

Lemmatization exercise

Erick Gonzalez Parada ID: 178145

Emiliano Ruiz Plancarte ID: 177478

Andre Francois Duhamel Gutierrez ID: 177315

Antonio Gutiérrez Blanco ID: 177442

March 23, 2025

Abstract

This document explores the implementation of a lemmatization system using the Snowball stemmer library to optimize text processing for search engines. By reducing words to their base forms and removing stop words, the system improves document retrieval efficiency. The integration of the Snowball stemmer into a Java application demonstrates its effectiveness in enhancing search engine performance.

Keywords: Stemmer , word, string, documents.

1 Characteristics of the problem

Goals

The primary goal of this project is to identify the limitations of a medium-level stemmer string processor and enhance its capabilities to be used as part of a future basic search engine. In document retrieval systems, optimization is critical, and all string processing must be as efficient as possible. The project focuses on improving the stemmer's ability to handle complex word structures, reduce words to their root forms, and remove stop words to enhance search engine performance.

Materials

- *JDK 17 or above*

Text processing is a fundamental task in search engines, where efficiency and accuracy are critical for retrieving relevant documents. Traditional string processing methods often struggle with handling complex word structures, such as inflectional forms and stop words, which can degrade search engine performance [1]. For example, words like "running," "ran," and "runs" should be reduced to their base form "run" to improve search accuracy. Additionally, stop words like "the," "and," and "or" add little semantic value but increase processing overhead [3]. The Snowball stemmer, a widely used tool for lemmatization, addresses these challenges by reducing words to their root forms and removing stop words [4]. However, integrating and optimizing such tools for specific applications, like search engines, requires careful customization and testing.

2 Proposed Solution

We, as a team, developed an improved version of the Snowball stemmer software, which is available at <https://github.com/HugeErick/Stemmer>. The solution involves the following steps:

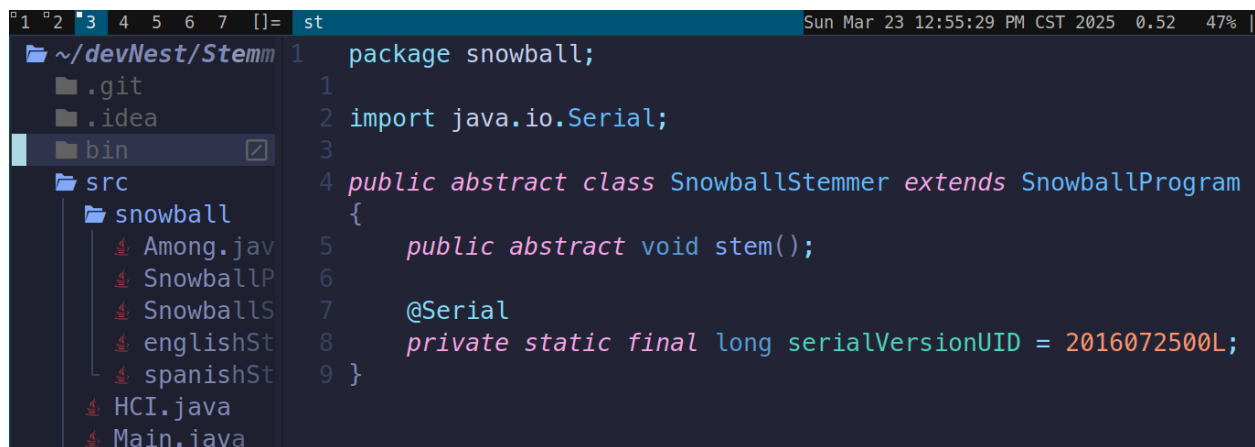
- **Refactoring the Snowball Stemmer:** The original Snowball stemmer code was refactored to integrate it into our Java application. This involved modifying the stemmer's API to handle English and Spanish text effectively.

- **Lemmatization and Stop Word Removal:** The application processes input text by reducing words to their root forms (lemmatization) and removing stop words (e.g., "the", "and", "or") to optimize search engine queries.
- **User Interaction:** The application allows users to input text and select the stemmer language (English or Spanish). The processed text is then displayed, showing the lemmatized output and the removed stop words.
- **Validation and Testing:** The system was tested with various text inputs to ensure accurate lemmatization and stop word removal. The results were validated against expected outputs to confirm the system's effectiveness.

The proposed solution demonstrates the successful integration of the Snowball stemmer into a Java application, providing a robust tool for text processing in search engine optimization.

3 Interpretation of the findings

proof that "snowball" was used



```

1 package snowball;
2
3 import java.io.Serial;
4
5 public abstract class SnowballStemmer extends SnowballProgram
6 {
7     public abstract void stem();
8
9     @Serial
10    private static final long serialVersionUID = 2016072500L;
11 }

```

Figure 1: The snowball software was refactored inside our program

The figure 1 shows that there was a severe refactorization of the code to be able to activate the API of the snowball.

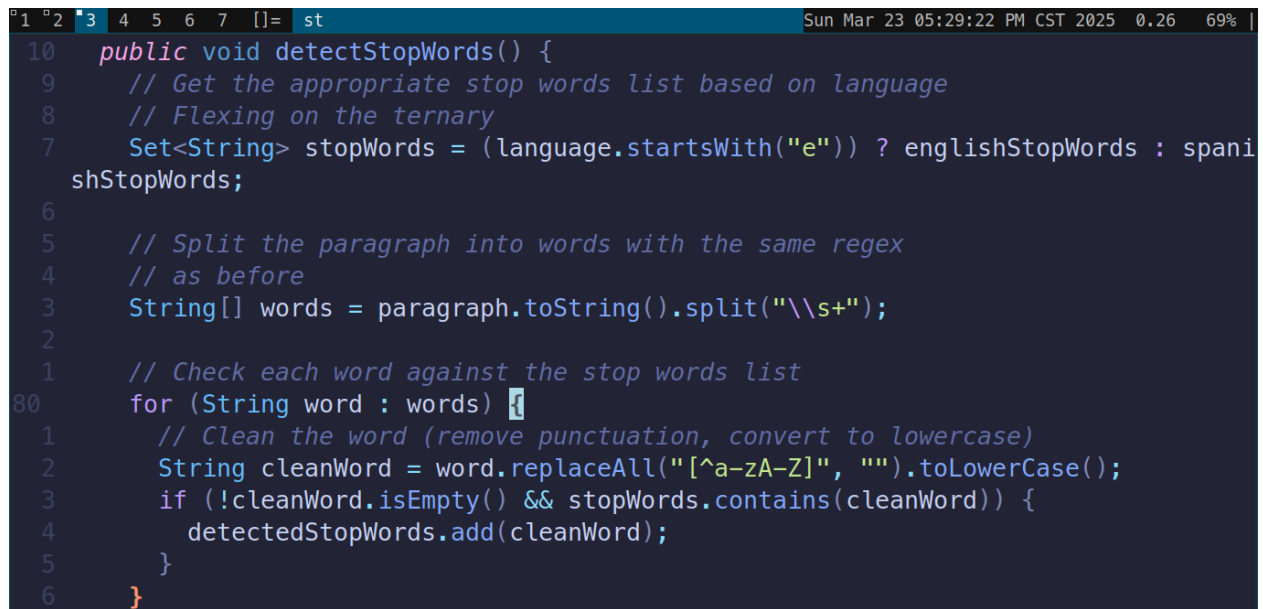
```
1 2 3 4 5 6 7 []= st Sun Mar 23 01:14:34 PM
[erick@draven Stemmer]$ java -cp bin Main
Enter the stemmer language ('english' or 'spanish'):
e
Enter text to lemmatize (type 'exit' to quit):
The goal is to identify bla bla tomatoe onion and or not
Lemmatized: the goal is to identifi bla bla tomato onion and or not
Stop words detected: the, not, or, and, is, to

Removing stop words too

Result after removing stop words: goal identifi bla bla tomato onion
[1] 0:java* 1:nvim- "erick@draven:~/devNes"
```

Figure 2: usage sample

The figure 2 shows a usage the example of the app



```

10 public void detectStopWords() {
9     // Get the appropriate stop words list based on language
8     // Flexing on the ternary
7     Set<String> stopWords = (language.startsWith("e")) ? englishStopWords : spanishStopWords;
6
5     // Split the paragraph into words with the same regex
4     // as before
3     String[] words = paragraph.toString().split("\\s+");
2
1     // Check each word against the stop words list
80 for (String word : words) {
1     // Clean the word (remove punctuation, convert to lowercase)
2     String cleanWord = word.replaceAll("[^a-zA-Z]", "").toLowerCase();
3     if (!cleanWord.isEmpty() && stopWords.contains(cleanWord)) {
4         detectedStopWords.add(cleanWord);
5     }
6 }

```

Figure 3: snapshot of some of the processing of the stop-words

The figure 3 shows a fragment of code where we are comparing the stemmed input against the stop words of the respective language[2].

4 Access to Source Code

The source code for this project is publicly available on GitHub at <https://github.com/HugeErick/Stemmer>. Anyone can access the repository to review the implementation, contribute to the project, or run the application locally. To get the application running, simply follow the instructions provided in the README file. The repository includes detailed steps for setting up the environment, compiling the code, and executing the application.

5 Conclusions

The project highlights the importance of efficient string processing in search engine optimization and demonstrates the successful use of the Snowball stemmer in achieving this goal.

References

- [1] Applications, advantages and disadvantages of string. (2022, May 25). GeeksforGeeks. from <https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-string/>
- [2] Gupta, M. (2016, November 11). Java StringTokenizer class. GeeksforGeeks. from <https://www.geeksforgeeks.org/stringtokenizer-class-in-java/>
- [3] Improve, G. (2017, July 10). How do search engines work? GeeksforGeeks. from <https://www.geeksforgeeks.org/search-engines-work/>
- [4] Snowball. (n.d.). Snowballstem.org. Retrieved March 23, 2025, from <https://snowballstem.org/>