**2019510094 - Elif Özker**

**Algorithm Definition**

**Functions/Procedures:**

1. **readFile(File file)**:
   * **Inputs**: Takes a **File** object representing the text file to be read.
   * **Procedure**: Reads the file, tokenizes it into words, removes punctuation, and returns a list of all the words in the file.
   * **Output**: Returns a **List<String>** containing all the words.
2. **groupWordsOfNGram(int n, List<String> allSplittedWords)**:
   * **Inputs**: Takes an integer **n** representing the desired n-gram size and a **List<String>** of words.
   * **Procedure**: Generates n-grams of size **n** from the given list of words.
   * **Output**: Returns a **List<String>** containing the generated n-grams.
3. **concat(int start, int end, List<String> allSplittedWords)**:
   * **Inputs**: Takes integers **start** and **end** specifying the range of words and a **List<String>** of words.
   * **Procedure**: Concatenates words within the specified range into an n-gram.
   * **Output**: Returns a concatenated string representing the n-gram.
4. **findFrequencyMap(List<String> words)**:
   * **Inputs**: Takes a **List<String>** of words.
   * **Procedure**: Creates a frequency map where keys are unique words, and values are the corresponding word frequencies.
   * **Output**: Returns a **Map<String, Integer>** representing the frequency map.
5. **calculateProbability(Map<String, Integer> mp, int n, List<String> allSplittedWords)**:
   * **Inputs**: Takes a frequency map (**mp**), an integer **n** specifying the n-gram size, and a **List<String>** of words.
   * **Procedure**: Calculates and prints the probability of each n-gram based on its frequency and the frequency of its preceding words.
   * **Output**: Returns the probability as a **float**.
6. **ngramCalculation(List<String> allSplittedWords)**:
   * **Inputs**: Takes a **List<String>** of words.
   * **Procedure**: Calls **printNgramResults** for unigram, bigram, and trigram calculations and prints the total count of words.
   * **Output**: Prints the n-gram results and total running time.
7. **printNgramResults(String ngramType, int n, List<String> allSplittedWords)**:
   * **Inputs**: Takes a string **ngramType** representing the type of n-gram, an integer **n** specifying the n-gram size, and a **List<String>** of words.
   * **Procedure**: Prints the n-gram type, calls **calculateProbability**, and prints the total running time.
   * **Output**: Prints the n-gram results and total running time.
8. **main(String[] args)**:

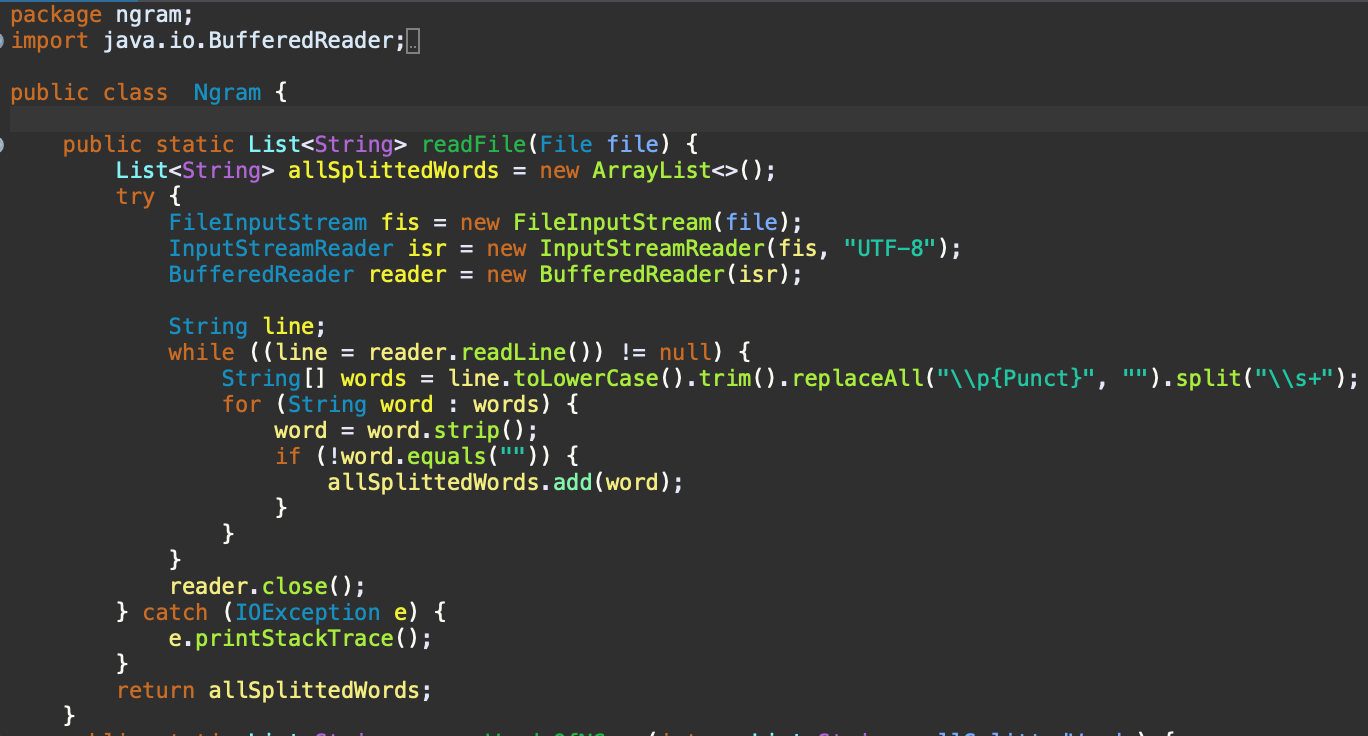
**Procedure**: Reads four sample novel files, performs n-gram analysis on each, and prints the results.

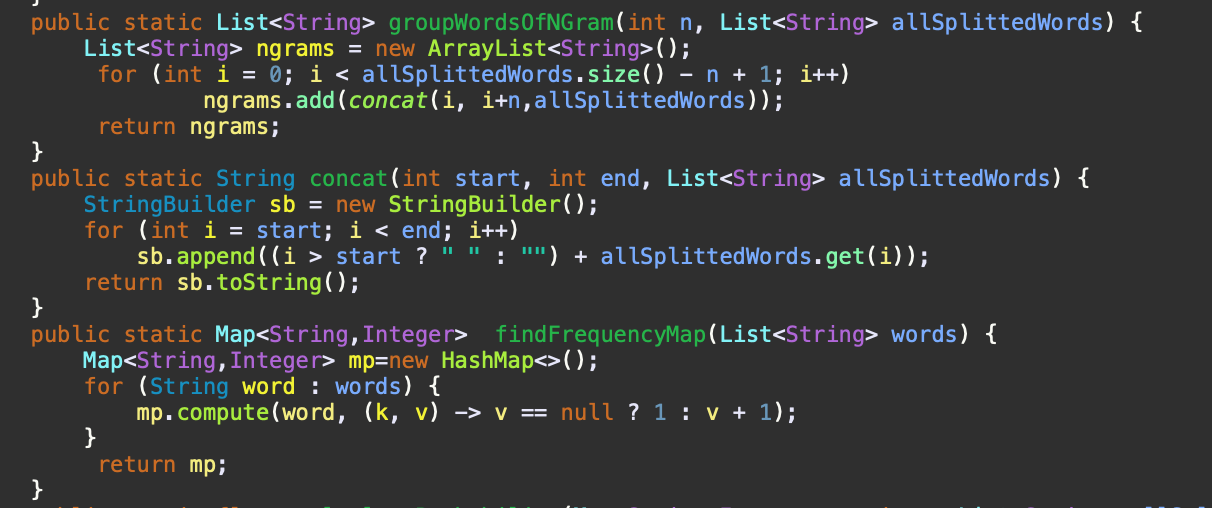
* + **Output**: Prints n-gram analysis results for each novel.

**Outputs:**

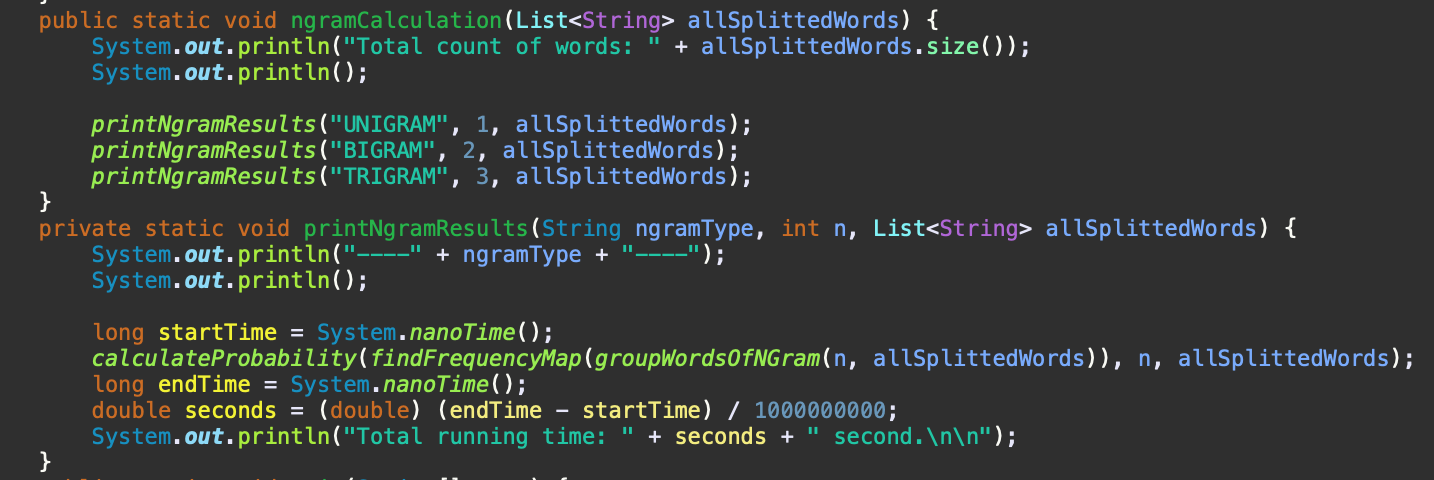
* The program prints the total count of words and the top 20 n-grams for unigrams, bigrams, and trigrams in each novel.
* It also prints the total running time for each novel.

**Screenshots**





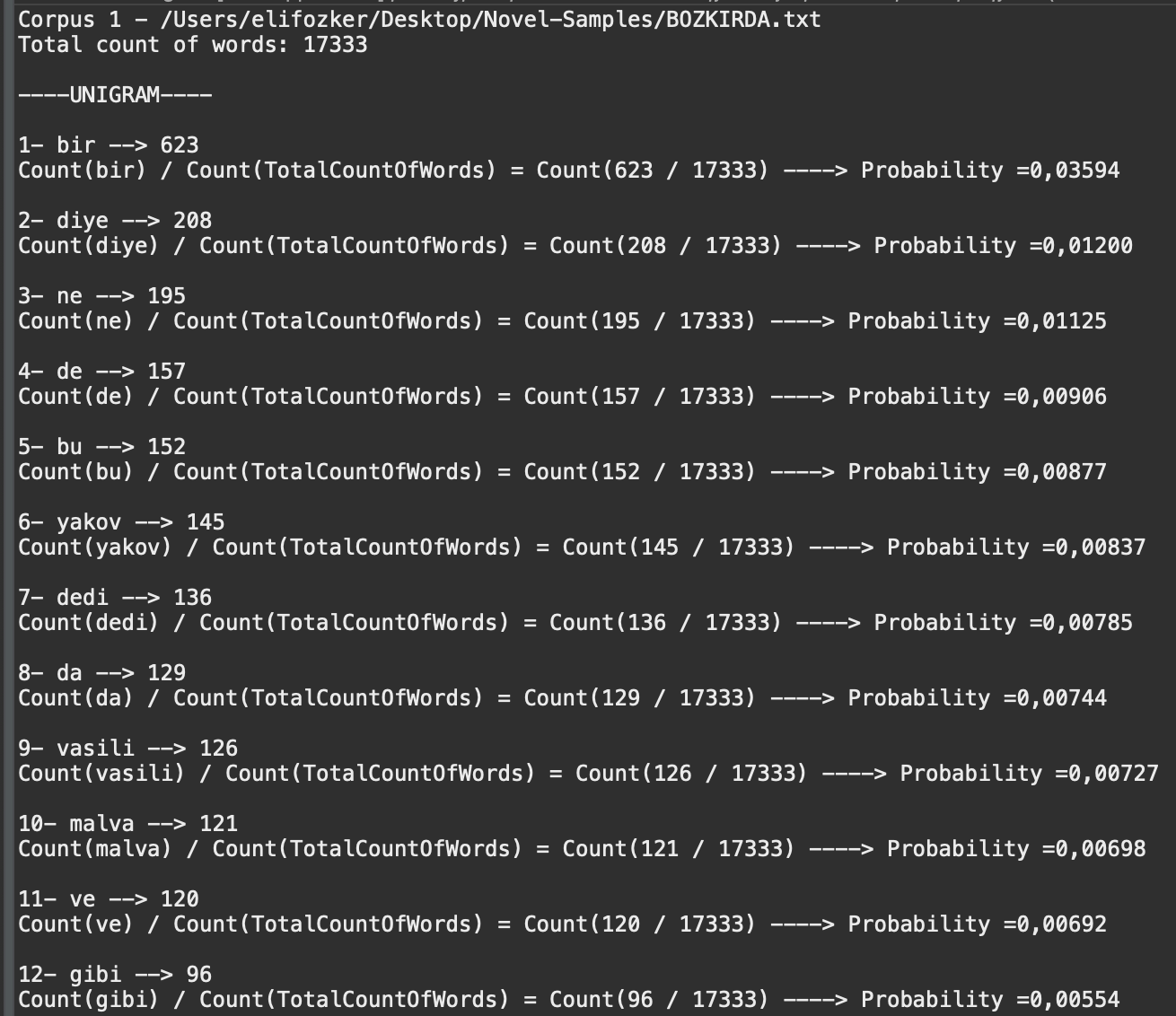


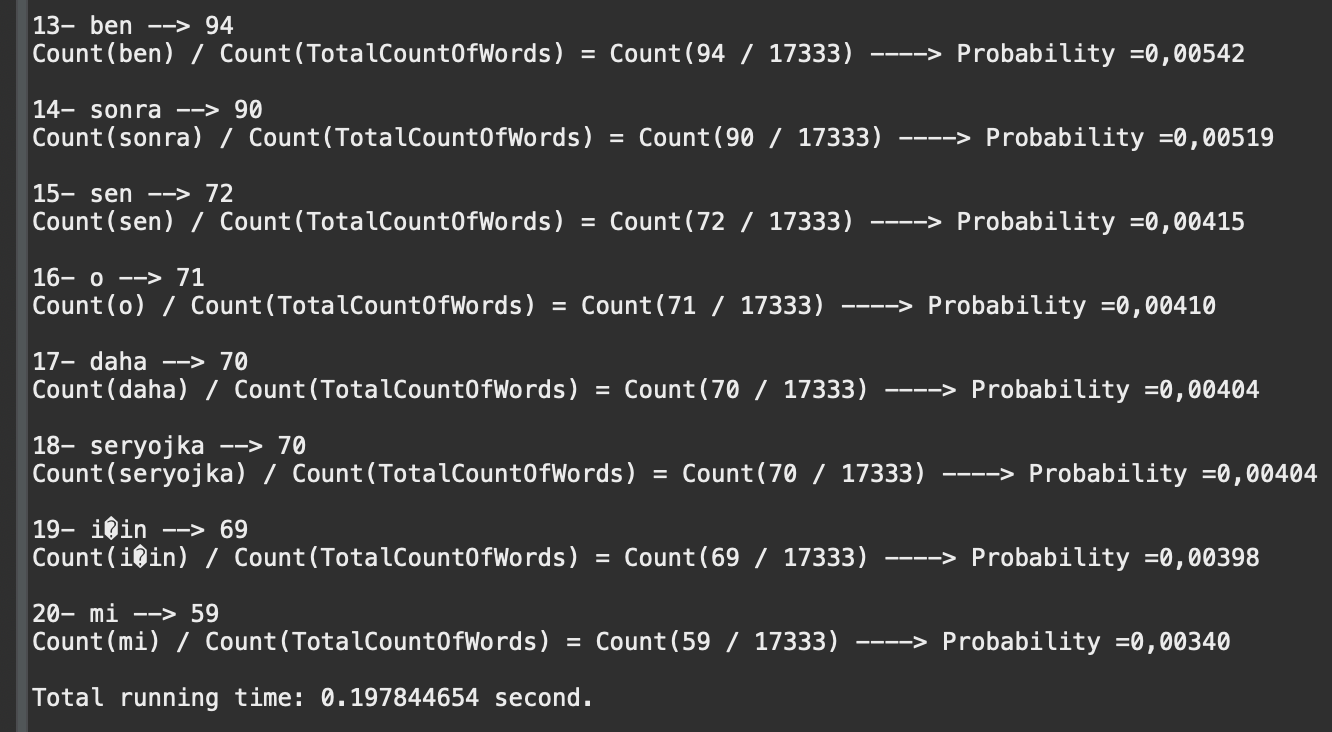


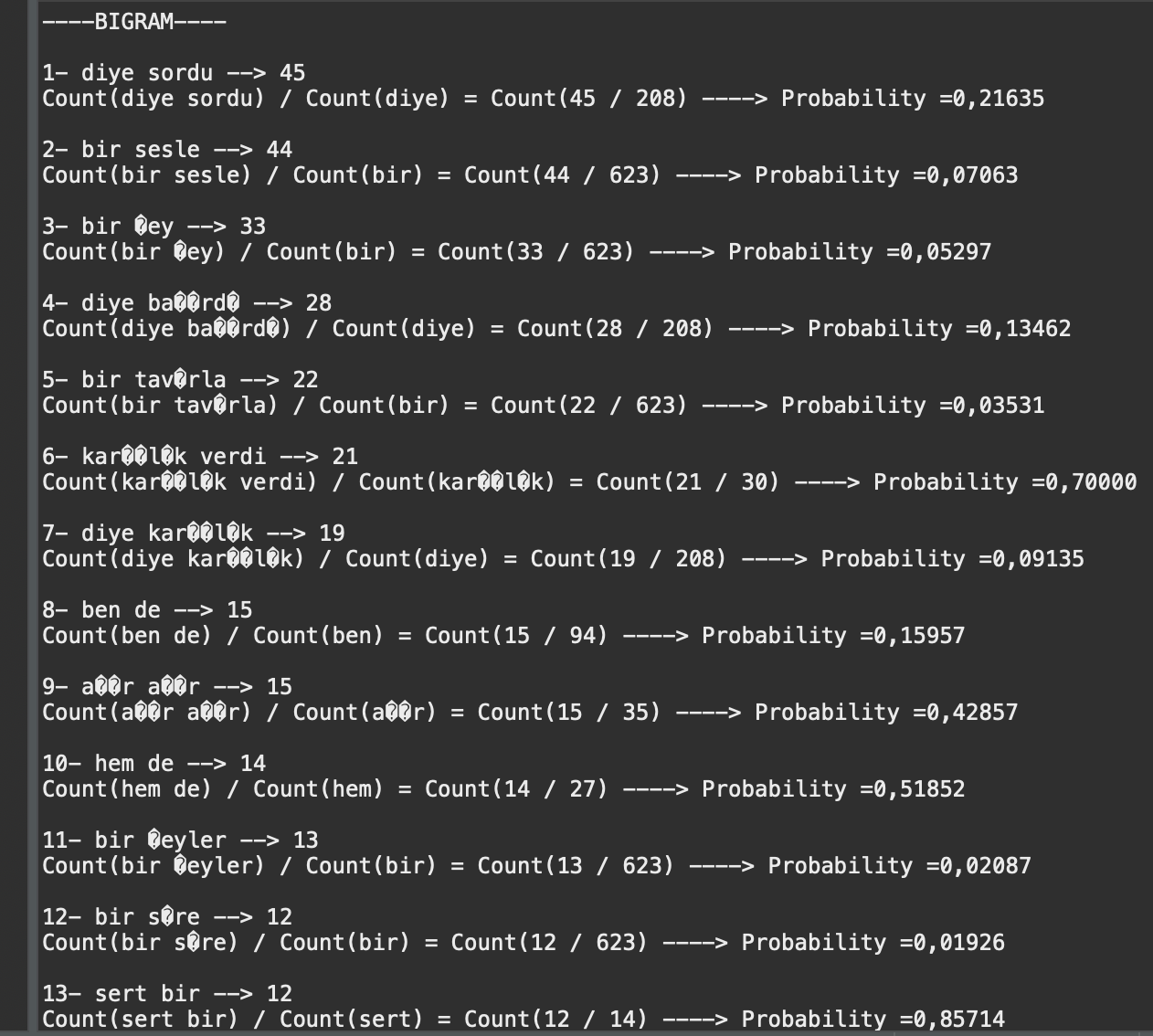


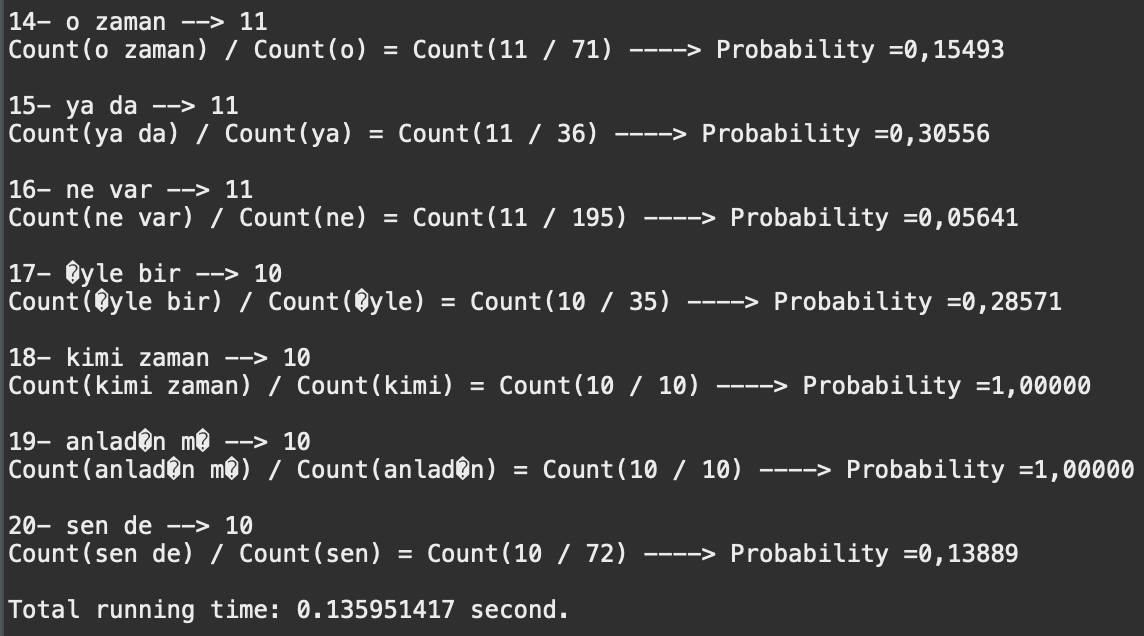
**OUTPUTS:**

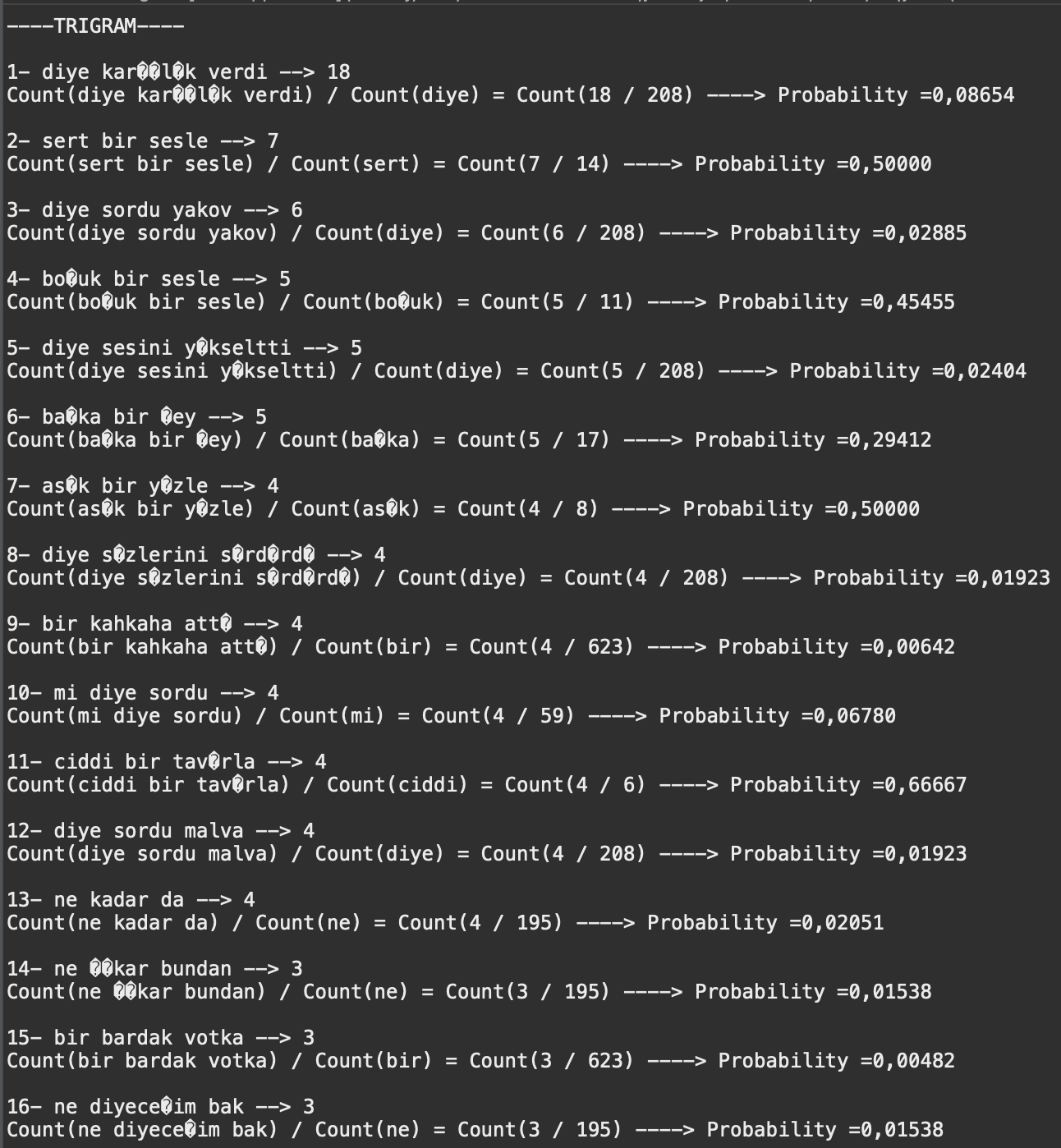
**BOZKIRDA.TXT**

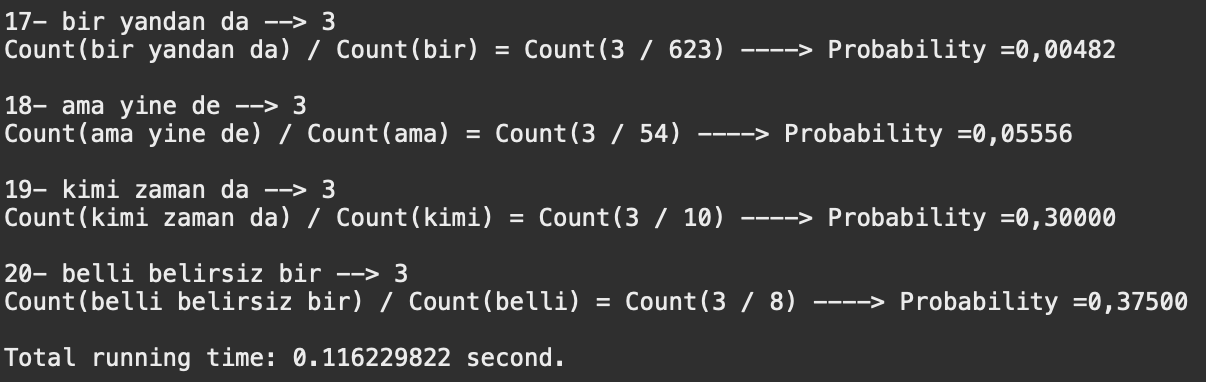




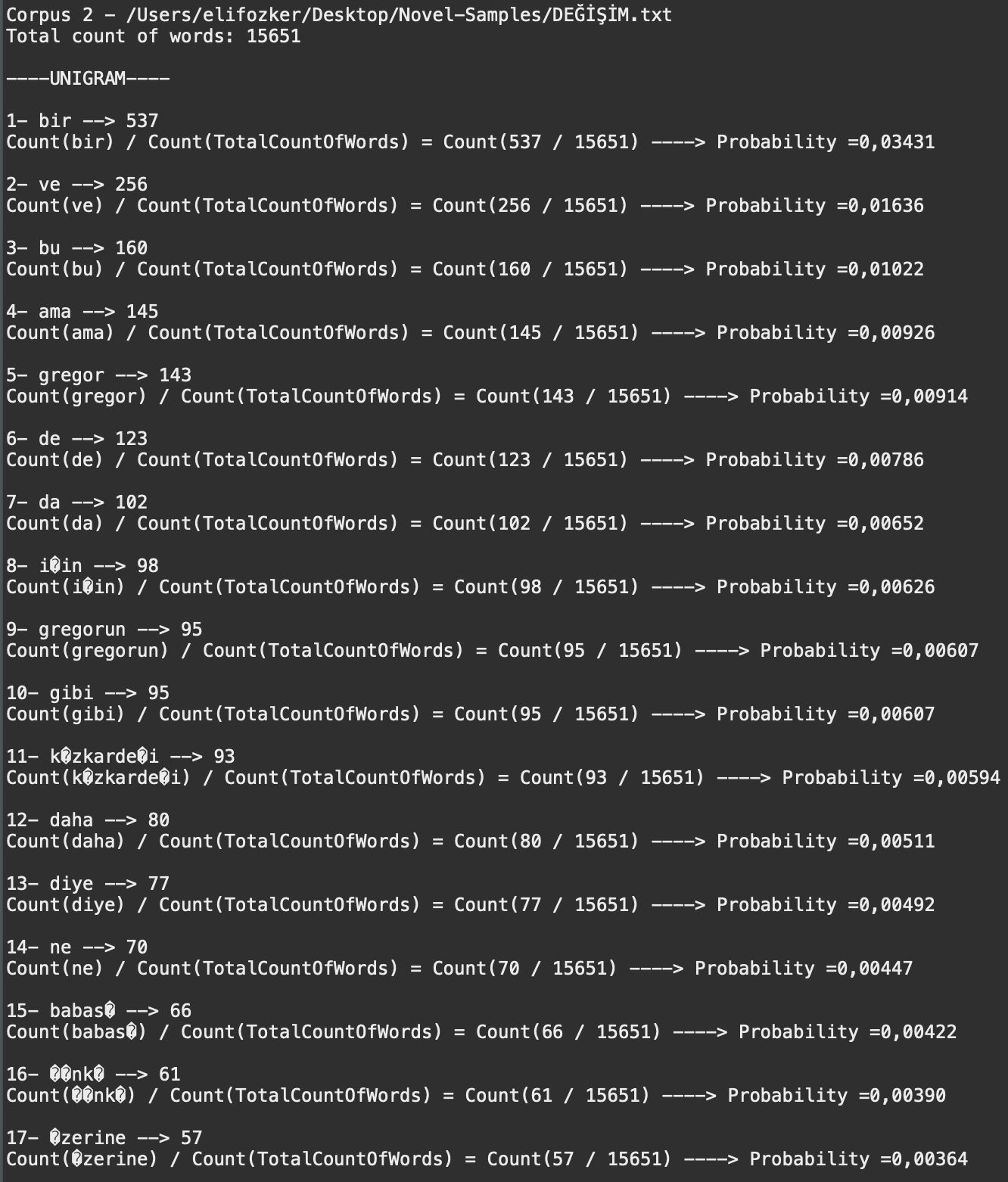


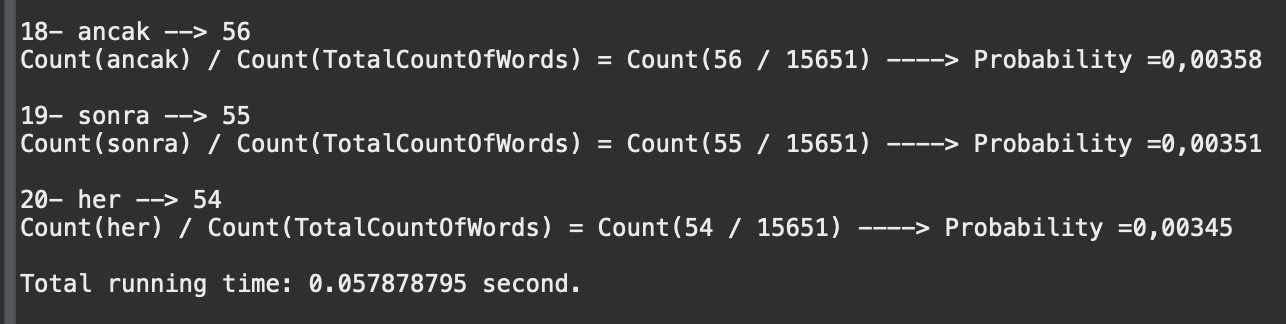


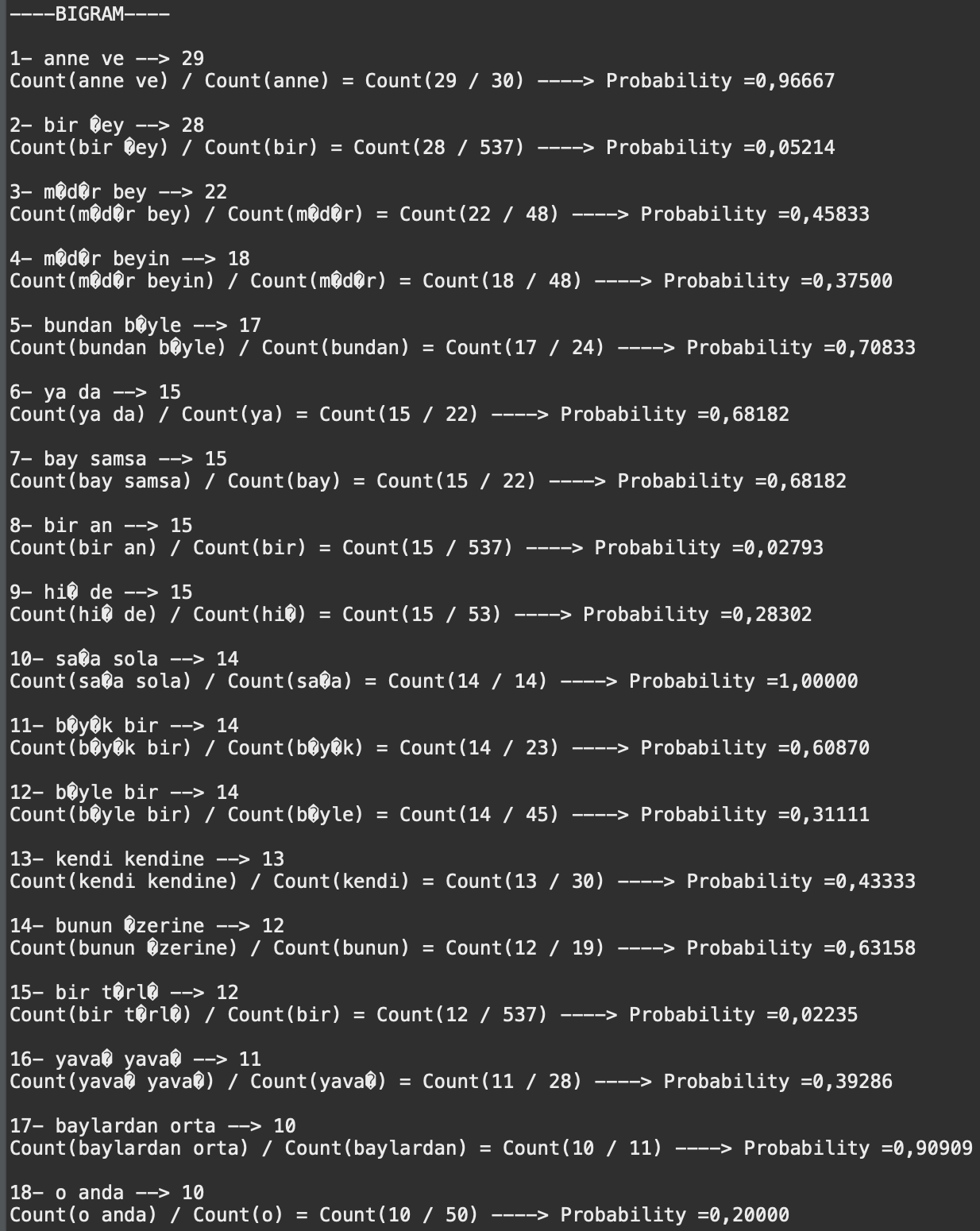


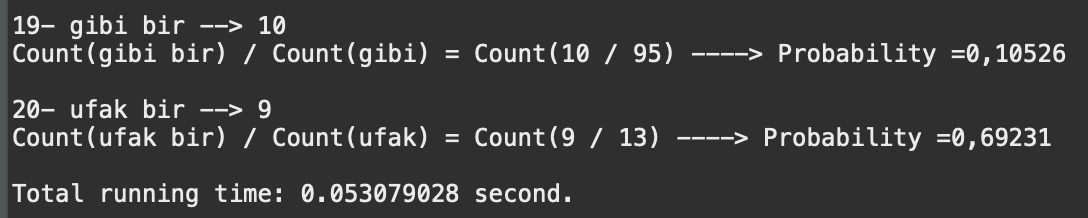


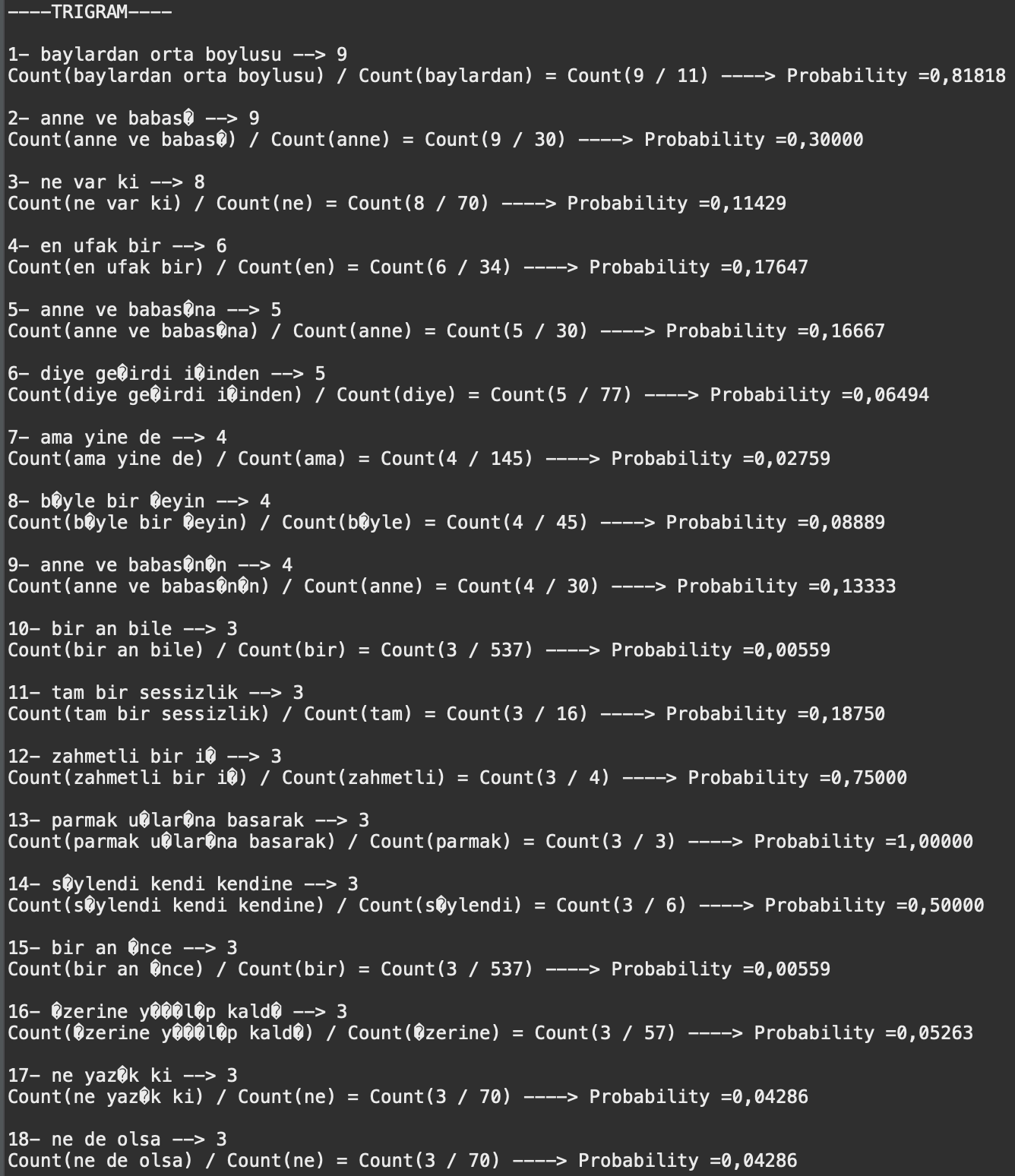
**DEĞİŞİM.TXT**

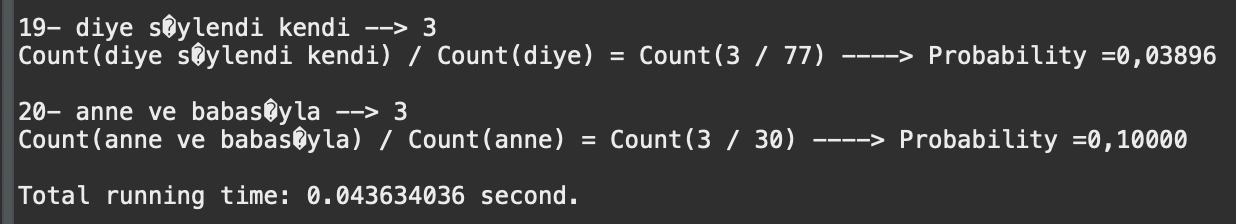




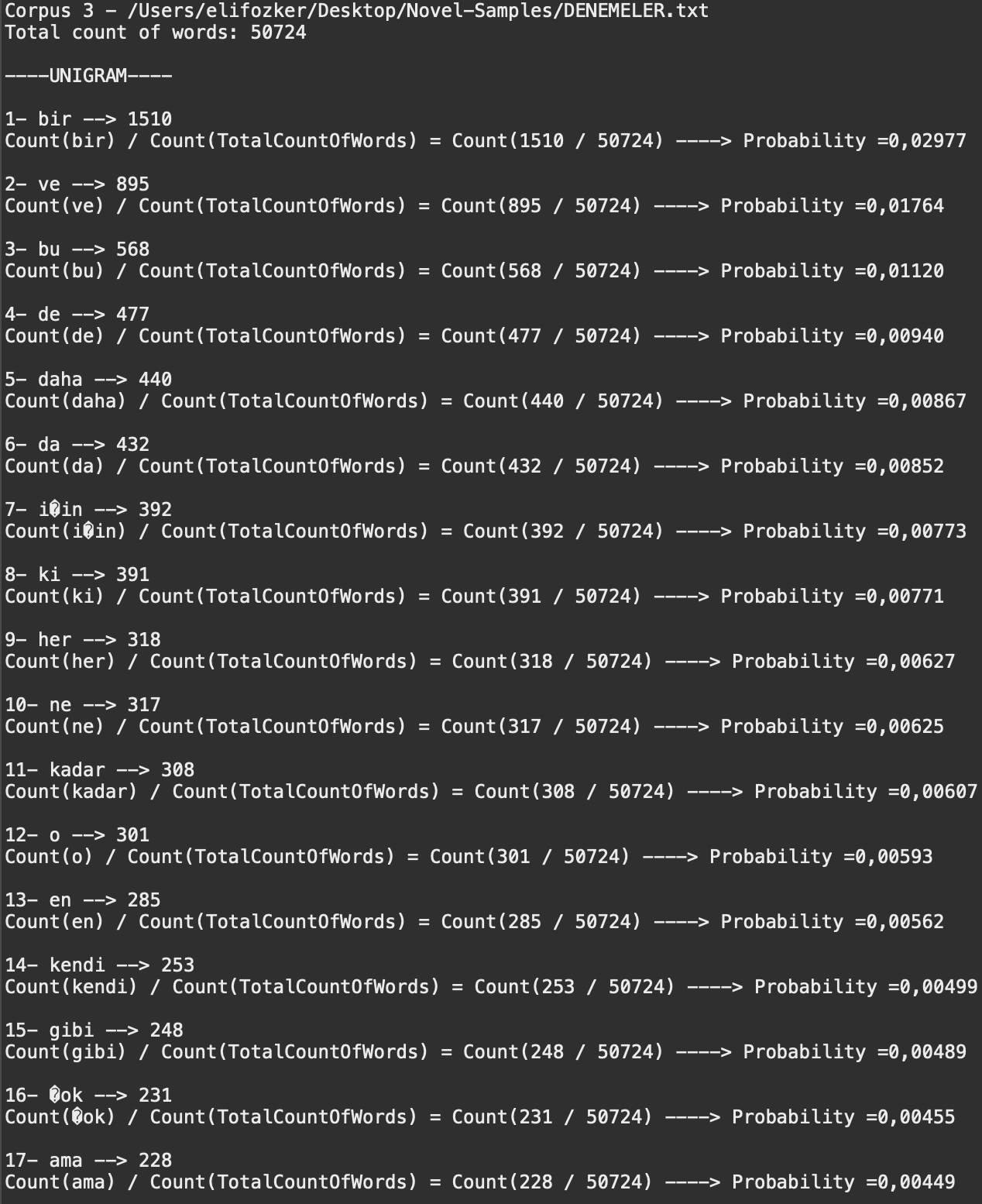


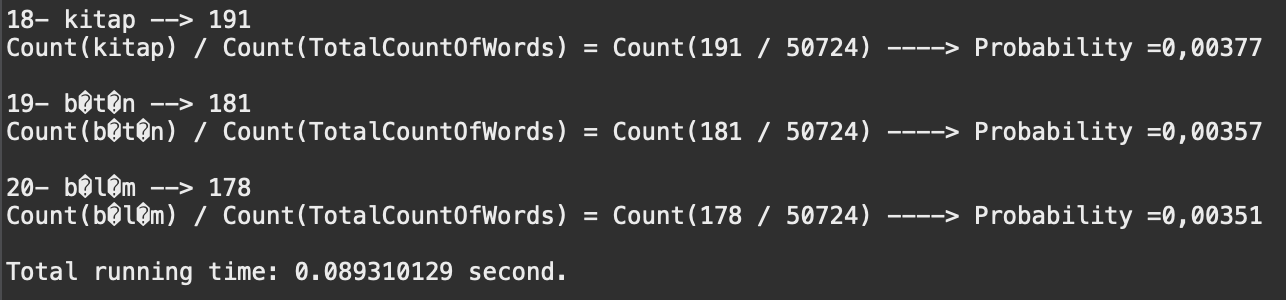


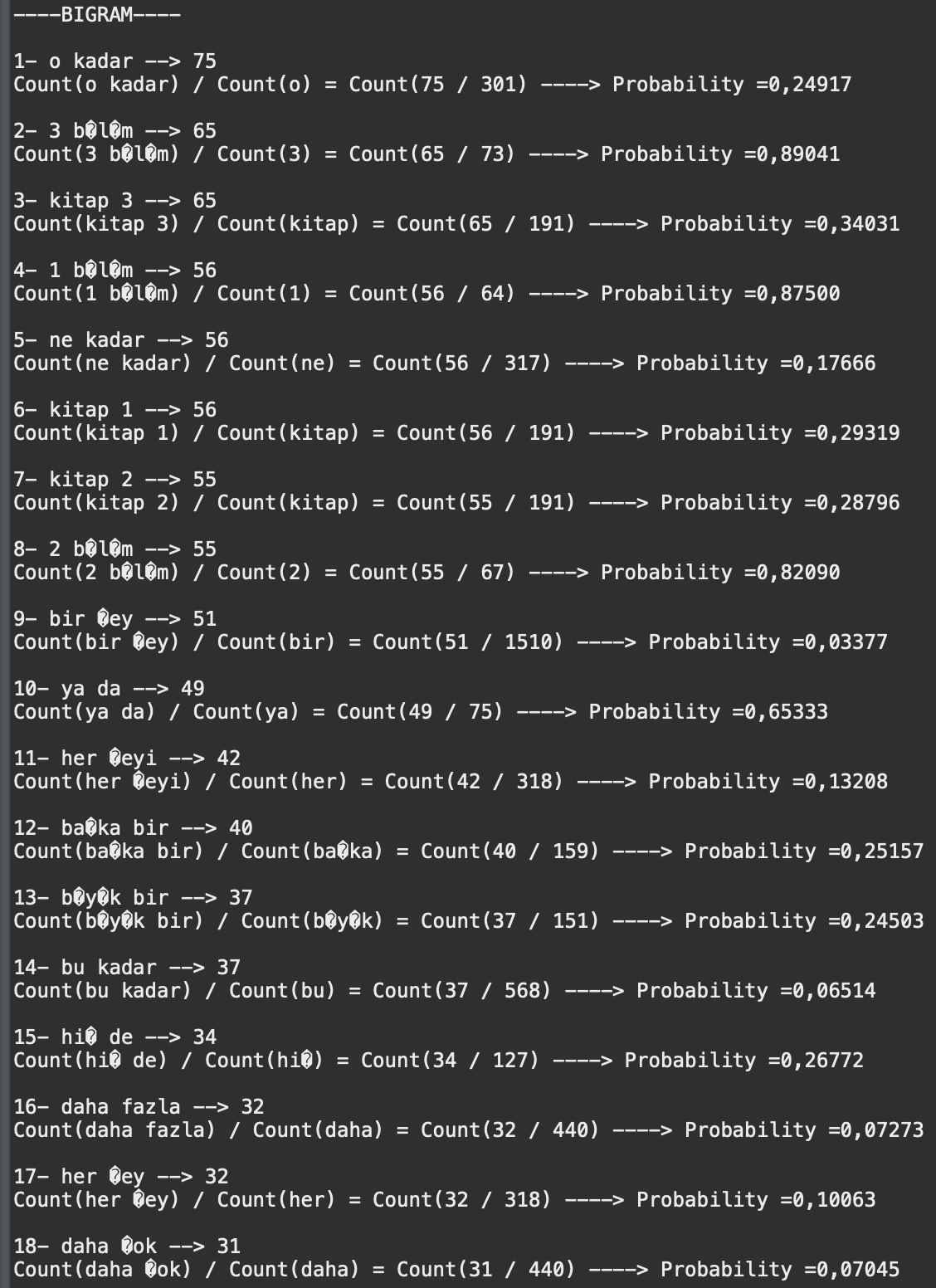


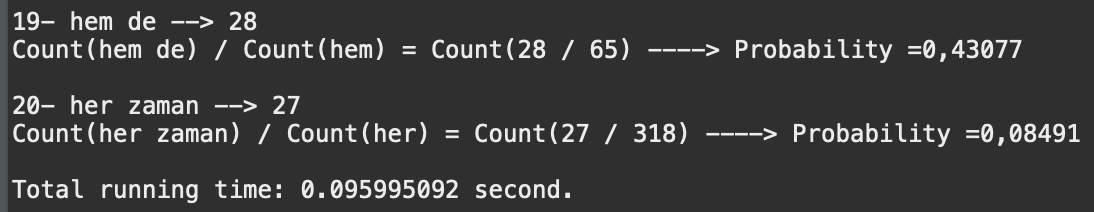


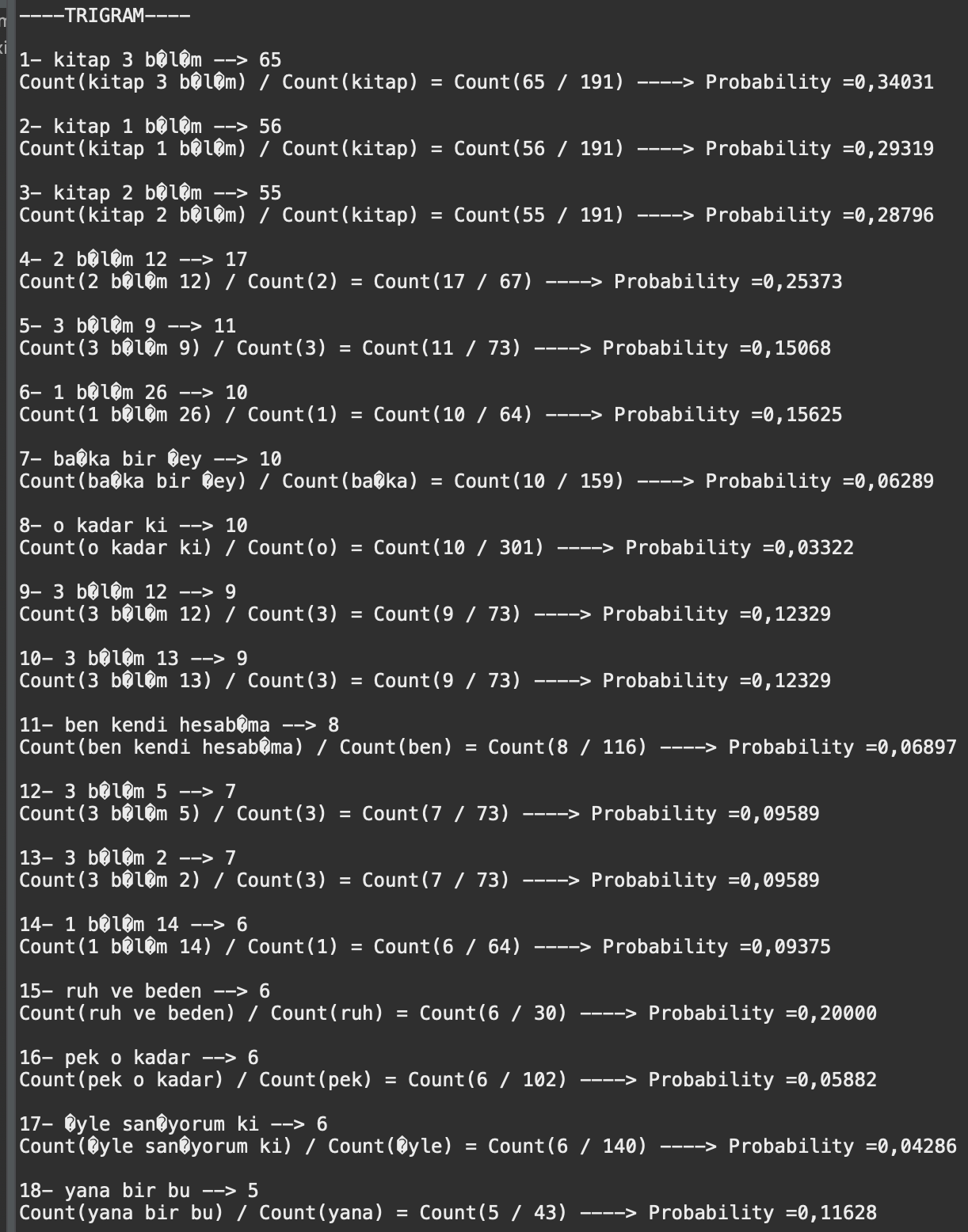
**DENEMELER.TXT**

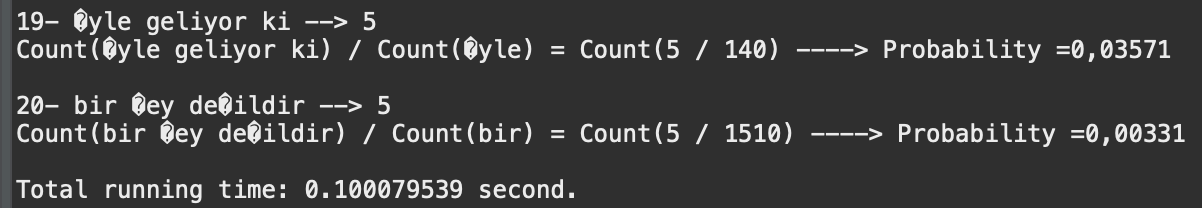




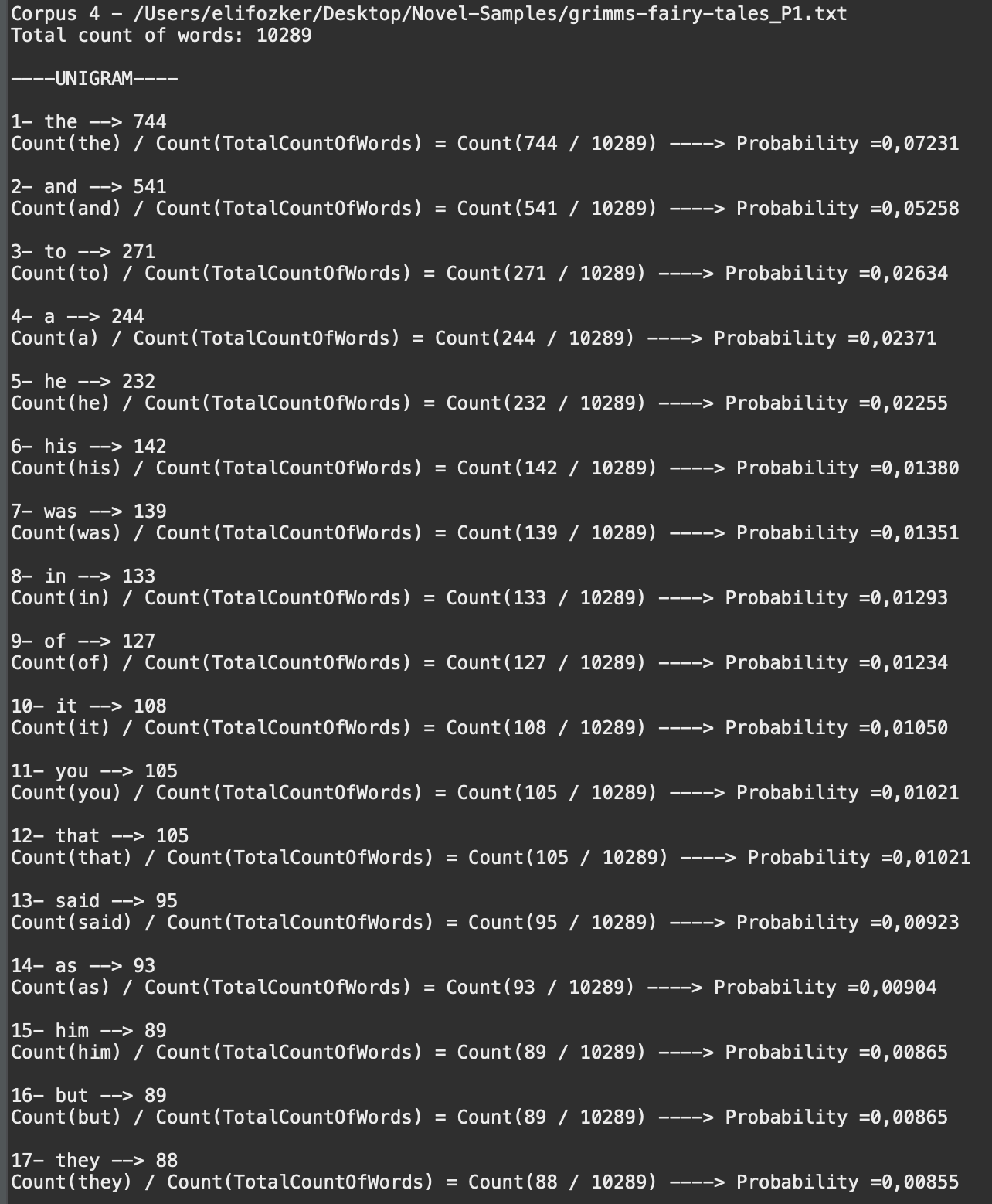


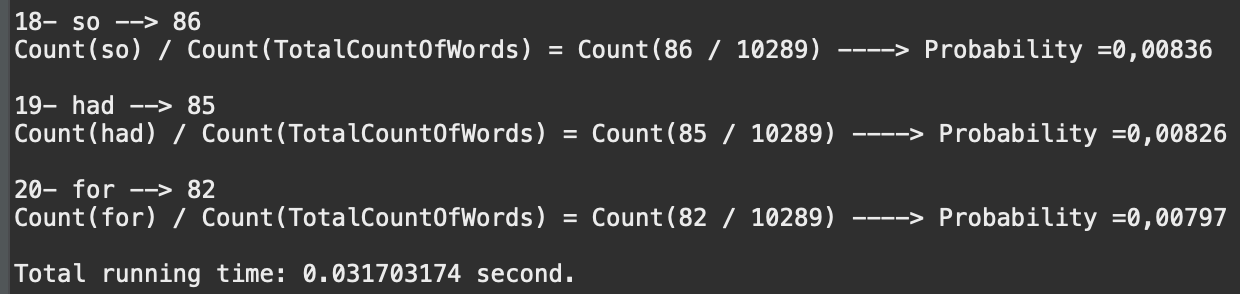


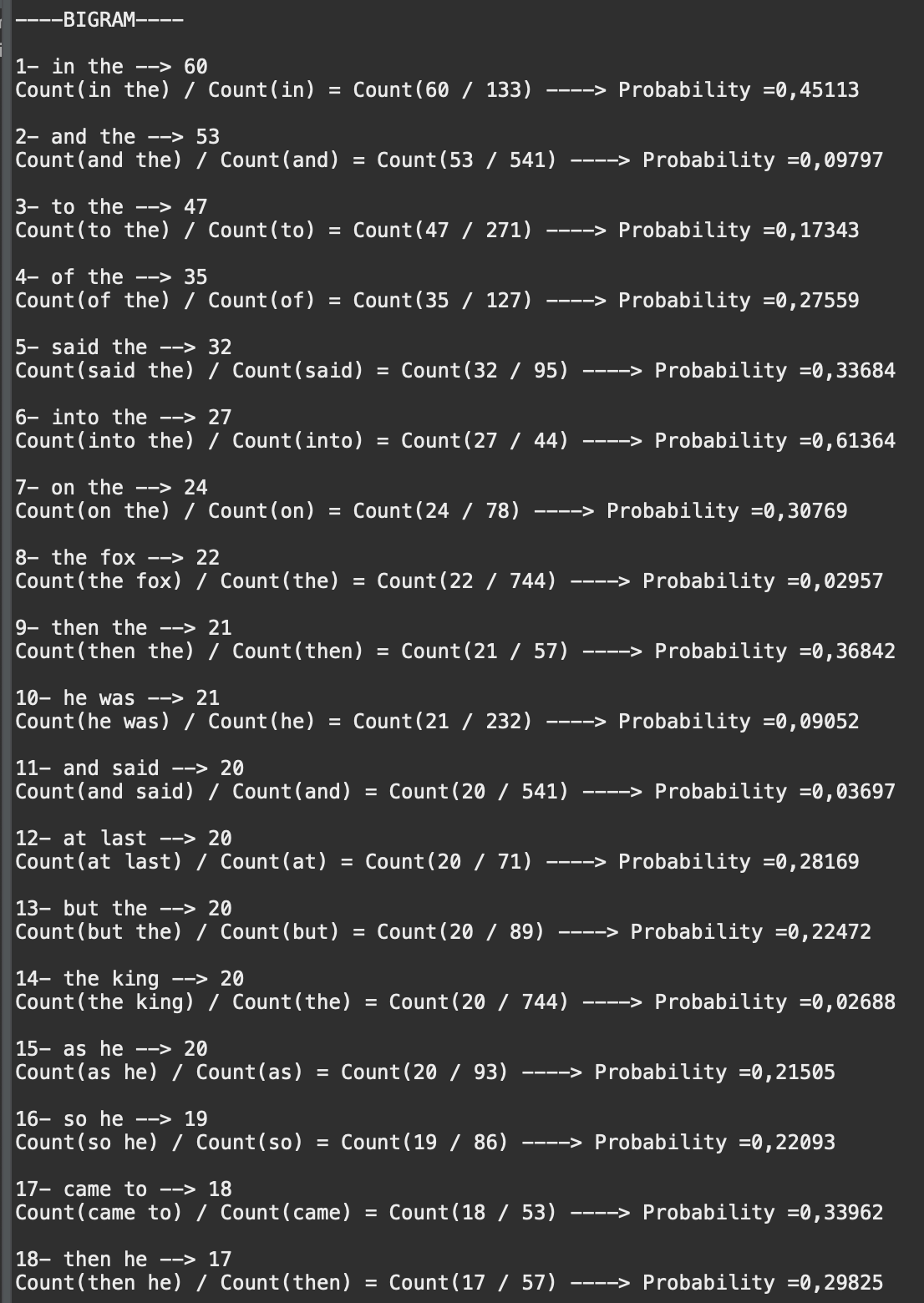




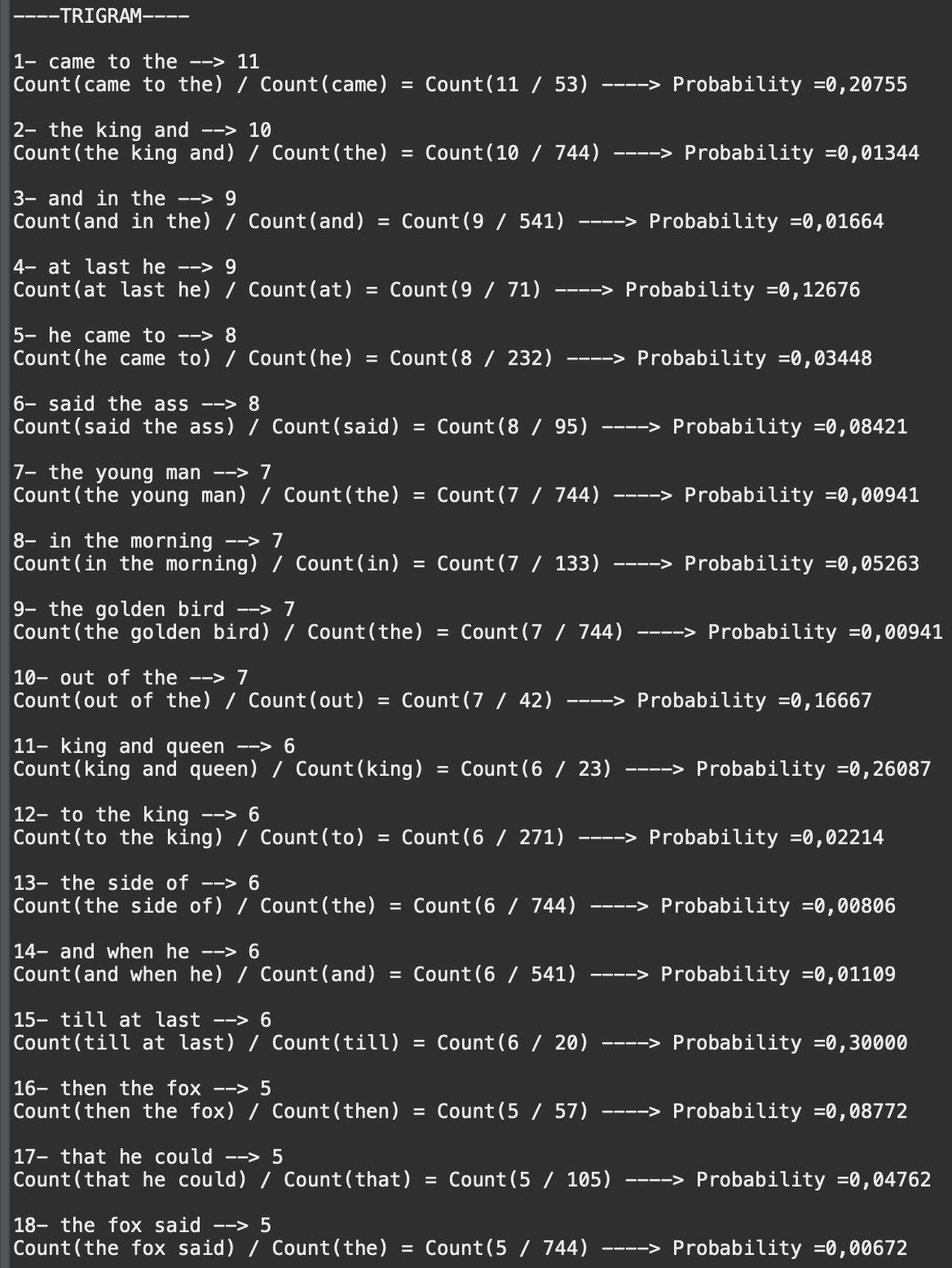
**GRIMMS-FAIRY-TALES.TXT**

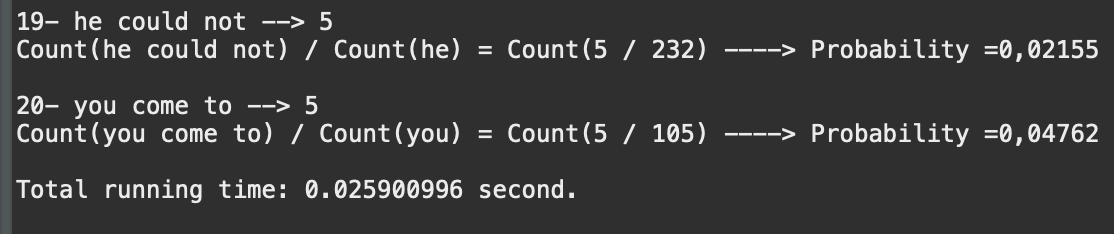












**References:**

<https://stackoverflow.com/questions/3656762/n-gram-generation-from-a-sentence>

**https://www.baeldung.com/java-word-frequency**