Documentation for the “Words Statistics Analysis Tool” application.

Authors: Everest Alonso, Patrik Homola, Kamil Pecela

Project developed for the course “Development of Information Systems”

Faculty of Mathematics, Physics and Informatics

Comenius University Bratislava

2023-2024

# Introduction

This document was created within the subject Development of Information Systems in 2023/2024 and serves as a complete design of the application “Words Statistics Analysis Tool”. It contains information necessary for the explanation and understanding the functionality as well as the method of system implementation. This document refers to a requirements catalog. The following chapters will present the implementation plan, system architecture description, and test scenarios.

# Implementation Plan

## User Interface Design

Design the graphical user interface (GUI) with an initial page containing three buttons for main functionalities. Create separate pages for uploading text, deleting statistics file, and viewing statistics and producing graphs.

## System Launch

Develop the executable application. Implement the functionality for the user to open the system by clicking the executable app.

## Text Upload Functionality

Design and implement the "Upload text" button on the initial page. Create a new page for uploading text with fields for choosing a text file, adding author name, title, and optional description. Show disclaimers during the uploading process. Implement the transformation of the text into an array of words. Count the frequency of consecutive word pairs. Save results in a special directory as a \*.txt file with utf-8 encoding, named after the author and title.

## File Deletion Functionality

Implement the "Delete file" button on the initial page. Create a new page for file removal, listing available statistics files. Allow the user to choose and delete a file, with confirmation. Automatically redirect the user to the initial page after deletion. Log the file deletion operation with proper details in a log file.

## View Statistics and Produce Graph Functionality

Implement the "View statistics and produce graph" button on the initial page. Create a new page for viewing statistics with options to choose a file, search by a word or pair of words, and choose display options. Allow the user to produce a graph, tick checkboxes for specific options, and choose a logarithmic scale. Display the results on a new page with a bar graph. Provide the option to save the results in image format to a chosen directory. Include a "Go to start" button at the bottom of the page for redirection to the initial page.

## Overall System Flow

Implement the overall flow to redirect the user to the initial page after each operation (upload, delete, view). Ensure proper error handling and user feedback throughout the application.

## Testing

Conduct thorough testing for each functionality, including positive and negative scenarios. Debug and refine the application based on testing results.

## Documentation

Document the implementation details, including code structure, functions, and modules. Create user documentation explaining how to use the application.

## Deployment

Package the application for deployment on relevant platforms.

## Maintenance

Plan for ongoing maintenance, updates, and support.

# **Architecture & Design Overview**

Application Overview:

The application is designed as a text analysis tool implemented in Python using the Tkinter library for the graphical user interface. It allows users to upload text files, perform various analyses, and visualize the results through a user-friendly interface.

## Modules:

1. Main Application Module (main.py):
   * Initializes the Tkinter application.
   * Manages different frames representing various sections of the application.
   * Handles the overall flow of the application.
2. Frames:
   * BaseFrame:
     + Serves as the base class for other frames.
     + Implements common functionality like switching frames and updating.
   * MainMenu:
     + Displays the main menu with buttons to navigate to different functionalities.
   * UploadText:
     + Allows users to upload text files.
     + Collects information like name and description for the uploaded text.
     + Performs analysis on the uploaded text.
   * DeleteText:
     + Enables users to delete previously uploaded texts.
     + Lists available texts for deletion.
   * SelectToAnaliseFrame:
     + Provides options to select a text for analysis.
     + Takes input for words and parameters for analysis.
   * StatsFrame:
     + Displays general information about the selected text.
   * SearchPairFrame:
     + Extends StatsFrame.
     + Displays results for a search pair analysis.
   * Analyses:
     + Base class for specific analyses frames.
     + Contains a scrollable table and a bar chart for visual representation.
   * FullAnalyses:
     + Displays results for a full analysis.
   * ForWordAnalyses:
     + Extends Analyses.
     + Displays results for a word-specific analysis.
3. Supporting Modules:
   * classQueryModule:
     + Implements the QueryModule class handling text file analysis queries.
   * Analysis:
     + Provides functions for analyzing text files and counting word connections.
   * file\_check:
     + Checks if the uploaded file is compatible (UTF-8 encoded text).
   * SaveLoadSystem:
     + Handles saving and loading objects using pickle.
   * scrollableTable:
     + Implements a scrollable table and bar chart using Tkinter and Matplotlib.
4. Data Structures:
   * TextPath:
     + Represents a text file's path, title, and description.

## Data Flow

1. User uploads a text file through the UploadText frame.
2. The application performs analysis on the uploaded text using the Analysis module.

## Query Module

1. Loads data from the selected text file.
2. Runs queries based on user input (word searches, pair searches, full analysis).
3. Returns results to the respective frame for visualization.

## Graphical Interface

* The Tkinter library is used to create a responsive and intuitive GUI.
* Frames are dynamically switched based on user interactions.

## Extensibility

* The modular design allows for easy addition of new analysis functionalities or frames.
* QueryModule can be extended to include additional query types.

Conclusion: The architecture follows a modular and extensible design, providing a user-friendly interface for text analysis. The combination of Tkinter for GUI, analysis modules, and supporting functions creates a robust tool for exploring relationships within text data.

# **Testing Scenarios**

## Uploading Text

* Scenario 1: Successful Upload
  1. User selects the "Upload Text" option from the main menu.
  2. User selects a valid text file (UTF-8 encoded).
  3. User provides a unique name and description for the text.
  4. User clicks the "Upload" button.
  5. Verify that the application processes the file without errors.
  6. Verify that the new text appears in the list of available texts.
* Scenario 2: Upload Failure
  1. User selects the "Upload Text" option from the main menu.
  2. User selects an invalid file (non-UTF-8 encoded or unsupported format).
  3. User provides a name and description for the text.
  4. User clicks the "Upload" button.
  5. Verify that the application displays an appropriate error message.
  6. Verify that the text list remains unchanged.

## Deleting Text

* Scenario 1: Successful Deletion
  1. User selects the "Delete Text" option from the main menu.
  2. User selects a text from the list for deletion.
  3. User clicks the "Delete" button.
  4. Verify that the application prompts for confirmation.
  5. User confirms deletion.
  6. Verify that the text is removed from the list.
* Scenario 2: Cancel Deletion
  1. User selects the "Delete Text" option from the main menu.
  2. User selects a text from the list for deletion.
  3. User clicks the "Delete" button.
  4. Verify that the application prompts for confirmation.
  5. User cancels the deletion.
  6. Verify that the text list remains unchanged.

## Selecting Text for Analysis

* Scenario 1: Successful Selection
  1. User selects the "Select Text to Analyze" option from the main menu.
  2. User chooses a text from the list.
  3. User enters valid words and parameters for analysis.
  4. User clicks the "Analyze" button.
  5. Verify that the analysis results frame is displayed.
  6. Verify that the correct analysis results are shown.
* Scenario 2: Invalid Selection
  1. User selects the "Select Text to Analyze" option from the main menu.
  2. User does not choose any text or enters invalid parameters.
  3. User clicks the "Analyze" button.
  4. Verify that the application displays an error message.
  5. Verify that the analysis results frame is not displayed.

## General Interaction:

* Scenario 1: Navigation
  1. User navigates through different frames using the provided buttons.
  2. Verify that frames switch correctly based on user interactions.
* Scenario 2: Closing the Application
  1. User clicks the close button.
  2. Verify that the application closes without errors.