# Most and Less Frequent Words

Manuel Diaz

*Resumo* - **Este projeto investiga métodos para identificar as palavras mais frequentes e menos frequentes em ficheiros de texto, utilizando contadores exactos, contadores aproximados e um algoritmo de fluxo de dados. Os dados textuais foram obtidos a partir de várias edições e línguas de "*Alice no País das Maravilhas*", obtidas no *Project Gutenberg*, para analisar a distribuição da frequência das palavras em diferentes traduções.**

**Os métodos implementados foram testados e comparados para avaliar a sua eficiência e precisão computacional. O estudo incluiu etapas de pré-processamento, como a remoção de palavras de paragem, a limpeza da pontuação e a normalização do texto para garantir a consistência. Os contadores exactos forneceram uma linha de base para a precisão, enquanto os contadores aproximados e os algoritmos de fluxo de dados foram avaliados quanto à sua capacidade de equilibrar a eficiência e a precisão.**

*Abstract* - **This project investigates methods for identifying the most frequent and least frequent words in text files, leveraging exact counters, approximate counters, and a data stream algorithm. Textual data was sourced from various editions and languages of "*Alice's Adventures in Wonderland"*, obtained from *Project Gutenberg*, to analyze word frequency distributions across different translations.**

**The implemented methods were tested and compared to evaluate their computational efficiency and accuracy. The study included preprocessing steps such as stop-word removal, punctuation cleaning, and text normalization to ensure consistency. The exact counters provided a baseline for accuracy, while the approximate counters and data stream algorithms were assessed for their ability to balance efficiency and precision.**

## I. INTRODUCTION

The analysis of word frequencies is a fundamental task in computational text processing, with applications ranging from natural language processing (NLP) to information retrieval and text mining. Identifying the most frequent and least frequent words in textual data provides insights into linguistic patterns, vocabulary distribution, and text characteristics. However, the increasing size of modern textual datasets demands efficient methods for processing and analyzing word frequencies.

This project explores three distinct approaches for word frequency analysis: exact counters, approximate counters, and a data stream algorithm. These methods vary in terms of computational complexity, memory requirements, and precision, making them suitable for different scenarios. The primary aim is to evaluate their performance and accuracy in estimating word frequencies and to compare their outputs across multiple editions and translations of "*Alice's Adventures in Wonderland"*, a widely studied literary text, using data from *Project Gutenberg* [1] [2] [3] [4] [5].

Key preprocessing steps were applied to the text files to ensure consistency, including the removal of stop words and punctuation, and the conversion of text to lowercase. This ensures that the analysis focuses solely on meaningful word content. The project also assesses whether frequent word patterns remain consistent across languages and editions of the same book.

By analyzing these methods, this study contributes to understanding the trade-offs between accuracy and computational efficiency in word frequency analysis, providing a comparative framework that can guide future applications in similar contexts.

## II. ALGORITHMS

This section describes the three algorithms implemented to analyze word frequencies in the text data: **Exact Counting**, **Approximate Counting**, and **Lossy Counting**. These algorithms were chosen for their complementary trade-offs in terms of accuracy, memory usage, and computational efficiency.

### *A. Exact Counting*

The Exact Counting algorithm provides a precise frequency count for all words in the dataset. It iterates through the text, tokenizing it into words, and maintains a dictionary where each unique word is a key, and its occurrence count is the corresponding value.

The process begins by splitting the input text into individual words. For each word, the algorithm checks if it is already present in the dictionary. If so, the count for that word is incremented; otherwise, the word is added to the dictionary with an initial count of one.

This algorithm serves as a baseline for comparison with the approximate methods. While it offers perfect accuracy, it has significant computational and memory requirements, particularly for large datasets, as all words and their counts must be stored in memory.

### B. *Approximate Counting*

The Approximate Counting algorithm uses probabilistic sampling to estimate word frequencies, reducing computational and memory demands. Instead of counting every word, it selects words probabilistically based on a predefined sampling rate. Words that pass the sampling test are counted, and the final counts are scaled by the inverse of the sampling rate to approximate the actual frequencies.

In my example, the sampling rate is set to $p$ = 1/2, then approximately 1 in every 2 words is included in the count. The counts for these sampled words are then multiplied by 2 to estimate their overall frequency in the dataset. This approach introduces estimation errors, particularly for less frequent words, but offers a substantial reduction in resource usage.

To ensure reliable results, the algorithm was run multiple times (500 iterations), and the average counts were computed. This method balances efficiency and accuracy, making it suitable for large datasets where exact counting is impractical.

### C. *Lossy Counting*

The Lossy Counting algorithm is designed for streaming data and focuses on identifying the most frequent words. It processes the text in fixed segments, updating word counts incrementally while discarding words with counts below a dynamically updated threshold (Algorithm 1).

The input text is divided into equal-sized segments. For each segment, the algorithm updates word counts in an internal dictionary. A threshold, determined by the total number of processed words divided by the segment size, is used to periodically prune the dictionary. Words with counts below this threshold are removed, reducing memory usage. As a result, the Lossy Count algorithm is particularly well-suited for large data streams where maintaining counts of all unique items is not feasible due to memory constraints.

An important characteristic of this algorithm is its adaptability to different values of *segment_size.* A smaller *segment_size* results in a larger threshold, leading to more agressive filtering of items. Conversely, a larger *segment_size* value lowers the threshold, allowing more items to be retained. This tunability makes the Lossy Count algorithm versatile, as it can be adjusted according to the specific requirements of a task, balancing between memory usage and the granularity of frequency data retained.

```
def lossy_frequency_counter(input_text: str,
segment_size: int = 10):
    frequency_map = defaultdict(int)
    total_words = 0
    current_threshold = 0

    words = input_text.split()

    for word in words:
        if word not in frequency_map:
            frequency_map[word] = current_threshold +
1
        else:
            frequency_map[word] += 1

        total_words += 1
        new_threshold = floor(total_words /
segment_size)

        if new_threshold != current_threshold:
            current_threshold = new_threshold
            for key, value in list(frequency_map.items()):
                if value < current_threshold:
                    del frequency_map[key]

    return frequency_map
```

Algorithm 1: Lossy Count algorithm implementation.

## III. METHODOLOGY

This project aims to test the implemented algorithms to analyse word frequency distributions across different books provided by *Project Gutenberg*. For the sake of simplicity, all the benchmarks presented in this paper will be made under "*Alice's Adventures in Wonderland*" by Lewis Carroll in different languages that simulate the data stream.

### A. *Text Preprocessing*

Before applying the algorithms, the text files were preprocessed to ensure consistency:

- Headers, footers, and metadata unrelated to the book's content were removed.
- All text was converted to lowercase to eliminate case-sensitivity issues.
- Stop words were excluded using the Natural Language Toolkit (NLTK) library [6].
- Non-alphabetic tokens, such as punctuation and numbers, were filtered out.

This preprocessing ensured that the analysis focused only on meaningful words, providing a reliable basis for word frequency comparisons across editions and languages.

## B. *Parameter Selection*

To evaluate the algorithms' performance, the following parameters were chosen:

- For approximate counting, 500 runs were performed to compute average results.

- For benchmarking purposes, the lossy counter was initialised with *segment_size* ∈ {5,10,15,20}, which are the top n most frequent words.

## C. *Evaluation Metrics*

The algorithms were compared using several metrics, including:

- **Mean Absolute Error (MAE):** The average absolute difference between estimated and exact word counts.
- **Mean Relative Error (MRE):** The average relative difference normalised by the exact counts.
- **Mean Accuracy Ratio:** Evaluates how closely the algorithm's frequency estimates align with actual frequencies on average.
- **Smallest and Largest Values:** The range of frequency counts, giving an idea of the spread in the algorithm's predictions.
- **Mean, Mean Absolute Deviation, and Standard Deviation:** These measure the central tendency and variability in the frequency estimates, indicating the average count and consistency of predictions.
- **Variance and Maximum Deviation:** Examine the spread and the furthest count from the mean, respectively, providing insights into the consistency and reliability of the algorithms.
- **Time Efficiency:** The execution time of each algorithm for different datasets.

## IV. RESULTS

## A. *Overview of Word Counts*

The total and unique word counts for each edition were computed as part of the preprocessing phase. These statistics provide insights into the linguistic richness and variability across different translations of "*Alice's Adventures in Wonderland*". In table I is a summary of the word counts:

| **Books** | **Total** | **Unique** |
|---|---|---|
| Alice A. In W. (English) | 13,083 | 2,392 |
| Alice A. In W. (French) | 23,428 | 3,191 |
| Alice A. In W. (German) | 23,464 | 3,571 |
| Alice A. In W. (Italian) | 21590 | 4196 |
| Alice A. In W. (Finnish) | 18,360 | 5,544 |

TABLE I
TOTAL AND UNIQUE NUMBER OF WORDS

## B. *Top Words Analysis*

The most frequent words identified by each algorithm were compared to evaluate their consistency. We can see in tables II to VII a comparison of the most frequent words determined by the exact counter, the fixed probability counter, and the lossy counting algorithm at 5 and 20 *n* values, to simplify. The exact and fixed probability counters demonstrate a strong alignment in all cases, which underscores the reliability of the fixed probability counter in approximating the exact frequency counts across different languages.

By contrast, the lossy counting algorithm demonstrates more variability in its performance, especially at lower *n* values, where it occasionally fails to identify the most frequent letters captured by the exact method. However, as *n* increases, its results become closer to the exact counts, suggesting its potential effectiveness in scenarios where balancing accuracy and computational efficiency is essential due to resource constraints.

We can also see similarities in the most commonly used words in the various languages, with an emphasis on propositions and articles.

To visually compare the performance of the algorithms, bar charts were generated for the top-n most frequent words (n = 5, 10, 15, 20) across the analysed editions of the books. The charts present in figures 1 to 4 illustrate how each algorithm captures the frequency of the most commonly ocurring words and highlights differences in their outputs. It is noteworthy, that the missing values from the lossy counting are caused by the incapability of it to identify them as the most frequent words.

| **Algorithm** | **Most Frequent Words** |
|---|---|
| Exact | I, Said, Alice, Little, The, One, Like, Went, Would, Queen |
| Approximate | I, Said, Alice, Little, The, One, Like, Went, Would, Queen |
| Lossy (n = 5) | Days, The, End, Simple, Joys, Remembering, Happy, Summer |
| Lossy (n = 20) | Would, Said, Simple, Days, The, End, Many, Strange, Tale, Perhaps, Even, Dream, Wonderland, Long, Ago, Feel, Sorrows, Find, Pleasure, Joys, Remembering, Happy, Summer |

TABLE II
MOST FREQUENT WORDS FOR ALICE_EN.TXT

| **Algorithm** | **Most Frequent Words** |
|---|---|
| Exact | De, La, Le, Et, À, Que, Dit, Alice, Elle, Je |
| Approximate | De, La, Le, Et, À, Que, Dit, Alice, Elle, Je |
| Lossy (n = 5) | Heureux, Jours, Fin, Sa, Propre, Enfance, Et, Les |
| Lossy (n = 20) | Et, Leurs, Sa, Propre, Enfance, Les, Heureux, Jours, Fin, Le, Elle, Des, Du, Marveilles, Temps, Jadis, La, Voyait, Partager, Petits, Chagrins, Trouver, Plaisir, À, Innocentes, Joies, Se, Rappelant |

TABLE III
MOST FREQUENT WORDS FOR ALICE_FR.TXT

| **Algorithm** | **Most Frequent Words** |
|---|---|
| Exact | Sie, Und, Die, Ich, Der, Es, Zu, Alice, Sagte, Das |
| Approximate | Sie, Und, Die, Ich, Der, Es, Zu, Alice, Sagte, Das |
| Lossy (n = 5) | Wunderschöner, Traum, Gewesen, Sei, Und, Recht, Daß, Es, Doch, Ein |
| Lossy (n = 20) | Und, Es,Wunderschöner, Traum, Gewesen, Sei, Sie, Ein, Wird, Da,Stand, Alice, Auf, Rannte, Fort, Dachte, Dabei, Zwar, Mit, Recht, Daß, Doch |

TABLE IV
MOST FREQUENT WORDS FOR ALICE_GR.TXT

| **Algorithm** | **Most Frequent Words** |
|---|---|
| Exact | E, Il, Che, La, Di, Non, Alice, Un, Disse, Si |
| Approximate | E, Il, Che, La, Di, Non, Alice, Un, Disse, Si |
| Lossy (n = 5) | E, Le, Gioconde, Giornate, Fine |
| Lossy (n = 20) | E, Il, Con, Quanta, Alle, Loro, Riandando, Beati, Giorni, Della, Fanciullezza, Le, Gioconde, Giornate, Fine, Delle, Simpatica, Tenerezza, avrebbe, Ella, Stessa, Partecipato, Innocenti, Angosce, Letizia, Gioje |

TABLE IV
MOST FREQUENT WORDS FOR ALICE_IT.TXT

| Algorithm | Most Frequent Words |
|---|---|
| Exact | Ja, Hän, Liisa, Oli, Sanoi, Ei, Se, Mutta, Niin, Että |
| Approximate | Ja, Hän, Liisa, Oli, Sanoi, Ei, Se, Mutta, Niin, Että |
| Lossy (n = 5) | Ja, Sen, Onnellisia, Kultapäiviä, Loppu |
| Lossy (n = 20) | Ja, Hän, Monta, Pikku, Liisa, Vanha, Ihmisenäkin, Ymmärtää, Lasten, Yksinkertaiset, Huolet, Nauttii, Heidän, Iloistaan, Muistellen, Omaa, Lapsuuttaan, Sen, Onnellisia, Kultapäivia, Loppu |

TABLE V
MOST FREQUENT WORDS FOR ALICE_FI.TXT

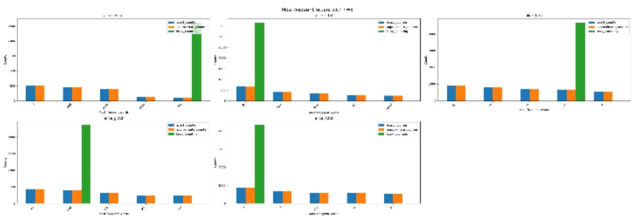

Figure 1: Most frequent words count with n = 5 for all algorithms

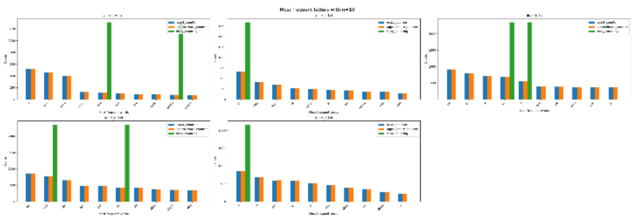

Figure 2: Most frequent words count with n = 10 for all algorithms

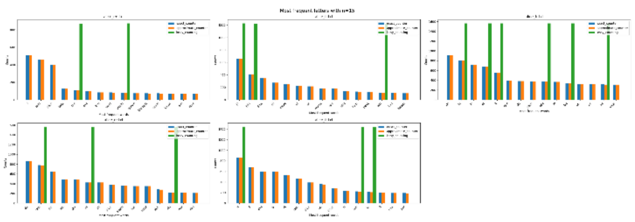

Figure 3: Most frequent words count with n = 15 for all algorithms

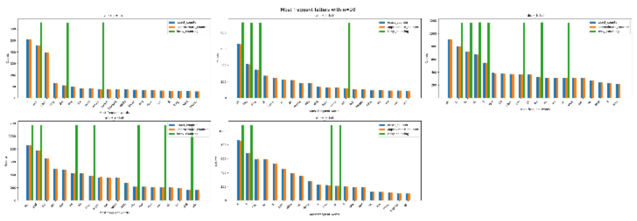

Figure 4: Most frequent words count with n = 20 for all algorithms

## C. *Evaluation of Metrics*

To assess the performance of the implemented algorithms, a comprehensive evaluation was conducted using various statistical and accuracy-related metrics. The tables from VI to X present the results for the English edition of "*Alice's Adventures in Wonderland*". However, all the remaining statistics from the other available languages are in the results folder from the project.

As we can see in the tables, the mean absolute error and mean relative error metrics indicate that lossy counting algorithm's accuracy improves with higher n values, as the count converge closer to the true frequencies. The mean accuracy ratio further substantiates this observation, with lossy counting demonstrating an acceptable level of precision at *n* = 20.

The Approximate Counting algorithm showed low mean absolute error and mean relative error, indicating that its overall estimates were close to the exact counts. However, it exhibited higher variance and standard deviation compared to the Lossy Counting algorithm. This discrepancy is due to the probabilistic nature of Approximate Counting, which can cause greater variability in the estimated counts of less frequent words.

In contrast, the Lossy Counting algorithm, which discards infrequent words below a threshold, produced more stable results with lower variance and standard deviation. This makes it particularly effective for identifying the most frequent words.

While Approximate Counting is accurate on average, its variability may affect results for less frequent words. Increasing the number of iterations or analyzing absolute differences between exact and approximate counts could help improve its stability.

The smallest and largest values, along with the mean, give insight into the range and central tendency of the counts.

| Metric | Value |
|---|---|
| Mean Absolute Error | 14.10 |
| Mean Relative Error | 5.47 |
| Mean Accuracy Ratio | 4.48 |
| Smallest Value | 2616 |
| Largest Value | 2617 |
| Mean | 2616.38 |
| Mean Absolute Deviation | 0.47 |
| Standard Deviation | 0.48 |
| Maximum Deviation | 0.625 |
| Variance | 0.23 |

TABLE VI
STATISTICS FOR ALICE_EN.TXT WITH LOSSY COUNTING (N = 5)

| Metric | Value |
|---|---|
| Mean Absolute Error | 12.37 |
| Mean Relative Error | 4.15 |
| Mean Accuracy Ratio | 3.16 |
| Smallest Value | 1308 |
| Largest Value | 1309 |
| Mean | 1308.30 |
| Mean Absolute Deviation | 0.42 |
| Standard Deviation | 0.46 |
| Maximum Deviation | 0.69 |
| Variance | 0.21 |

TABLE VII
STATISTICS FOR ALICE_EN.TXT WITH LOSSY COUNTING (N = 10)

| Metric | Value |
|---|---|
| Mean Absolute Error | 11.4 |
| Mean Relative Error | 3.52 |
| Mean Accuracy Ratio | 2.54 |
| Smallest Value | 872 |
| Largest Value | 875 |
| Mean | 872.41 |
| Mean Absolute Deviation | 0.58 |
| Standard Deviation | 0.77 |
| Maximum Deviation | 2.59 |
| Variance | 0.60 |

TABLE VIII
STATISTICS FOR ALICE_EN.TXT WITH LOSSY COUNTING (N = 15)

| Metric | Value |
|---|---|
| Mean Absolute Error | 11.09 |
| Mean Relative Error | 3.06 |
| Mean Accuracy Ratio | 2.08 |
| Smallest Value | 654 |
| Largest Value | 658 |
| Mean | 654.48 |
| Mean Absolute Deviation | 0.71 |
| Standard Deviation | 1.02 |
| Maximum Deviation | 3.52 |
| Variance | 1.03 |

TABLE IX
STATISTICS FOR ALICE_EN.TXT WITH LOSSY COUNTING (N = 20)

| Metric | Value |
|---|---|
| Mean Absolute Error | 0.75 |
| Mean Relative Error | 0.47 |
| Mean Accuracy Ratio | 0.52 |
| Smallest Value | 0 |
| Largest Value | 511 |
| Mean | 6.74 |
| Mean Absolute Deviation | 7.40 |
| Standard Deviation | 22.08 |
| Maximum Deviation | 504.26 |
| Variance | 487.36 |

TABLE X
STATISTICS FOR ALICE_EN.TXT WITH FIXED PROBABILITY COUNTER

### D. *Time Analysis*

In the tables XI to XV we can see the execution times of each algorithm for all the book editions tested.

The Exact Counting algorithm consistently demonstrated the fastest execution times across all editions. For instance, the English edition (alice_en.txt) was processed in approximately 0.0027 seconds. This efficiency is due to the simplicity of the method, which directly iterates through the dataset without probabilistic or segment-based adjustments.

The Approximate Counting algorithm performed similarly to Exact Counting, with marginally faster times for most editions. For example, in the Finnish edition (alice_fi.txt), it achieved 0.0029 seconds compared to 0.0034 seconds for Exact Counting. The close performance between these two algorithms reflects the minimal additional overhead introduced by the probabilistic sampling mechanism.

Compared with these two, the Lossy Counting algorithm exhibited higher execution times, with durations increasing as the number of top-n words to track rose. For example, in the French edition (alice_fr.txt), tracking the top-5 words took 0.0071 seconds, while tracking the top-20 words took 0.0063 seconds. This behavior is expected since Lossy Counting involves maintaining and updating a dynamic dictionary, as well as periodically pruning infrequent words.

In this way, we can conclude some efficiency trade-offs**:**

- **Exact Counting** offers the quickest processing times but lacks the memory optimizations of the other methods.
- **Approximate Counting** balances speed and efficiency, achieving times comparable to Exact Counting with a slight trade-off in accuracy.
- **Lossy Counting** requires more time due to its dynamic nature but is optimized for identifying the most frequent words in resource-constrained environments.

| **Algorithm** | **Time (seconds)** |
|---|---|
| Exact Counter | 0.0027 |
| Approximate Counter | 0.0026 |
| Lossy Counting (n = 5) | 0.0038 |
| Lossy Counting (n = 10) | 0.0035 |
| Lossy Counting (n = 15) | 0.0034 |
| Lossy Counting (n = 20) | 0.0037 |

TABLE XI
COMPUTATIONAL TIMES FOR ALICE_EN.TXT

| **Algorithm** | **Time (seconds)** |
|---|---|
| Exact Counter | 0.0036 |
| Approximate Counter | 0.0030 |
| Lossy Counting (n = 5) | 0.0071 |
| Lossy Counting (n = 10) | 0.0070 |
| Lossy Counting (n = 15) | 0.0064 |
| Lossy Counting (n = 20) | 0.0062 |

TABLE XII
COMPUTATIONAL TIMES FOR ALICE_FR.TXT

| **Algorithm** | **Time (seconds)** |
|---|---|
| Exact Counter | 0.0051 |
| Approximate Counter | 0.0031 |
| Lossy Counting (n = 5) | 0.0072 |
| Lossy Counting (n = 10) | 0.0071 |
| Lossy Counting (n = 15) | 0.0067 |
| Lossy Counting (n = 20) | 0.0071 |

TABLE XIII
COMPUTATIONAL TIMES FOR ALICE_GR.TXT

| **Algorithm** | **Time (seconds)** |
|---|---|
| Exact Counter | 0.0038 |
| Approximate Counter | 0.0029 |
| Lossy Counting (n = 5) | 0.0065 |
| Lossy Counting (n = 10) | 0.0066 |
| Lossy Counting (n = 15) | 0.0061 |
| Lossy Counting (n = 20) | 0.0058 |

TABLE XIV
COMPUTATIONAL TIMES FOR ALICE_IT.TXT

| **Algorithm** | **Time (seconds)** |
|---|---|
| Exact Counter | 0.0034 |
| Approximate Counter | 0.0029 |
| Lossy Counting (n = 5) | 0.0055 |
| Lossy Counting (n = 10) | 0.0058 |
| Lossy Counting (n = 15) | 0.0053 |
| Lossy Counting (n = 20) | 0.0054 |

TABLE XV
COMPUTATIONAL TIMES FOR ALICE_FI.TXT

## V. CONCLUSION

This project focused on the application and evaluation of three algorithms—Exact Counting, Approximate Counting, and Lossy Counting—for analyzing word frequencies across different editions and translations of *Alice's Adventures in Wonderland*. The objective was to assess the trade-offs between accuracy, computational efficiency, and memory usage in textual data processing.

The Exact Counting algorithm provided perfect accuracy and served as a reliable baseline for evaluating the performance of the other methods. However, its high computational and memory demands make it less suitable for large-scale datasets or streaming data scenarios. On the other hand, Approximate Counting demonstrated a strong balance between efficiency and accuracy.

The Lossy Counting algorithm proved effective in identifying the most frequent words with minimal memory usage, making it particularly useful for resource-constrained environments. While its accuracy for less frequent words was slightly lower, it maintained stable performance for top-n word analysis, offering a practical solution for streaming data.

The variability in word distributions and linguistic structures across the different languages analyzed slightly influenced the performance of all algorithms. Texts with greater lexical richness or higher word counts required marginally more computational resources, which is expected in natural language processing tasks.

Overall, this study demonstrated that Exact Counting is best suited for scenarios where perfect accuracy is critical, while Approximate Counting offers efficient and near-accurate results for large-scale datasets. Lossy Counting, on the other hand, excels in environments requiring memory efficiency and prioritizes the identification of frequent words.

## REFERENCES

[1] Lewis Carroll, *Alice's Adventures in Wonderland*, 2008. Available: https://www.gutenberg.org/ebooks/11

[2] Lewis Carroll, *Aventures d'Alice au pays des merveilles*, 2017. Available: https://www.gutenberg.org/ebooks/55456

[3] Lewis Carroll, *Alice's Abenteuer im Wunderland*, 2006. Available: https://www.gutenberg.org/ebooks/19778

[4] Lewis Carroll, *Le avventure d'Alice nel paese delle meraviglie*, 2009. Available: https://www.gutenberg.org/ebooks/28371

[5] Lewis Carroll, *Liisan seikkailut ihmemaassa*, 2014. Available: https://www.gutenberg.org/ebooks/46569

[6] NLTK Project, 2024, Accessed: 2024-12-29. Available: https://www.nltk.org/