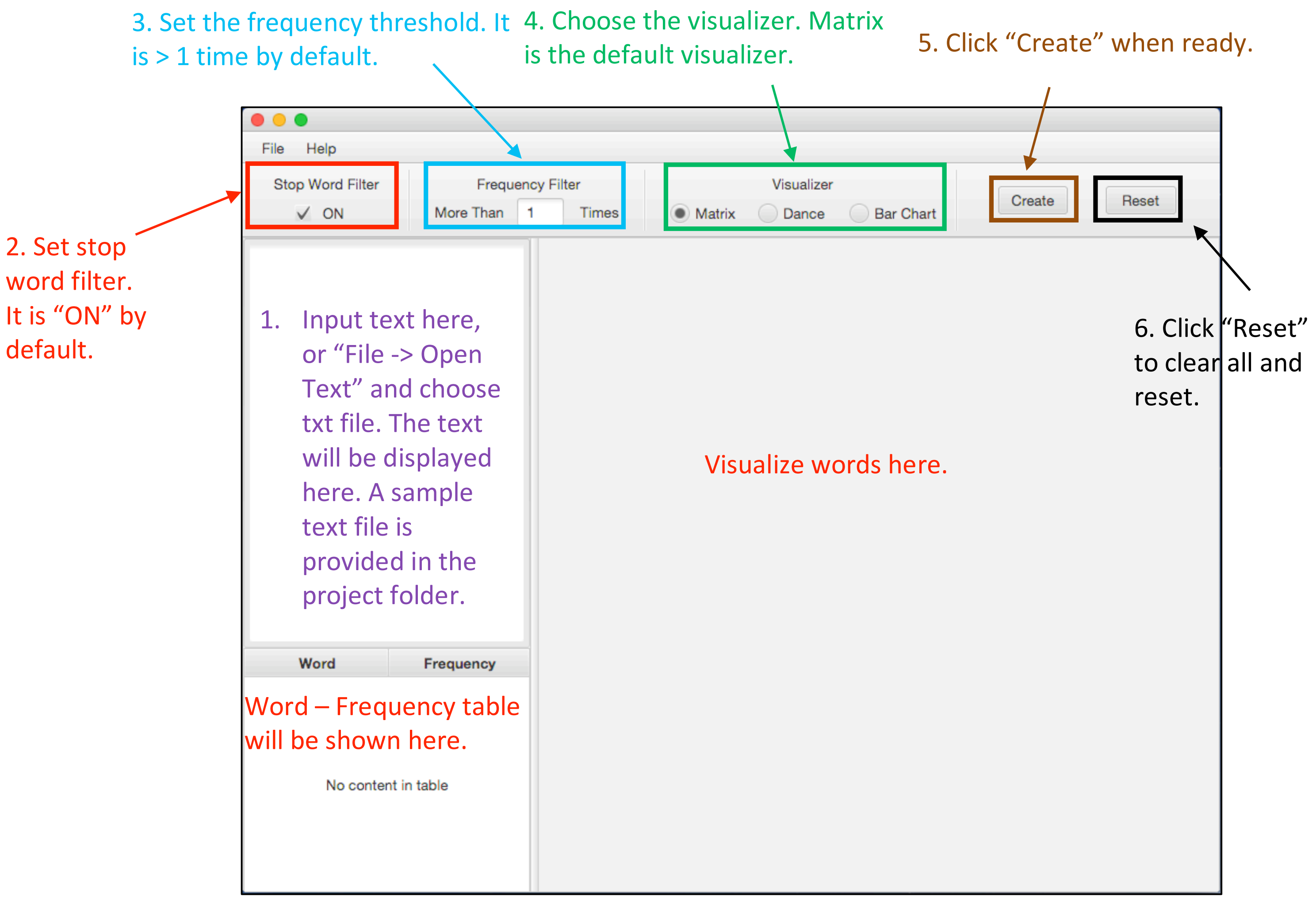
# Project Documentation

**Description**

This program was developed to count the frequency of each word in a text and visualize the words based on their frequencies in a dynamic way or in a bar chart. The inspiration comes from Wordle – an online static word cloud program (www.wordle.net). The features Wordle is missing are that it cannot show a dynamic word cloud and cannot report the exact occurrence frequency of each word. This project addresses these issues by creating dynamic visualizations and displaying the frequency of each word in table and bar chart to user.

**How to use the program**

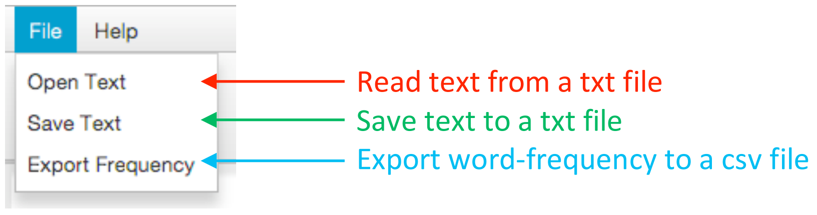
Main UI


3. Set the frequency threshold. It is > 1 time by default.
4. Choose the visualizer. Matrix is the default visualizer.
5. Click "Create" when ready.
File
Help
Stop Word Filter
✓ ON
Frequency Filter
More Than
1
Times
Visualizer
Matrix
Dance
Bar Chart
Create
Reset
2. Set stop word filter. It is "ON" by default.
6. Click "Reset" to clear all and reset.
1. Input text here, or "File -> Open Text" and choose txt file. The text will be displayed here. A sample text file is provided in the project folder.
Visualize words here.
Word
Frequency
Word – Frequency table will be shown here.
No content in table

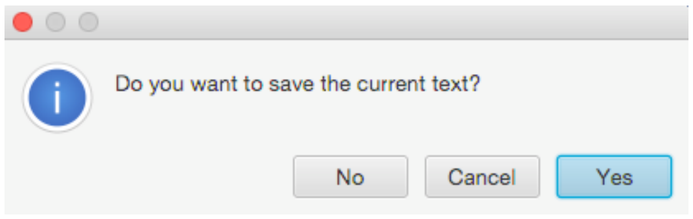
**Note:**

If user changes the settings, such as Stop Word Filter, Frequency Filter, and Visualizer during the display, the user needs to click "Create" button to make the changes take effect.

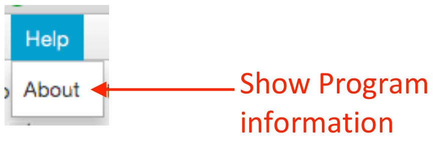
File Menu

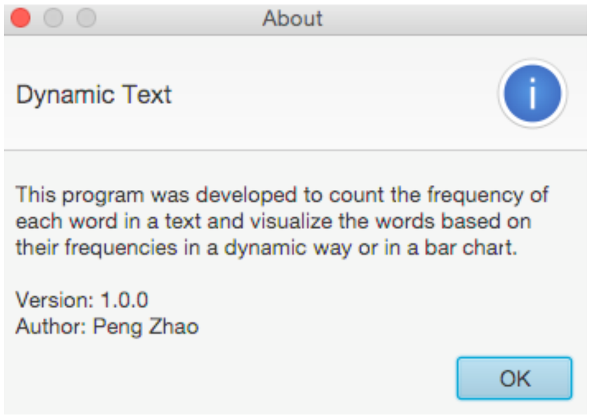


If user clicks “Open Text” while there is text in the text input area, a message dialog will show to the user with 3 buttons: No, Cancel, and Yes. If user clicks “Yes”, he/she will be directed to the save dialog to save the current text. If user clicks “No”, he/she will be directed to the open dialog to open a text file and overwrite the current text. If user clicks “Cancel”, nothing will happen, as he/she decides to cancel the open operation.


Do you want to save the current text?

Help Menu




About

Dynamic Text

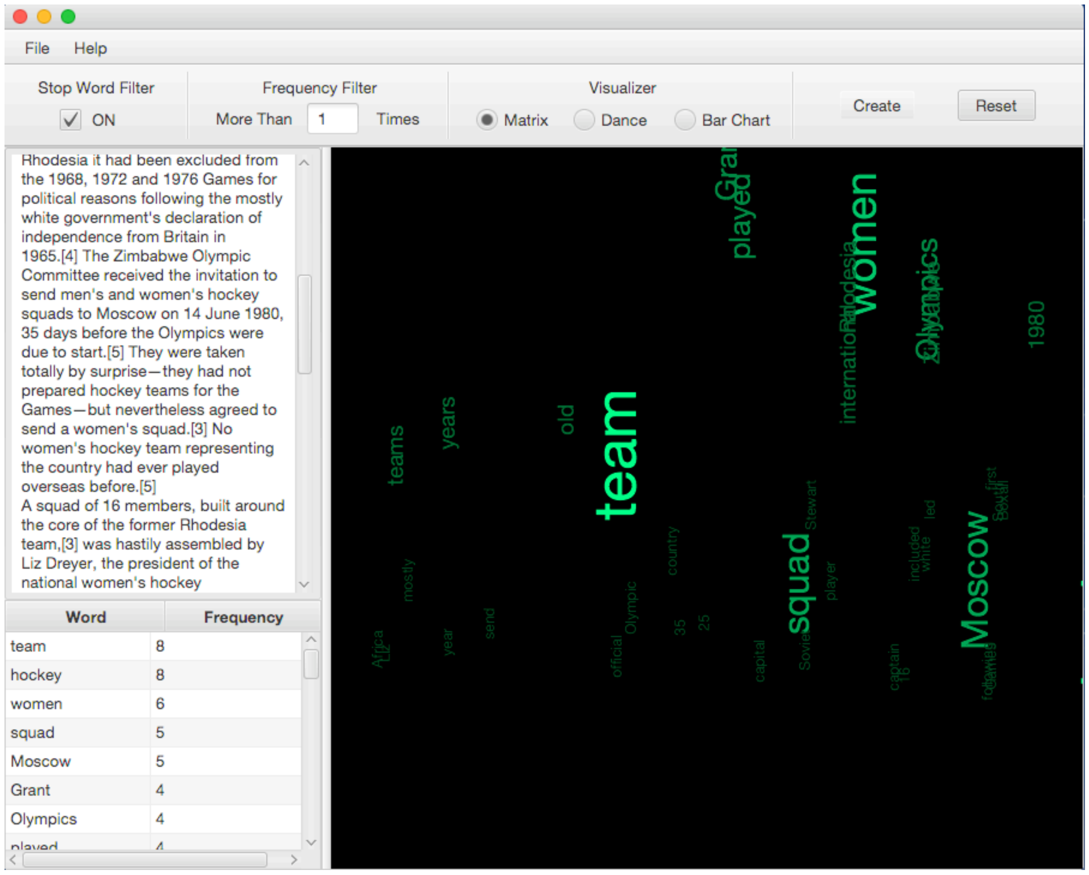
This program was developed to count the frequency of each word in a text and visualize the words based on their frequencies in a dynamic way or in a bar chart.

Version: 1.0.0
Author: Peng Zhao

**Example:**

Analyze the “Invitation and team selection” part in article “Zimbabwe women’s national field hockey team at the 1980 summer Olympics” at Wikipedia.org.

The sample text file is provided in the project folder.



Stop Word Filter: ✓ ON

Frequency Filter: More Than 1 Times

Visualizer: ● Matrix ○ Dance ○ Bar Chart

Rhodesia it had been excluded from the 1968, 1972 and 1976 Games for political reasons following the mostly white government's declaration of independence from Britain in 1965.[4] The Zimbabwe Olympic Committee received the invitation to send men's and women's hockey squads to Moscow on 14 June 1980, 35 days before the Olympics were due to start.[5] They were taken totally by surprise—they had not prepared hockey teams for the Games—but nevertheless agreed to send a women's squad.[3] No women's hockey team representing the country had ever played overseas before.[5]
A squad of 16 members, built around the core of the former Rhodesia team,[3] was hastily assembled by Liz Dreyer, the president of the national women's hockey

| Word | Frequency |
|---|---|
| team | 8 |
| hockey | 8 |
| women | 6 |
| squad | 5 |
| Moscow | 5 |
| Grant | 4 |
| Olympics | 4 |
| played | 4 |

**How the program meets requirements**

1. This program was developed in Java 8 SE with NetBeans IDE and JavaFX Scene Builder. The interface was developed based on FXML.

2. This program was built on the MVC architecture.

3. Object oriented elements

   a. Classes
   This program contains 9 classes:
   **Pz72bDynamicText.java**: entry point of this program.
   **FXMLDocumentController.java**: controller class to link the UI elements with different functions.

**FreqCalculator.java**: counts the occurrence of each word in the text with or without stop words and return a frequency-descending map with words as keys and frequencies as values.
**WordFreq.java**: gets the word and frequency of each entry in the returned Map. It is used to populate the word-frequency table in the UI.
**StopWord.java**: contains a list of stop words from: https://javaextreme.wordpress.com/category/java-j2se/java-string/remove-stop-words-from-a-string/
**Visualizer.java**: abstract class with two abstract methods "create" and "cleanUp".
**ChartVisualizer.java**: show words and frequencies in a bar chart.
**DanceVisualizer.java**: display words in a way that the words are dancing. The font size and color hue are proportional to frequencies.
**MatrixVisualizer.java**: display words in a way like the digital rain in the movie "Matrix". The font size and brightness are proportional to frequencies. The falling speed is inversely proportional to frequencies.

b. Subclasses
"ChartVisualizer.java", "DanceVisualizer.java", and "MatrixVisualizer.java" are subclasses of abstract class "Visualizer".

c. At least one abstract class and/or interface
"Visualizer.java" is an abstract class, which contains abstract methods "create" and "cleanUp".

4. Code elements

a. One or more collection classes
"FreqCalculator.java" uses ArrayList, Map, HashMap, LinkedHashMap, List.

b. Exception handling
In "FXMLDocumentController.java" class, "openHandler", "saveHandler", and "exportHandler" methods use exception handling.

5. The application must have a clearly defined model

The "FreqCalculator.java" class is the core of this program and it directly manages the data of the program. The behavior of it is independent of View and Controller.

6. The UI must utilize multiple scenes and/or a scene where the contents of the scene graph are changed based on application state.

The scene graph will change when the user switches the visualizer.

7. There must be a way to access "About" information that includes information about you

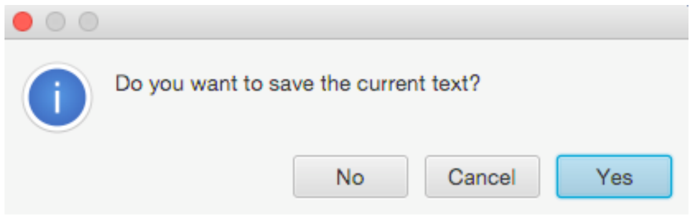
and the application.

The “About” information will be displayed as a dialog if the user clicks “About” in the “Help” menu. The “aboutHandler” method in “FXMLDocumentController.java” class handles this “About” information.

8. The application must save data and load data.

   User can save the text as a txt file to local drive and read text from a txt file from local drive. User can also export the word-frequency information to a csv file to local drive. “openHandler”, “saveHandler”, and “exportHandler” methods in the “FXMLDocumentController.java” class handle the open, save, and export.

   If user clicks “Open Text” while there is text in the text input area, a message dialog will show to the user with 3 buttons: No, Cancel, and Yes. If user clicks “Yes”, he/she will be directed to the save dialog to save the current text. If user clicks “No”, he/she will be directed to the open dialog to open a text file and overwrite the current text. If user clicks “Cancel”, nothing will happen, as he/she decides to cancel the open operation.


Do you want to save the current text?