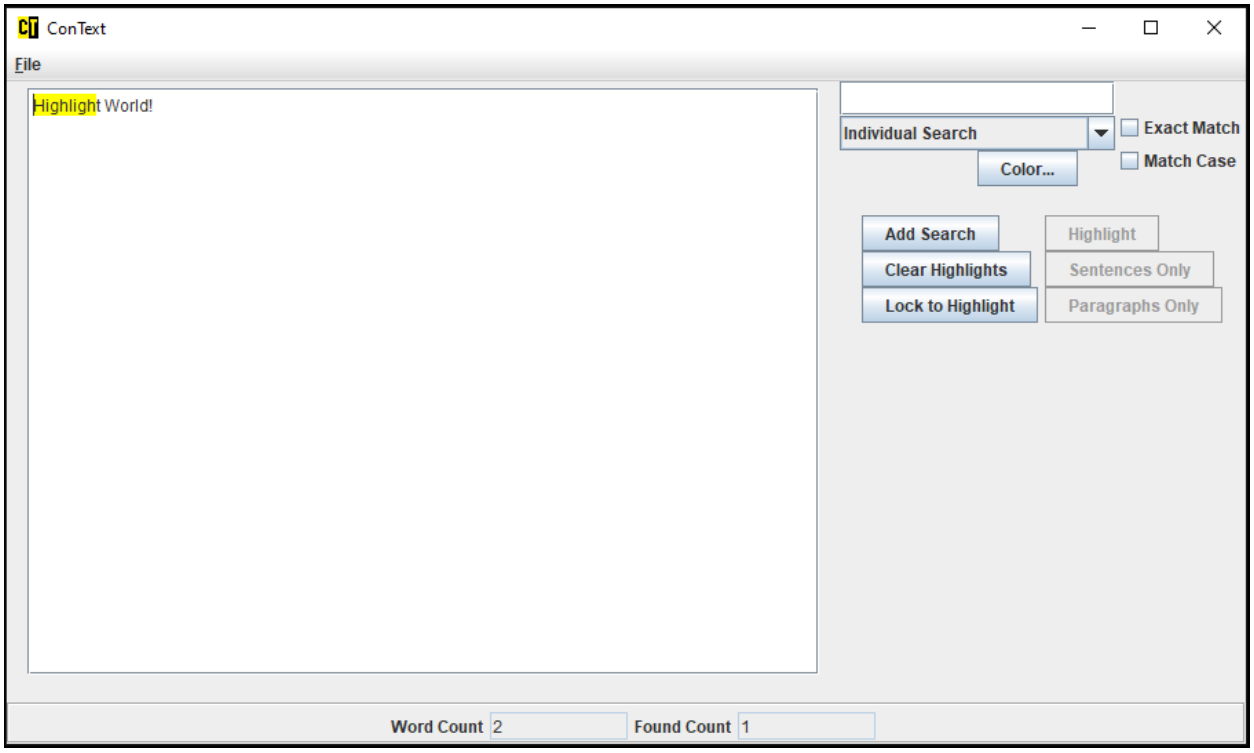
Final Project Report

document search



CMSC 495-7380
December 11, 2021
Yousaf Chaudry, John Ingle, Michelle March, Patrick McCully
ConText website

Table of Contents

Table of Contents

Overview

ConText was conceived as a program that would facilitate the task of finding connections within a given document for the end-user. The premise is simple: how could we make it easier to determine if two (or more) words or phrases appear together in the same document? Maybe not right next to each other, but not fifty pages apart, either. More than a simple 'Ctrl-F', this idea compares at least two sets of string-type input data, and tries to confirm whether the connection the user is looking for exists. The parameters for this search can be defined as:

1) within the whole document,
2) within the same sentence, and
3) within the same paragraph.

For any kind of document that requires analysis, the possibilities are infinite.

Member Contributions:

Michelle – The team leader of ConText, she was responsible for back-end data management and manipulation and helped to refine the GUI. She also ensured that features were as complete and thorough as possible within our given time constraints.

John – Our point man for GitHub integration, using the IntelliJ IDE, and figuring out how to do the things none of us had done before. He constructed the GUI interface and developed the Export function. He also provided insight into whether theorized features would be feasible to implement and made us all want to get better at GitHub.

Yousaf – The glue that held our team together. Assisting with back-end data manipulation and documentation, he also kept track of program changes, features that end-users might look for, and kept everyone aware of upcoming deadlines and requirements. His most frequent contributions were the ones we didn't know we needed, especially in demonstrating and inspiring better communication.

Patrick – Our Quality Assurance tester, he helped to ensure that the program features actually worked as they were supposed to, recording any issues found within a log file we designed for the purpose. As our project developed, we were shockingly grateful at his unparalleled ability to break whatever we thought was unbreakable.

Table of Contents

Project Plan

Main Window
allows users to pick a file to read or enter text manually
searches document for user-specified search terms
displays text with multicolor and contextual highlights
can shorten document to relevant sections by removing sentences or paragraphs without highlights

File Selection
allows the user to select a file from a local drive

Export
allows user to save shortened version with or without highlights

---

Hardware Interfaces

This application requires the use of a mouse and keyboard or a touch screen to interact with the application user interface. A monitor or display is required to view the application.

Software Interfaces

The user must have operating system level permissions to read files on disk (ex: the user must have the ability to open a text document). Administrators should allow the user to interface with this software by granting them permission to run this application.

Table of Contents

System Features

Multicolor Text Highlight

Description and Priority

- High priority - primary function of product.
- Users are given a set of text boxes to enter search terms (strings). Each box is associated with a user-selected color that will be used to highlight instances of that string.

Stimulus/Response Sequences

- User enters search term (string) in text box
- User selects color to highlight each term from dropdown menu associated with text box
- User clicks “Search” button
- Program identifies instances of terms
- Program displays document with identified words highlighted with user-specified colors

Functional Requirements

- Multicolor word highlight must be available
- If user enters the same word more than once with different colors, default will be to highlight using last color
- If user enters search terms without selecting a color, default will be to highlight in next unused color
- If user enters no search terms, default will be to display document with no highlights

---

Contextual Search

Description and Priority

- High priority - primary function of the product.
- The user is given the option to search based on proximity. Users can specify that search keywords should be within a certain number of words, sentences, or paragraphs from each other.

Stimulus/Response Sequences

- After entering keywords and selecting colors, user specifies context search

Functional Requirements

- If more than one search term is entered, instances of at least two (2) terms will be identified.

Table of Contents

Exact Match and Match Case

Description and Priority

- High priority - standard feature
- Allow users to specify that instances of search term should be considered to match only if words match exactly, as well as with or without regard to capitalization

Stimulus/Response Sequences

- After entering search term(s) and specifying highlight color, the user selects “Match Case”
- During search, only instances that match capitalization of search terms are highlighted
- After entering search term(s) and specifying highlight color, the user selects “Exact Match”
- During search, only instances that match search terms exactly are highlighted

---

Display and Export

Description and Priority

- Medium priority
- Export allows the user to save the document based on identified instances of search terms. The user may choose to include highlights in the exported file. The exported file may include sentences or paragraphs including identified terms, as specified by the user.

Stimulus/Response Sequences

- After entering and searching for terms, the user may remove sentences or paragraphs without matches
- The user may select HTML export to retain highlights or TXT export otherwise
- User is prompted for file name
- In a new window, the saved document is displayed

Functional Requirements

- If the user does not remove sentences or paragraphs, full text will appear.
- If the user has not searched and/ or found any terms, the text in the text area will be saved as-is

Table of Contents

Scenarios in Which Software May Be Used

Scenario One

A patent lawyer has to compare the design specifications (spec B) of a certain product against the design specs (spec A) of a rival company's product. Before submitting a patent application, the lawyer knows the chances of his client's success are much better if their product can be sufficiently differentiated from the competition. The design spec documents for each product are over 50 pages each.

Using the ConText software, he is able to search for the exact parameters that apply to both products. For example: the voltage used when their product is plugged in while under expected use. The steps to accomplish this can be done by loading spec B into the ConText program, entering the text that would include mention of voltage inside one of the text boxes, and then executing the Export function that displays a new window with the results. These results show details such as number of matching hits and their location within the document.

While keeping this Export window open, the second spec A document is loaded into the main ConText GUI. The same steps are repeated here, and the Export function will bring up a new Results window that contains the findings of this second search. Side-by-side, the lawyer now has all the information needed to either approve this stage of his client's product, or inform them that it requires revision according to this particular parameter.

Scenario Two

A customer recently purchased a very expensive TV, but two weeks into using it, some strange flickering is noticed. She opens the User Guide on her laptop, available as a PDF file, and wants to find information on flashing of the screen.

Using traditional find and replace, she sees that there are over 100 instances of the word "flashing" in the User Guide, many of which refer to normal behavior. She instead tries the word "screen" and finds many more instances of the word throughout the document. Finally she tries putting the words together and searches both "flashing screen" and "screen flashing", but finds no occurrence of the words next to each other.

She loads the document into the ConText Document Search and enters "screen" in the first search box and "flashing" in the second search box. She then chooses to return all occurrences of the words within 1 paragraph of each other. ConText returns only fifteen instances of both words in the same paragraph and she is now able to quickly find the information she is looking for.

Table of Contents

Scenario Three

A student has been referred to a textbook by their professor, but they think the chapters the professor referenced are not correct. They are supposed to discuss “clarification methods for ambiguity” in their assignment, but the Chapter doesn’t seem to mention anything about that. The student would like to try to find the correct chapter on their own.

Using a standard search they find that the words “clarification”, “methods”, and “ambiguity” appear many times throughout the 400-page textbook. They try to search several combinations of the words and even synonyms of the words, but find that the results are all either far too many or far too few.

They open the document in ConText and enter their search terms in separate boxes, then select that they want to find all instances within 1 sentence of each other. They find only a few instances and are then able to identify the correct chapter, as well as further information in other chapters.

The student then decides to save the report. They select to maintain paragraphs, but not highlights and are then able to view a two-page document covering the topic they need to discuss and no unrelated topics.

Table of Contents

Project Risks, Constraints, and Assumptions

This software package requires that developers only use Java libraries and APIs compatible with 64 bit operating systems listed in the Specifications section. Platform specific or low level API calls should be avoided to ensure the application functions properly on all major operating systems. This project also depends on having a JDK runtime with the Swing library installed on the user's machine, as well as the PDFbox jar. For production, the application will bundle a copy of the JDK and the required libraries in the application's installation folder, along with the requisite tmp directory and a PDF user guide.

One of the major risks present in this project's development is the potential overexpansion of scope, due to how many features or functions could be added. There are so many areas where small tweaks or even major program overhauls could be thought of, that it will be important to keep in mind the original purpose of ConText and make an attempt to keep it as simplified and user-friendly as possible.

Another risk lies in algorithmic complexity resulting in reduced runtime performance during common use cases. It's often easier to test using small, common files, but the average user is more likely to want to search large, intricate documents for related words than short, simple ones (where the average person could find the related words quickly, without computational assistance). It is important to remember that tests that are easy to confirm may not be indicative of normal use behaviors.

Table of Contents

Key Milestones and Timeline:

Week 1 - Designated staffing roles for each member

Michelle	back-end data manipulation, GUI refinement
John	GUI construction, code integration/dissemination within IDE and Git
Yousaf	assist back-end manipulation, compose documentation
Patrick	quality assurance testing, reporting of any bugs and unintended software performance

Week 2 - Program Specifications created – contributed by all members

Week 3 - User Guide and Test Plan – contributed by all members

Test files

User Guide

Week 4 - Design – contributed by all members

Class Hierarchy - overall project structure

Public variables - those in Main, accessible by other classes

Data structures - how to store words and search keys

Input and output parameters (primarily document type and search key data)

Week 5 - Software 1/4 - contributed by Michelle, John

GUI framework

- main window display
- all buttons respond to click (visual or action)
- all checkboxes toggle on and off
- all text fields and areas allow user input
- all drop-down menus open and allow selection
- “File Opener” allows file selection
- file contents displayed as text in main GUI

Estimated cumulative testing time (including report): 1-2 hours

Table of Contents

Week 6 - Software 2/4 - contributed by Michelle, John

- Accept document
- Be able to pass data between classes
- Word search functionality
- Single word search
- Multi-word search
- Proximity word search
- Color change for single word search
- Multi-color change for multi-word search
- Export functionality
- Export displays with abbreviated document
- highlights maintained/ removed based on user selection

Estimated cumulative testing time (including report): 1-2 hours.

Week 7 - Software 3/4 - contributed by Michelle, John, Yousaf

- Most functions working properly
- Class interdependencies verifiably functional
- Thorough testing to find bugs
- Complete functionality
- GUI displays as expected
- “File Opener” allows appropriate file types to be read and displayed - text boxes and areas allow user to type in words
- all checkboxes toggle on and off
- all drop-down menus open and allow selection
- single-, multi-, and proximity- search perform as expected
- single- and multi- word highlighting performs as expected
- “Search” button performs as expected
- “Export” button and selections perform as expected
- document saving is available and performs as expected

Table of Contents

Week 8 - Software Final - contributed by Patrick, Michelle, John, Yousaf

- All functions of application perform as required
- Final polishing of look and feel
- All {known} bugs fixed
- Finalized documentation

General Milestones for Each Week

- All code for the milestones for the week will be available for testing by EOD Thursday any potential issues or priority testing should be emailed to Patrick. Error reports from testing will be posted by EOD Saturday.
- Resolutions or assistance requests will be completed by EOD Monday (testing should be performed upon resolution, but will also be verified in final phases)
- Tuesday is reserved as a buffer for unforeseen issues

Resource Management

At Patrick's suggestion, communication was primarily conducted through Microsoft Teams, allowing faster, more direct explanations and discussion, as well as direct file and image sharing whenever needed or convenient. Screen sharing and video conferencing were utilized on a number of occasions for demonstrations and troubleshooting.

Table of Contents

Requirements Specification

Operating Environment

The following lists document the system requirements for this application. Requirements are broken down by operating system. The minimum requirements for this application are based on the minimum system requirements for running Java 8 as listed on Oracle's website.

Windows:

- Windows 10 (8u51 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- RAM: 2GB
- Disk space: 512 MB
- Processor: Minimum Pentium 2 266 MHz processor

Mac OS X:

Intel-based Mac running Mac OS X 10.8.3+, 10.9+ Administrator privileges for installation RAM: 2GB

Linux:

- Red Hat Enterprise Linux 5.5+[1] 6.x (32-bit), 6.x (64-bit)[2]
- Red Hat Enterprise Linux 7.x (64-bit)[2] (8u20 and above)
- Suse Linux Enterprise Server 10 SP2+, 11.x
- Suse Linux Enterprise Server 12.x (64-bit)[2] (8u31 and above)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 14.x (8u25 and above)
- Ubuntu Linux 15.04 (8u45 and above)
- Ubuntu Linux 15.10 (8u65 and above)

Design and Implementation Constraints

The design implementation of this software package requires that developers only use Java libraries and APIs compatible with 64 bit operating systems listed in Operating Environment. Platform specific or low level API calls should be avoided to ensure the application functions properly on all major operating systems.

Assumptions and Dependencies

This project depends on having a JDK runtime with the JavaFX library installed on the user's machine. For production, the application will bundle a copy of the JDK and the required libraries in the application's installation folder.

Table of Contents

Nonfunctional Requirements

Performance Requirements

The file selected by the user must be of a valid file type (such txt/ docx/ etc). As the goal of this program is to search for matching strings, trying to upload an incompatible (such as JPG) is disallowed. Any file selected within ConText must first be closed (will display error message and end process if file is currently in use by another program).

Legal Requirements

Notice will appear upon opening of the program, alerting the user that this program should not be used for copyright infringement or any related activity.

Security Requirements

The user is required to have system access to be able to (a) read files in order to view documents, (b) write files in order to save documents. This application can be used by entering text manually, highlighting relevant search terms, and displaying the shortened report. For this use only, the user does not require system access privileges.

Table of Contents

System Specification

Product Functions

The major functions of this application involve highlighting and searching within a text document. The application is capable of the following types of text searches:

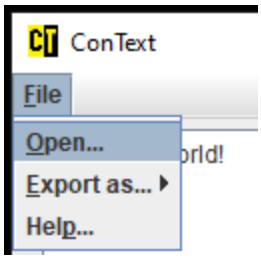
- Multiple words or strings can be searched and highlighted simultaneously
- Words can be highlighted in different colors
- Searches can be based on relative word placement
    - Search for multiple words that are within specified number of characters, or in the same sentence, paragraph, or section
- The same text can be highlighted in different colors based on capitalization
- Words can be highlighted by placement in sentences (i.e. firstword, last word, etc.)
- Text can be separated based on word locations (i.e. newline created before word instances)
- Results can be jumped to or scrolled through (“Find Next” button or browse manually)
- After highlighting, allows user to view document in full-screen while maintaining highlights

Table of Contents

User's Quick Start Guide

ConText is a document editing tool that allows the user to manually enter text or upload text or PDF files, then highlight words based on their position relative to each other. After the keywords are identified, it allows the user to abridge the document to only the sentences or paragraphs that include the specified word(s) and export the new document as a text or HTML file.

After opening the program and reviewing the Terms of Service, manually type text, copy and paste from another source, or open a PDF or TXT document.

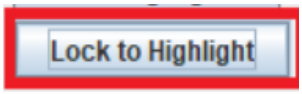


Open a Document

Click "File" in the top left corner, then choose "Open". A file chooser window will display where you can select the file you want to open. You can make changes to the text after you open the document.

Lock to Highlight

When you are finished entering text in the main text area, click



Search Widgets

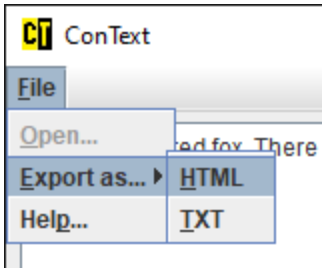
Type your search terms in the **Keyword Entry Box**.
Exact Match specifies that similar terms should not be matched. NOTE : ignores capitalization
Match Case specifies only words whose capitalization matches that of the entered keywords.
Search Type specifies whether keywords identified by this search should only be highlighted if they are in the same sentence or paragraph of another search. **Individual Search** highlights without regard to other searches.
Color button is to choose the color the identified keywords should be highlighted in.

Search Controls

Add Search adds an additional **Search Widget** to the set of searches. (max 8)
Highlight button executes the search and highlights identified words.
Remove all highlights from the document with the Clear **Highlights** button.
The **Sentences Only** button will remove all sentences that do not include a highlighted word.
The **Paragraphs Only** button will remove all paragraphs that do not include a highlighted word.
If you want to make changes to the document, you must click the **Unlock to Edit** button

---

Export / Save



To save, go to **File** > **Export as…** > **HTML** or **TXT**
To keep any highlights you must save as HTML
If you don't need the highlights, you can save it as a TXT file.
A dialog will be displayed, asking if you want to define the filename
Otherwise the program will generate a temporary filename for you
If you choose to name your file, a new dialog will appear to enter the new name

Your saved document will automatically open for you to view.
Note that it will be saved in the **tmp** subdirectory of the same directory in which **ConText** is stored.

Table of Contents

Test Plan and Results

Introduction

The current software to be tested is “ConText Document Highlighter” which accepts text files, then allows the user to search for multiple words or phrases simultaneously. The user also has the option to search for multiple words, based on their position to one another (such as being in the same sentence or paragraph).

Testing Strategy

Along with Macro + Micro Objectives

The objectives of this test plan are to determine if the software is able to successfully isolate and highlight the number of search parameters entered. This would begin by ensuring that the file types that are expected to be readable, are actually done so without error. In addition to the actual program code files, testing will require use of the IntelliJ IDE and a variety of text and PDF files as input source material. Test cost and resources are reasonably defined by access to electricity, a computer with requisite programs installed, and access to the internet. The primary focus of testing is to discover and correct as many bugs and unintended program operations as possible.

Each interaction test will include 2-5 interactions per item. Each test requiring documents will use 2-5 documents. Each test requiring string input will use 2-5 unique strings for each test. Strings and documents will vary in character set, length, and capitalization. Interaction tests will vary in assumed user familiarity and involve correct software use, as well as incorrect use. All testing will be performed by Patrick McCully, except in cases where initial testing failed and original producer of code performs follow-up testing for verification purposes.

Initial testing will focus on functionality of GUI components independent of user input and therefore will not require extensive use of test documents. GUI functionality will use varied combinations of interactions (clicking the same button or checkbox several times, entering excessive characters in a text box, etc) to determine whether some combination yields unexpected results.

Further testing will be performed as features are completed, requiring the use of test documents. Each performance test will be performed with a variety of input data, especially focusing on varied character types, relative location within text, and similar inputs (such as same word with different capitalization).

Upon weekly code completion, all members contributing code will alert the testing coordinator of any priority features that should be tested first (such as a feature that other features are dependent upon).

Table of Contents

Test Criteria for ConText

GUI Interaction

This test will be conducted on the GUI itself. Determine if GUI performs each task as intended.
Test whether a document can be opened and read by the GUI
Test whether GUI can display the document
Test whether the user can interacted with the document

GUI Performance

Determine whether the GUI can perform the logic requirements to find and highlight words individually or in sets. Determine whether the GUI can display the exported file with the user's specified settings.

Single Word Search

Determine whether the GUI can identify and highlight specific words. Especially focus on unusual character combinations.

Read Appropriate File Types

Determine if text of a doc, pdf, or txt file will be read and displayed within the GUI

Color Highlight - single word

Test that the specified word is highlighted with the user-selected or default color.

Proximity Word Search

Determine whether word sets are identified when the words appear no more than a specified number of words or paragraphs apart.

Search and Export

Use the "Search" button to determine whether the correct search is executed. Use the "Export" button to determine whether the appropriate file is displayed.

Document Saving

Test whether the application has properly saved the requested document, along with highlights and abbreviations.

Multi-word search

Test whether the application will properly search for more than one word simultaneously.

Table of Contents

Test Deliverables

Test Cases
Test Incident Reports
Test Summary Reports
Error Logs (collated in BUGS document)

Testing Environment

Hardware	Computer, Mouse, Keyboard, Monitor
Networking	Access to Google Drive and GitHub
Software	JDK : 8+

Operating System : Windows Vista or 7+; Mac OS X 10.8.3+; Red Hat Enterprise Linux 5.5+; Suse Linux Enterprise Server 10 SP2+, 11.x; Suse Linux Enterprise Server 12.x (64-bit)[2] (8u31+); Ubuntu Linux 12.04 LTS, 13.x; Ubuntu Linux 14.x (8u25+); Ubuntu Linux 15.04 (8u45+); Ubuntu Linux 15.10 (8u65+)
(Testing on Linux kernels is not critical, as VIM can perform similar operations and is likely preferable for Linux users.)

Control Procedures

Error Reporting

Bug reporting will use the form in section 14. Reassignment of bug handling will require name change to report title. Each member should check for bugs as often as possible, but at least weekly, or within three days of last code submission. Person reporting the bug report may additionally send an email notification to the person assigned. If a bug is preventing the execution of another part of the code, the person responsible for the other part of the code should be notified by email when the bug is resolved.

Change Requests

Major design changes are not anticipated, however, in the interest of planning for the unanticipated, any member that believes the design needs to be modified should email the rest of the group with the subject line: Change Request: (*suggested change*) and include individual tags to any other member that would be affected by the change. (Such individual tagging has already served well for more directed group communication.)

Features Not to Be Tested

Ability to open, read, and write to all file types. File types specified should open, but the user should be alerted that non-text files cannot be opened and the attempt to open the file should cease. Allowed file types include: txt, doc, docx, pdf (TBD, based on filetype conversion.)

Table of Contents

Error Log Samples

Using personal PDF test file as input

Bug: 1.1 : RESOLVED

Expected to highlight full sentence using "Same Sentence Search" for the sentence "All civilizations unified under into one race called "Omnixans"

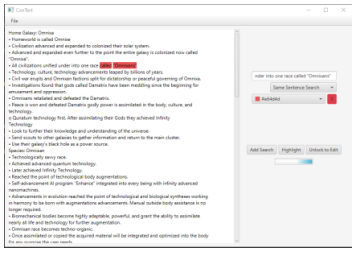
Parameters:

Text: All civilizations unified under into one race called "Omnixans" Type: Same Sentence Search

Expected Highlight: All civilizations unified under into one race called "Omnixans"

Actual Highlight: called "Omnixans"

Resolution: Error in Search logic; list of found words not being added to returned list



Bug: 1.2 : RESOLVED

Expected for the paragraph to be highlighted when executing the "Same Paragraph Search" but will only select 2 words.

Bug: 1.3 : RESOLVED

Expected using "Same Sentence Search" to highlight a sentence with 4 other searches but failed to locate for the fourth search.

Result:

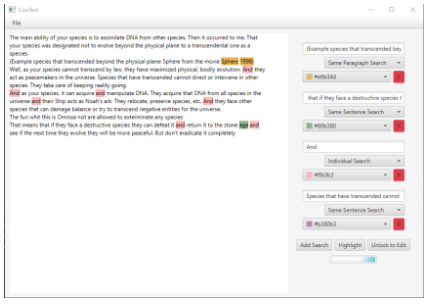
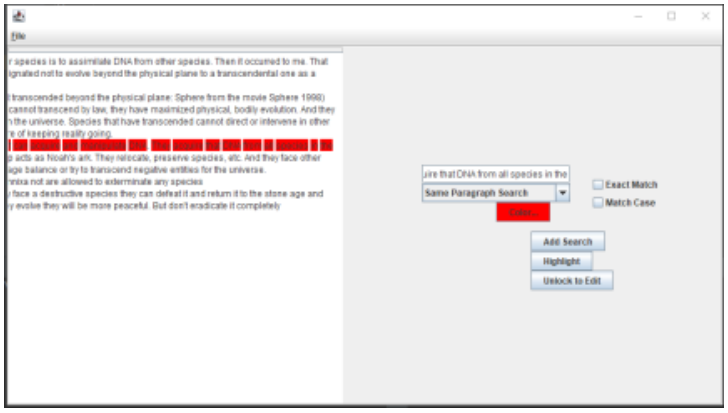


Table of Contents

Bug: 2 : RESOLVED

Unexpected resizing of window during use. Associated with search string longer than keyword entry box size

Result:



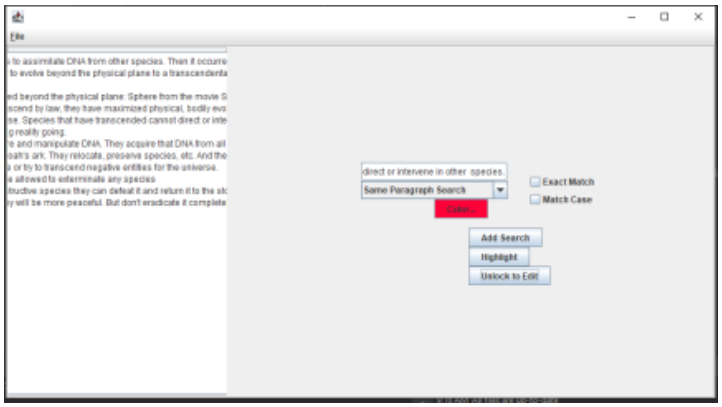


Table of Contents

Design and Alternate Design

Primary Design

Overview
The design uses three requisite classes: Main, Search, and Export. Main creates the GUI and controls the other classes. Search converts user input into regex text. Export converts the modified document into a portable format and allows the user to save. Additional classes may be added, depending on time constraints and performance results.

Main
The first class will consist of the main GUI code using the JavaFX library. The GUI will upload the contents of a text file into a large visual field, and because the purpose of our program includes altering the color and appearance of the text, some additional library support may be necessary. Based on prior experience using JavaFX (John Ingle), one excellent source for this would be the FXMisc Github, where a number of subprojects are suitable for our graphical needs. To implement the upload feature, an option will be added to the File menu of the GUI, allowing the user to navigate to the desired directory and selecting an appropriate file to use.

Search
The second class will house the searching algorithms and functions, and forms the backbone functionality of ConText. Regex will be used for said functionality, as it fits our needs perfectly. Regex was designed for searching and modifying text, and it has a near infinite flexibility in terms of how its expressions can be customized for each specific search parameter that may arise in our project's use. Java has this support already built-in, using Pattern and Matcher objects that would be custom-tailored for each search. Instances of Search will be called by the Main class upon the user's selection to search for additional keywords, and each instance will perform the actions required to convert the user's search into Regex, usable by methods in Main.

Export
The third class will implement the export feature, which allows a user to save the results of any given search as an HTML file with the highlight formatting preserved. This allows comparison and analysis of multiple documents against each other, in a more comfortable viewing interface (such as multiple tabs in a web browser), as well as the portability to share the results with others or across devices. This class will be fairly small, as it only requires
straightforward Java I/O and file system manipulation capabilities, after the data to be written is prepared.

Potential Adjustments

File Conversion

Table of Contents

One additional class that we may need to include is for file conversion of different text based formats into TXT. Unlike the Export class, this will need to be integrated into the upload functionality and become a process that is automatically performed at file selection if the file format is not detected as TXT. The only modification to the Main class will be a conditional to detect file type and instantiate this class if necessary.

Performance Considerations

Since the implementation of the Tokenizer class, we have had minimal performance issues compared to the use of RegEx with an increasing number of search boxes and parameters. Basically all of the previous suggestions we devised earlier have been implemented: by switching to tokenization of words rather than RegEx via creation of a separate class, and introducing a limit of maximum number of searches. In the current version, no performance-based issues have been noted thus far.

Alternate Design Possibilities

- JavaFX – Initial program design used JavaFX, before conversion to Swing. The main reason behind the change was due to not all members being able to build based on FX correctly, perhaps due to a specific system configuration. However, FX would still be a viable option to base the project on, given that it seems to have more customizability options in the look and feel of the GUI.
- RegEx – Also another initial feature of our Search class, RegEx was excellent up until the point where we began adding multiple search boxes, with each box allowing the user to enter in any number of words or phrases. This began causing significant performance issues, system lag, and was generally unsustainable. If the program was restricted to minimal design specifications, such as a single search box with a few search terms to compare, then likely RegEx would have no issues and be a viable alternative.

Test Cases

Test cases will aim to test the functionality of the program during use as intended, as well as to understand the behavior under misuse and identify unanticipated results. Test cases will include the basic file operations (opening a file, entering data manually, exporting the completed file, and program opening and closing processes), as well as function testing as described below. Beyond standard testing, we expect to identify additional areas where issues may arise as we get further into development and include testing of those areas in our final product assessment.

Functionality testing will include:

- Combinations of
    - Individual Search
    - Sentence Search
    - Paragraph Search
- Varying number of words entered into each search box.
- Different colors for each situation.

Sample testing below

Table of Contents

Sample test cases may be conducted with the following parameters:

Search Boxes	Contextual	Number of Words	Color
1	WD	1	C1
1	WS	1	C2
1	WP	1	C3
2	WD, WD	1, 1	C1 , C2
2	WD, WS	1, 1	C3 , C4
2	WD, WP	1, 1	C5 , C6
2	WS, WD	1, 1	C1 , C2
2	WS, WS	1, 1	C3 , C4
2	WS, WP	1, 1	C5 , C6
2	WP, WD	1, 1	C1 , C2
2	WP, WS	1, 1	C3 , C4
2	WP, WP	1, 1	C5 , C6
3	WD, WD, WD	1, 1, 1	C1 , C2 , C3
3	WD, WD, WS	1, 1, 1	C4 , C4 , C6
3	WD, WD, WP	1, 1, 1	C7 , C8 , C9
3	WD, WS, WD	1, 1, 1	C1 , C2 , C3
3	WD, WS, WS	1, 1, 1	C4 , C4 , C6
3	WD, WS, WP	1, 1, 1	C7 , C8 , C9
3	WD, WP, WD	1, 1, 1	C1 , C2 , C3
3	WD, WP, WS	1, 1, 1	C4 , C4 , C6
3	WD, WP, WP	1, 1, 1	C7 , C8 , C9
3	WS, WD, WD	1, 1, 1	C1 , C2 , C3
3	WS, WD, WS	1, 1, 1	C4 , C4 , C6
3	WS, WD, WP	1, 1, 1	C7 , C8 , C9
3	WS, WS, WD	1, 1, 1	C1 , C2 , C3
3	WS, WS, WS	1, 1, 1	C4 , C4 , C6
3	WS, WS, WP	1, 1, 1	C7 , C8 , C9
3	WS, WP, WD	1, 1, 1	C1 , C2 , C3
3	WS, WP, WS	1, 1, 1	C4 , C4 , C6
3	WS, WP, WP	1, 1, 1	C7 , C8 , C9
3	WP, WD, WD	1, 1, 1	C1 , C2 , C3
3	WP, WD, WS	1, 1, 1	C4 , C4 , C6
3	WP, WD, WP	1, 1, 1	C7 , C8 , C9
3	WP, WS, WD	1, 1, 1	C1 , C2 , C3
3	WP, WS, WS	1, 1, 1	C4 , C4 , C6
3	WP, WS, WP	1, 1, 1	C7 , C8 , C9
3	WP, WP, WD	1, 1, 1	C1 , C2 , C3
3	WP, WP, WS	1, 1, 1	C4 , C4 , C6
3	WP, WP, WP	1, 1, 1	C7 , C8 , C9

Note - Lighter colors for highlighting are usually preferable for reading, to allow readability and good contrast. Preference should be given to lighter color testing, but not to an extent as to discount darker color testing

Table of Contents

Development History

Version:

0.1	GUI framework with JavaFX created within Main class
0.2	Added Search class using RegEx pattern matching
0.3	Added highlight functionality in Main class
0.4	Added Export class, currently to HTML only
0.5	Completed conversion to Swing for GUI
0.6	Removed RegEx, created Tokenizer class to expand searching capacity
0.7	Incorporated maximum number of search boxes allowed at 8
0.8	Added Error class to allow tailored response and user input as needed
0.9	Refined elements of GUI, such as fixed width of Search widget's frame
1.0 (current)	Further improved GUI elements, this time for main text area

Table of Contents

Conclusions

There were many lessons to be taken from this project. One of the most important take-aways was the experience of working together on a team. Given the nature of our particular project, as well as the exchange of software amongst group members, it also increased exposure to the differences between Swing and JavaFX, the usefulness of GitHub, Google Drive, and Microsoft Teams for collaboration, and the nuances of different IDEs. Some members of our group were surprised at how much they enjoyed IntelliJ over their previously preferred IDE, some were introduced to unfamiliar concepts in algorithm design, and some learned that hype over a particular product may be exactly that.

The ConText program is designed to help users locate connections within a given document and the potential applications therein are limitless. Users from any field, any background, can benefit from what ConText offers. Currently our limitations are primarily based on importing and exporting file types, as we only support PDF and TXT as input and HTML and TXT as output. While many important documents and files will have this format (and a user also has the ability to copy and paste text from any other filetype), for future improvement it will be essential for us to incorporate conversion to and from a broader range of file types, especially those related to commonly-used text editors (such as DOC and DOCX).

While searching the web for information related to building this project, we came up with the idea of creating a browser extension, which could provide even more flexibility and benefit for users as it would permit extensive search capabilities within the text of web pages.

We also imagined additional features that could be added to the design of the original product. In some cases, assumptions are made regarding how to search. For instance, we assume that when a user selects “Exact Match” they want all words in that box to match exactly, but an improvement would be to add shortcuts to give the user control of which words should match exactly and which don’t have to. (Google search uses quotation marks to identify exact match word sets. We could incorporate the same.) We could also give the user control over what order the words should appear in Same Sentence or Same Paragraph search.

Overall, this project was an excellent introduction to collaboration, problem solving, and long term planning.

Table of Contents