CS4442 – Assignment 2

Dustin Dobransky 250575030

1. This question required us to compare n-grams between two texts.
   1. For Dostoyevsky’s works, the results were as follows:

|  |  |
| --- | --- |
| nGram size | Percentage of words in 2 not in 1 |
| 1 | 40.647 |
| 2 | 72.681 |
| 3 | 89.796 |
| 4 | 97.606 |
| 5 | 99.536 |
| 6 | 99.891 |

The largest common n-gram occurs at n = 17, with “repulsion that s what i m afraid of that s what may be too much for me”

* 1. For Dickens vs. Kafka’s works, the results were as follows:

|  |  |
| --- | --- |
| nGram size | Percentage of works in 2 not in 1 |
| 1 | 32.880 |
| 2 | 77.458 |
| 3 | 95.433 |
| 4 | 99.195 |
| 5 | 99.885 |
| 6 | 99.977 |

The largest common n-gram occurs at n = 7 with two n-grams:

1. “in the middle of the table and”
2. “there is no such thing as a”
   1. For “MarxEngelsManifest” vs. “SmithWealthNations”, the results were:

|  |  |
| --- | --- |
| nGram size | Percentage of works in 2 not in 1 |
| 1 | 84.185 |
| 2 | 97.468 |
| 3 | 99.534 |
| 4 | 99.920 |
| 5 | 99.987 |
| 6 | 99.998 |

The largest common n-grams occur at n = 6 with:

1. of nature and of reason the
2. is the same as that of
3. to keep up the rate of
4. in order to keep up the
5. of a man s own labour
6. from them what they have not
   1. Discussion:

From the results in a), it can be seen that books written from the same author will have a large number of similar n-grams. This is expected, as the author has a distinct writing style, which will be reflected in their works. In this case, similarity in writing style is identified by the recurrence of n-grams across different novels.

For the remaining two tests, both were authored by different people, and the differentiation between n-grams was large. Past 6-grams, the works did not contain significant overlap.

1. This problem involved determining the percentage of 0-probability sentenced in a test document given a training document. This was done by constructing a hash set of n-grams from the first document, and then generating a hash set for each sentence in the second document. Any of the n-grams from a sentence from the second document was not found to be an n-gram from the first document, then the probability of the sentence existing in the first document was 0. The results from these tests were as follows:

|  |  |
| --- | --- |
| n-gram size | 0-probability sentences [%] |
| 1 | 85.878 |
| 2 | 96.756 |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

1. This problem involved estimating sentences using a given text. Ngrams ranging from size 1 to n were generated from the text, and the probabilities of generating a next word from the entire vocabulary.
   1. For n = 1,2,3,4,6 the results were as follows:

|  |  |
| --- | --- |
| **n-gram size** | **Sentence** |
| 1 | the begun s you the was and <END> |
| 2 | the cover around his trial at k <END> |
| 3 | the cover of one who can go with you <END> |
| 4 | the cover of one of the people up on the window sill and went to the front entrance waited there in ambush and every time a lawyer tried to enter the building he would throw him down the steps <END> |
| 6 | the cover of one of them had nearly broken through in its middle and it was held together with a few threads <END> |

For n = 1, the results were gibberish. Without any context, the next generated word was just the next most frequent word. With n = 6, there was only one instance of the previous sentence existing, so only one word in the vocabulary had a probability of 1, and the rest 0. This resulted in an exact sentence from the text being generated.

* 1. For n = 3, running the program on MarxEngelsManifest, the result was:

“of nations and crusades <END>”

With little context, a 3-gram size was sufficient to generate a sentence that was actually found in MarxEngelsManifest.txt. This suggests that as more data is provided to the algorithm, larger n-grams can be used, and therefore more context can be generated, leading to better sentence generation.

1. The results for computing the probabilities for the sentence using Add-Delta were as follows:

|  |  |
| --- | --- |
| Program Params | Results |
| KafkaTrial.txt testFile.txt 1 1 1 | 0.00  0.00 |
| KafkaTrial.txt testFile.txt 2 1 1 | -261.77  -273.97 |
| KafkaTrial.txt testFile.txt 2 0.001 1 | -335.46  -411.81 |
| KafkaTrial.txt testFile.txt 3 0.001 1 | -1187.74  -1335.67 |

These results generally make sense. As more context was added (increasing n-gram size), a larger number of probabilities could be calculated for each sentence, which produced the larger and larger negative numbers (added log probabilities).

An attempt was made at getting good-turing working, but it was to no avail. Generated probabilities were way out of proportion, and I couldn’t debug what was going wrong.

1. In this problem, sentences from various languages were tested against language models for 6 different languages, and classified with the most probable language using the Add-Delta classification method. The following is the resulting error for the various test cases:
2. No delta

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 1 0 50 | 14.34 |
| P5 2 0 50 | 19.12 |
| P5 3 0 50 | 47.25 |

1. Delta = 0.05

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 1 0.05 50 | 13.07 |
| P5 2 0.05 50 | 1.04 |
| P5 3 0.05 50 | 0.56 |

1. Varying Delta

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 3 0.05 50 | 0.56 |
| P5 3 0.005 50 | 0.48 |
| P5 3 0.0005 50 | 0.64 |

1. Discussion:

With only ML classification, as seen in b), error increases drastically with larger n-gram sizes. With the addition of Add-Delta smoothing, the error significantly decreases. This is because add-delta takes into account unseen n-grams, which evens out the probability of an n-gram occurring in a particular language.

1. The following is the error found while using varying sentence lengths to determine the language that sentence belongs to.

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 2 0.05 10 | 22.86 |
| P5 2 0.05 50 | 1.04 |
| P5 2 0.05 100 | 0.16 |

The reason why the error decreases with greater sentence length is because there is more context available per sentence. With a larger amount of context, more n-grams will match with the n-grams from a particular language, leading to greater probability of the language being estimated correctly.

1. The following section repeats the tests from b-d, but uses only latin characters, and a vocabulary size of 26. The results are as follows:
   * 1. No delta

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 1 0 50 | 69.64 |
| P5 2 0 50 | 68.61 |
| P5 3 0 50 | 69.08 |

* + 1. Delta = 0.05

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 1 0.05 50 | 65.66 |
| P5 2 0.05 50 | 19.04 |
| P5 3 0.05 50 | 7.41 |

* + 1. Varying Delta

|  |  |
| --- | --- |
| Program Parameters | Error [%] |
| P5 3 0.05 50 | 7.41 |
| P5 3 0.005 50 | 5.42 |
| P5 3 0.0005 50 | 4.54 |

The performance during this round of tests was significantly poorer than the previous round. Since a smaller vocabulary size was used, and all characters were lower case (latin\_only = true), there was a smaller set of data for the Add-Delta language model algorithm to operate on. By reducing the volume of data available, the algorithm was less accurate. Context which could have been important in determining the language of a sentence would have been removed by making all letters lower case.

1. This problem dealt with sentence autocorrection. The results were as follows:
2. Results

|  |  |
| --- | --- |
| P6 trainHuge.txt textCheck.txt dictionary.txt 2 3 1 1 | it would love to her the story  you will read in the garden  hello from the top of the world  i will drink milk in the morning  i will read the story |
| P6 trainHuge.txt textCheck.txt dictionary.txt 2 3 0.1 1 | it would love to her the story  you will read in the garden  hello from the top of the world  i will drink milk in the morning  i will read the story |
| P6 trainHuge.txt textCheck.txt dictionary.txt 2 3 0.01 1 | it would love to her the story  you will read in the garden  hello from the top of the world  i will drink milk in the morning  i will read the story |

There may be an error in the add-delta probability computation which leads to the above results. However, this did not hinder computations in problem 4, and the results were all very reasonable, leading me to believe I have correctly implemented ML with Add-Delta probability calculations correctly. All output is the same, regardless of the delta value. The two most probable sentences that a human would generate would be: “it would love to her the story” and “I would love to hear the story”. For all three trials, the probabilities of each sentences are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Input parameters [n, threshold, delta, mode] | Prob of “I would love to her the story” | Probability of “it would love to her the story” | Probability of “I would love to hear the story” |
| 2 3 1 1 | -110.117 | -109.303 | -115.063 |
| 2 3 0.1 1 | -98.5482 | -97.7299 | -103.516 |
| 2 3 0.01 1 | -96.4739 | -95.6551 | -101.444 |

From these trials, it can be seen that the word ‘it’ has a greater probability than the word ‘hear’ in the context of the train file. Therefore, regardless of the delta value set, the generated sentence will always produce “it would love to her the story”.

1. Results

|  |  |
| --- | --- |
| P6 trainHuge.txt textCheck.txt dictionary.txt 1 3 0.01 1 | i would love to her the story  you will red in the garden  hello from the tp of the world  i will drink mlk in the morning  i will read they story |
| P6 trainHuge.txt textCheck.txt dictionary.txt 2 3 0.01 1 | it would love to her the story  you will read in the garden  hello from the top of the world  i will drink milk in the morning  i will read the story |
| P6 trainHuge.txt textCheck.txt dictionary.txt 3 3 0.01 1 | it would love to her the story  you will read in the garden  hello from the top of the world  i will drink milk in the morning  i will read the story |

As in B, the results were the same across the board, except for n-grams of length 1. In the case of unigrams, there was no context for whether one word would follow another, and so the probability of one word vs. another was not computed. There would have been no difference between “her the” and “hear the” because there was no probability for either of those bigrams to occur, since only unigrams were generated for the sentence.

1. Since the Good-Turing implementation never resulted in meaningful data, only Add Delta and Add One models are compared. Using DostoevskyIdiot.txt and DostoevskyKaramazov.txt, the results are as follows:

|  |  |
| --- | --- |
| Prob. Add Delta | Prob. Add One |
| -1289.61 | -779.14 |

In this experiment, the log probabilities of all n-grams were computed with respect to the training file, and summed. The overall probability sum can roughly estimate the effectiveness of the each model.

The intended goal for this problem was to implement multiple error corrections per sentence. i.e. an incorrect sentence: “it wold love to her the story” would be corrected to: “I would love to hear the story”. The approach was going to be similar to the implementation in 6), with the addition of checking several permutations over misspelled words. That is, for every word in the sentence not found in the training set, a new sentence would be generated with that word replaced by each of the words found using the Levenshtein distance function. This approach would not replace existing words that are correct, for example “your have” vs. “you have” since all words are valid (assuming they exist in the training file).

For each of the newly generated sentences, the next incorrect word would be replaced by all words within 1 Levenshtein distance. This would generate X0\*X1\*…\*Xn different sentences, where X represents the number of words within 1 Levenshtein distance of the incorrect word, and n is the number of incorrect words in a sentence. After generating these sentences, the implementation found in 6 would be run on all of them to find the best sentence.