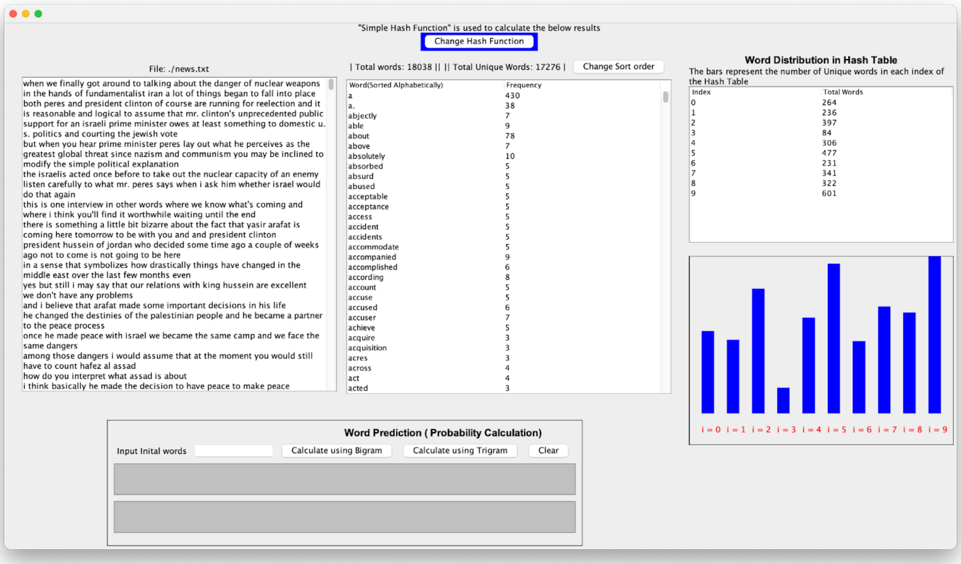
Large language Model

Author: Manu Kenchappa Junjanna
Email: mkenchappajunjanna1@sheffield.ac.uk

UserManual

Upon launching the program, users are presented with the initial choice of either directly selecting 'news.txt' or choosing any file using the 'Pick News' and 'Pick File' buttons, respectively. After the user makes a file selection, the program processes the data, and the results are displayed. In Figure 1, the interface illustrates the left section showcasing the contents of the selected file, the centre section presenting a table depicting words and their frequencies. The centre section also includes a "Change Sort Order" button for toggling between alphabetical and descending frequency sorting. In the rightmost corner, the Word Distribution in the Hash Table is displayed, offering insights into the word count at each index of the Hash Table. Additionally, a bar graph below enhances the visualisation of the distribution.



"Simple Hash Function" is used to calculate the below results

File: ./news.txt

```
when we finally got around to talking about the danger of nuclear weapons
in the hands of fundamentalist iran a lot of things began to fall into place
both peres and president clinton of course are running for reelection and it
is reasonable and logical to assume that mr. clinton's unprecedented public
support for an israeli prime minister owes at least something to domestic u.
s. politics and courting the jewish vote
but when you hear prime minister peres lay out what he perceives as the
greatest global threat since nazism and communism you may be inclined to
modify the simple political explanation
the israelis acted once before to take out the nuclear capacity of an enemy
listen carefully to what mr. peres says when i ask him whether israel would
do that again
this is one interview in other words where we know what's coming and
where i think you'll find it worthwhile waiting until the end
there is something a little bit bizarre about the fact that yasir arafat is
coming here tomorrow to be with you and and president clinton
president hussein of jordan who decided some time ago a couple of weeks
ago not to come is not going to be here
in a sense that symbolizes how drastically things have changed in the
middle east over the last few months even
yes but still i may say that our relations with king hussein are excellent
we don't have any problems
and i believe that arafat made some important decisions in his life
he changed the destinies of the palestinian people and he became a partner
to the peace process
once he made peace with israel we became the same camp and we face the
same dangers
among those dangers i would assume that at the moment you would still
have to count hafez al assad
how do you interpret what assad is about
i think basically he made the decision to have peace to make peace
```

| Total words: 18038 || || Total Unique Words: 17276 |

Word(Sorted Alphabetically)	Frequency
a	430
a.	38
abjectly	7
able	9
about	78
above	7
absolutely	10
absorbed	5
absurd	5
abused	5
acceptable	5
acceptance	5
access	5
accident	5
accidents	5
accommodate	5
accompanied	9
accomplished	6
according	8
account	5
accuse	5
accused	6
accuser	7
achieve	5
acquire	3
acquisition	3
acres	3
across	4
act	4
acted	3

Word Distribution in Hash Table

The bars represent the number of Unique words in each index of the Hash Table

Index	Total Words
0	264
1	236
2	397
3	84
4	306
5	477
6	231
7	341
8	322
9	601

Word Prediction (Probability Calculation)

Input Inital words

The bottom section displays the most likely 20-word sequence upon entering input. There are two options: one using Bigram and the other using Trigram. The results for each option are shown in different text fields. Clicking the "Calculate using Bigram" button displays the next most likely 20 words in the first text area, and clicking "Calculate using Trigram" shows the next most likely 20 words in the second text area. The "Clear" button erases both the input and output, allowing users to enter new data afresh.

The final crucial feature is the "Change Hash Function" option at the top. It enables users to switch the hash function during runtime. In the current assignment, two hash functions are available: the Simple Hash Function, provided in the assignment, and the Polynomial Hash Function, explained in detail below. Clicking on "Change Hash Function" toggles between the Simple and Polynomial hash functions. The entire dataset is then reprocessed with the new hash function, and the updated results are displayed in the UI.

Java Programming Concepts and Framework Utilisation

Data Abstraction is employed in the classes MyHashTable and MyLinkedObject, hiding the internal details of how a hash table or a linked list is created. Instead, these classes provide functionality for adding and retrieving data in both the case of a hash table and a linked list.

Polymorphism is effectively applied in the hash function classes. The MyHashFunction class acts as a superclass, establishing a shared interface for various hash functions. Both the SimpleHashFunction and PolynomialHashFunction classes serve as subclasses, each implementing its version of the "public int hash(String word);" function.

Encapsulation is applied in both the MyHashTable and MyLinkedObject classes. This involves declaring data members as private and interacting with them through public methods. For instance, in the MyLinkedObject class, the word, count, and next variables are kept private and accessed or modified using dedicated getter and setter methods.

Inheritance: Although I have not directly implemented inheritance, I applied the concept by creating panels for various UI components like ProbabilityCalculationPanel, InputReadFilePanel, WordAndCountTablePanel, etc. These panels inherit from the JPanel class, allowing the creation of custom panels with properties inherited from JPanel.

Additional concepts, such as **Composition**, involve the Frame class containing a (“has a” relationship) BarGraphPanel and a DistributionBarPanel. There is also an **Association** where MyHashTable class utilises a MyHashFunction class to calculate hash values. It's important to note that these two objects have separate lifecycles. Additionally, the assignment employs the use of **Anonymous classes** to set addActionListener for Buttons.

The **Collection Framework**, including classes like Map and TreeMap, as well as utilities like Comparator, is primarily used for sorting words alphabetically and in descending order of frequency count in the “WordAndCountTablePanel” class.

In the **GUI implementation** (Frame.java), I've utilised Swing components like JFrame, JPanel, JButton, JFileChooser, and SwingWorker. Additionally, custom panels (BarGraphPanel and DistributionBarPanel) extend JPanel. The ProbabilityCalculationPanel and WordAndCountTablePanel classes also extend JPanel and incorporate various Swing components.

For **Exception Handling**, I've employed this concept in the validation of input during the prediction of the most likely 20 words. I throw an IllegalArgumentException in methods like get20MostFrequentWordsUsingBigram and get20MostFrequentWordsUsingTrigram if the constraints are not met. The exceptions are then caught (in ProbabilityCalculationPanel class) to display appropriate messages in the UI.

User Interface Design and Features

MyLinkedObject Methods

1. Set and get methods of data members “word”,”count” and “next”
    a. public void setNext(MyLinkedObject newNext)
    b. public MyLinkedObject getNext()
    c. public void seCount(int count)
    d. public int getCount()
    e. String getWord()
    f. public int setWord(String w)

    The setWord method is used to insert a word into a linked list. It compares the input word with the current word. If they are equal, it increments a count. If the input word is greater, it inserts the word further into the list. If it is less, it inserts the word before the current node. The method returns 0 if the word was already in the list and 1 otherwise. This return type indicates that a new word has been added.
2. Clone method *“public MyLinkedObject clone()”*

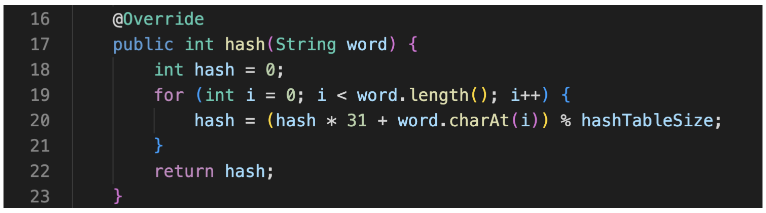
   The clone method in the MyLinkedObject class creates a new object with the same word, count, and next values as the current object and returns this new object

3. toString "*public String toString()*"

   The toString method in MyLinkedObject class returns a string representation of the object, combining the word and count fields, separated by " | ", and ends with " | --->". This is used for debugging purpose

Choice of hash function algorithm | Polynomial Hash Function

The Below hash function is inspired by the method toHashCode of Java, and taken from https://computinglife.wordpress.com/2008/11/20/why-do-hash-functions-use-prime-numbers/



```java
    @Override
    public int hash(String word) {
        int hash = 0;
        for (int i = 0; i < word.length(); i++) {
            hash = (hash * 31 + word.charAt(i)) % hashTableSize;
        }
        return hash;
    }
```

assignment/hash_table/PolynomialHashFunction.java

The hash method in the PolynomialHashFunction class calculates a hash value for a given word by iterating over each character in the word. It multiplies the current hash by 31, adds the ASCII value of the character, and takes the modulus of hashTableSize to ensure the value remains within the range of 0 to hashTableSize. The resulting hash value can be used as an index into the hash table.

Encapsulating MyLinkedObject and MyHashFunction

This has been accomplished by employing package and default access specifiers. The "hash_table" package in the assignment encapsulates the classes MyLinkedObject and MyHashFunction (SimpleHashFunction and PolynomialHashFunction). These classes are only accessible within this package, ensuring restricted visibility, i.e., they are "package-private" classes. In contrast, the MyHashTable class employs a public access specifier, enabling it to be accessed externally. Consequently, MyLinkedObject and MyHashFunction can only be utilised outside the package through the MyHashTable object.

Sorting Strategy: Alphabetical & Descending Frequency

To effectively implement this feature, I have made use of TreeMap<String, Integer>, where the key (String) represents the word and the value (Integer) represents the frequency.TreeMap maintains the elements in sorted order based on their natural ordering or according to a specified comparator.

1. Alphabetical Sorting:

   TreeMap<String, Integer> map = new TreeMap<>();

   The TreeMap sorts entries based on their natural ordering by default, When elements are added, the words are sorted alphabetically.

2. Frequency-Based Sorting:

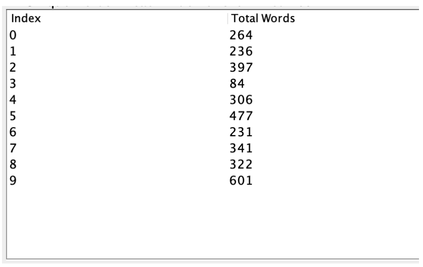
   TreeMap<String,Integer>frequencyOrderMap=new TreeMap<>(Comparator.<String,Integer>comparing(map::get).reversed().thenComparing(Comparator.naturalOrder()));

To sort based on frequencies, I have to provide a custom comparator to sort entries by their values (frequencies). This comparator first compares entries based on frequency in descending order (reversed()), and in case of a tie (equal frequencies), it resorts to natural order.

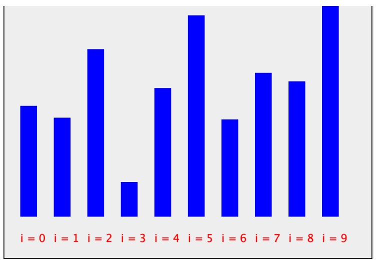
When Creating Vocabulary list one thing which is noticeable for the given data set most of the high frequency words are These are typically common words known as "stop words" like "the," "to," "a," "of," and "and" with more data set we can get more data which can be anlyzed

Analysing Hash Function Algorithms : Linked List Statistics and Observations

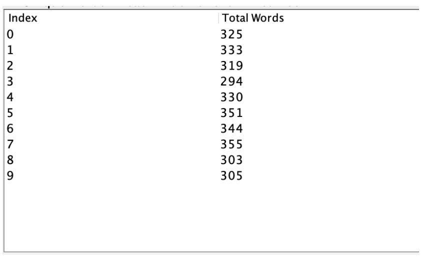
1. SimpleHashFunction: When using this hash function, the words are unevenly distributed in the hash table. For example, as shown in the image, indices 5 and 9 have more words compared to other indices



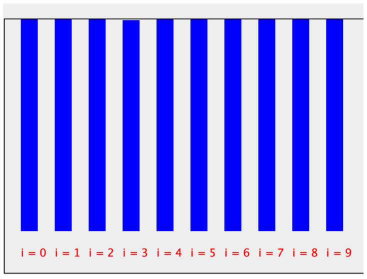
Index	Total Words
0	264
1	236
2	397
3	84
4	306
5	477
6	231
7	341
8	322
9	601



2. PolynomialHashFunction: When using the polynomial hash function, the words in each index are evenly distributed, providing a more balanced way to distribute the data



Index	Total Words
0	325
1	333
2	319
3	294
4	330
5	351
6	344
7	355
8	303
9	305



When comparing both hash functions, PolynomialHashFunction is the better choice. PolynomialHashFunction considers all characters of the given word to calculate the hash, unlike SimpleHashFunction, which only considers the first character. This results in higher collision resistance

When **adjusting hash table sizes** for Simple and Polynomial Hash Functions, both maintain consistent averages. However, the Polynomial Hash Function demonstrates lower standard deviations, ensuring more controlled word distribution. The choice of hash function has a greater influence than table size, highlighting the Polynomial Hash Function's stability and predictability in distribution.

Sorting words by frequency improves accessibility, enhances pattern recognition and trend spotting, outperforming alphabetical sorting. This optimization significantly boosts overall efficiency in processing and interpreting textual data

Analysing N-Gram Models in Natural Language Processing

When predicting the top 20 words using unigram and bigram probabilities (p(w2|w1)), we can skip calculating p(w1). With the user inputting at least one word (wk-1), we can directly compute the probability of wk given wk-1 using *p(wk|wk-1) = c(wk,wk-1) / c(wk-1).*

Similarly, for trigrams, omitting the need for p(w1) and p(w2|w1), if the user provides at least two words (wk-2 and wk-1), we can determine the probability of wk given wk-2 and wk-1 as *p(wk|wk−2,wk−1) = c(wk−2,wk−1,wk) / c(wk−2,wk−1).*

Predictive Analysis:

1. Using Unigrams and Bigrams :

   “you have” : “you have to be a lot of the president clinton when you know what the president clinton when you know what the”

   “this is one” : “this is one of the president clinton when you know what the president clinton when you know what the president clinton when you”

2. Using Trigrams

   “you have” : “you have to have better investment policies you have to have better investment policies you have to have better investment policies you”

   “this is one” : “this is one of the c. i. a. station chief colby became director of the c. i. a. station chief colby became director”

When entering "today in Sheffield," the system fails to generate the next 20 likely words using unigrams, bigrams, or trigrams. An error message in the UI states, "Cannot predict the next word since 'today in Sheffield' is not in the dataset." This issue arises because predicting the next word (wK+1) requires the presence of the preceding word (wK). In this context, "Sheffield" is absent, leading to prediction failure. In contrast, inputting "in Sheffield today" successfully predicts subsequent words, leveraging the presence of the preceding word "today."

Challenges of Larger n-grams:

Implementing n-grams with a value greater than '3' presents significant challenges, primarily due to memory constraints and computational complexity. Firstly, considering more than 4 words in a sequence increases uniqueness, demanding more memory—impractical for large datasets. Secondly, calculating probabilities for larger n-grams becomes computationally intensive, impacting real-time processing and user experience