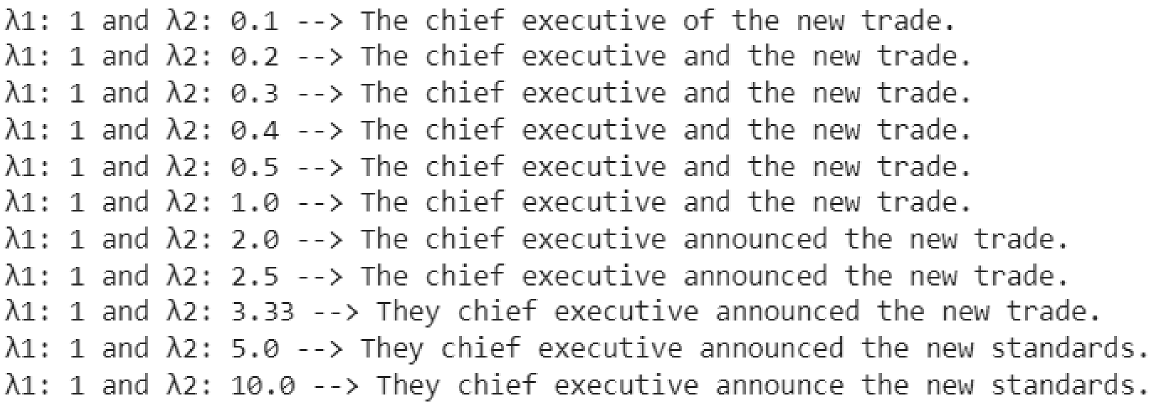
In conclusion, one can say with confidence that the corpus used to test the context-aware spelling corrector is not ideal, since some bigrams have a significantly larger probability (frequency) than others (e.g. [he, said] or [“, he]). This has probably caused the corrector to return feasible-friendly predictions only for combinations, in which the weight given to the Levenshtein part of the formula (*) is greatly larger than the weight given to the bigrams’ part, which, of course, means that the bigrams end up not affecting the predictions significantly. Thus, as it was mentioned in the beginning, we eventually fine-tune λ_1, λ_2 values to 1 and 5, respectively, since this combination’s predictions averagely suffice more.

In an attempt to improve our spelling corrector’s performance, we tried scaling the bigram and the Levenshtein log(probabilities). We did that by dividing each existing bigram log probability with the total negative bigram sum. In the same way, we scaled the non-existing bigrams’ log probabilities. Furthermore, we followed a similar approach for the Levenstein log probabilities. We can see below the distribution of the log probabilities of the existing bigrams before and after the transformation.





The algorithm is the same from now on, but the log probabilities are now comparable. Unfortunately, the new approach did not meet the expectations. In most cases it fixed some of the mistakes which were not detected in the previous approach, but it also did not detect some that were. This, of course, might have to do with the nature of the corpus we trained our models. For example, these are the results of the 7th sentence:



```
λ1: 1 and λ2: 0.1 --> The chief executive of the new trade.
λ1: 1 and λ2: 0.2 --> The chief executive and the new trade.
λ1: 1 and λ2: 0.3 --> The chief executive and the new trade.
λ1: 1 and λ2: 0.4 --> The chief executive and the new trade.
λ1: 1 and λ2: 0.5 --> The chief executive and the new trade.
λ1: 1 and λ2: 1.0 --> The chief executive and the new trade.
λ1: 1 and λ2: 2.0 --> The chief executive announced the new trade.
λ1: 1 and λ2: 2.5 --> The chief executive announced the new trade.
λ1: 1 and λ2: 3.33 --> They chief executive announced the new trade.
λ1: 1 and λ2: 5.0 --> They chief executive announced the new standards.
λ1: 1 and λ2: 10.0 --> They chief executive announce the new standards.
```

The combination of λ_1=1, λ_2=5, which prior of the scaling, as we concluded, was the best, in this case, even if it corrects “announce” to “announced”, it creates another spelling mistake, “the” to “they”.

Manner of working on the assignment:

The presented assignment was processed with the equal participation of all team members/contributors. The team worked together in group Discord calls, during which the understanding and structure of the assignment were discussed and cleared upon. Subsequently, more Discord calls took place with one member each time sharing the screen, during which the code was created with the simultaneous attention/participation of all members. The code was each day at night worked upon or modified by each member individually, to eventually reach its final form.

The report was split into 4 parts and after the main body of the report was structured, some more group calls took place, in order to unify, correct and improve certain pieces of the report by adding missing elements and information.