|  |
| --- |
| Dylan Phelps |
| Web Based Text Analysis |
| A2 Computer Science Project |

|  |
| --- |
|  |

Table of Contents

[Analysis 3](#_Toc482198915)

[Background to the Problem 3](#_Toc482198916)

[Interview with Mr. Martin – The Primary Client 3](#_Toc482198917)

[Observation of the current system 4](#_Toc482198918)

[Existing System Flowchart 5](#_Toc482198919)

[Assessing the Current System 5](#_Toc482198920)

[Limitations and Hardware Requirements 6](#_Toc482198921)

[Objectives for the proposed system 7](#_Toc482198922)

[Potential Solutions 9](#_Toc482198923)

[A semi-manual system 9](#_Toc482198924)

[Manual Analysis with Automated Storage 9](#_Toc482198925)

[Desktop Application 10](#_Toc482198926)

[Justification of Chosen Solution 10](#_Toc482198927)

[Dataflow Diagrams 12](#_Toc482198928)

[Text Analysis Diagram 12](#_Toc482198929)

[Sign-Up/Login Diagram 13](#_Toc482198930)

[Entity Relationship Diagram 14](#_Toc482198931)

[Documented Design 15](#_Toc482198932)

[Explanation of the solution 15](#_Toc482198933)

[Overall System Design 15](#_Toc482198934)

[Pre-Built Modules 15](#_Toc482198935)

[Flask Microframework 15](#_Toc482198936)

[NLTK 16](#_Toc482198937)

[File Structure 16](#_Toc482198938)

[Class Design 17](#_Toc482198939)

[Class Diagram 17](#_Toc482198940)

[User Interfaces 18](#_Toc482198941)

[Data Structures 20](#_Toc482198942)

[Stack 20](#_Toc482198943)

[Graph 21](#_Toc482198944)

[Circular Queue 21](#_Toc482198945)

[Algorithms 22](#_Toc482198946)

[Example Flask Function 22](#_Toc482198947)

[Alliteration Checker 24](#_Toc482198948)

[Sentiment 25](#_Toc482198949)

[TF-IDF 26](#_Toc482198950)

[Keyword Graphing 28](#_Toc482198951)

[Radix Sort 29](#_Toc482198952)

[Database Structure 30](#_Toc482198953)

[SQL Notation 31](#_Toc482198954)

[Search by Category 32](#_Toc482198955)

[Add Text 32](#_Toc482198956)

[Find Text Owner 32](#_Toc482198957)

[Delete User 32](#_Toc482198958)

[Validation and Exception Handling 33](#_Toc482198959)

[Prototype 34](#_Toc482198960)

[Initial User Feedback 34](#_Toc482198961)

[Changes to the System 34](#_Toc482198962)

[Evaluation of changes 35](#_Toc482198963)

[Optimisation 35](#_Toc482198964)

[Test Plan 37](#_Toc482198965)

[Code Overview 42](#_Toc482198966)

[URL Structure 42](#_Toc482198967)

[Flask Control Function 44](#_Toc482198968)

[Algorithm Training 46](#_Toc482198969)

[Testing 47](#_Toc482198970)

[Test Corrections 68](#_Toc482198971)

[Evaluation 70](#_Toc482198972)

[User Feedback 72](#_Toc482198973)

[Final System Evaluation 73](#_Toc482198974)

[Code Appendix 74](#_Toc482198975)

# Analysis

## Background to the Problem

The Ulverston Victoria High School English Department is a large department that teaches a wide range of age groups from year 7 to upper sixth. The teachers often have to analyse new texts which they want to introduce to their classes.

The current system that the teachers use is to find texts online and then to analyse these texts by hand. Because this system is manual it takes a long time for the teachers to analyse these texts and this has to be done many times for each class over the course of the year. The system the teachers use currently to store the analysis is to type or copy the text into Microsoft Word and then highlight and comment on the text. As this is a general purpose software, it is inadequate for the tasks the teachers are trying to perform, the software does not provide much information about the text and teachers have to do all the analysis themselves. It is also hard to keep track of the texts the teachers have analysed and it is difficult to find and search through the texts, or get texts that are needed for a specific purpose. Therefore, a system is needed that will allow the teachers to easily input texts, analyse them, giving them a range of information about the text and make it easy to find and search for texts though a range of ways.

## Interview with Mr. Martin – The Primary Client

**What is the existing system for finding and analysing texts?**

MM: We each manually analyse and find texts from the internet. We can sometimes type them ourselves using Microsoft Word from books and texts we already have in the department.

**What are the benefits of using the current system?**

MM: It’s quite easy to find the texts online from forums and google searches, as well we all have a knowledge of texts from books and anthologies that we use often. Also the teachers can style the analysis to their own classes, only showing the analysis that that specific class will need to see

**What are the drawbacks of the current system?**

MM: We are limited by the amount of time the teachers can spend analysing the texts, as the department is very busy and the teachers have very little free time during school hours. Also many of the texts have the same basic features and it can be extremely tedious to go through several texts and look for the same repetitive features. There is also no way for the teachers to store the texts effectively, this means that finding texts can be quite hard after they have been analysed, this is compounded by the fact there is no way to search for texts, especially texts with common themes, such as “texts with alliteration”. In addition, teachers often analyse the same texts as each other without knowing, which is inefficient.

**Which new features would be most important to you?**

MM: We should be able to add our own text from text files, pdfs, websites and by typing the texts in to the system directly. Also, the ability to save the texts so that they can be categorized and easily found by the teachers if they are needed again, and have them searchable. We would also like the analysis that is made to be able to be toggled so only the analysis that is needed can be seen.

**Which existing features are the most useful?**

MM: The existing features that I would like to see in the new system is the ability for the teachers to add their own analysis so that they can personalize the analysis to the specific class they are teaching.

**Which language features are the most important to be analysed?**

MM: There is a variety of features that the teachers look for, the main ones are the language and the punctuation used. Other features could be similes, metaphors, lists, repetition, juxtaposition, alliteration and rhetorical questions.

**How many members of staff will be using the system?**

MM: Seven teachers in the department will be using the system.

**How would you like the analysis to be displayed?**

MM: A selection menu where the teacher could choose which feature to highlight would be ideal. In addition, information about the longest sentences and words, as well as graphs showing the most used phrases, words or punctuation would be great.

## Observation of the current system

I observed one of the teachers use the current system and they interact with it in order to identify the problems.

The current process:

1. A text is selected
   1. In one case the teacher searched the internet to find a text that was in the style to be analysed, they then copied this into a word processor.
   2. In another case, the teacher found a text in a book and typed this into the same word processor.
2. The text is analysed
   1. The features in the text can be analysed by highlighting the text in the word processor.
   2. The features can also be commented within the word processor; this is used most often as it can identify what the analysis is without having to work it out every time you want to access the text.
3. The text is saved
   1. The text is either saved on the teachers drive on the school network, or saved onto their own laptops hard drive.
   2. As you can see, there is very little organization on most teacher’s laptops and home drive, making finding and reusing texts hard.

## Existing System Flowchart



This flowchart has allowed me to model the current system as it stands. Because of this I have been able to evaluate the parts of the program which are inefficient.

## Assessing the Current System

From my observations of the current system, I have discovered several weaknesses that contribute to make the current system inadequate. The first weakness is the amount of time it takes to actually do the analysis; this is both evident in the amount of time it takes to do each analysis and the number of times that the teacher will have to perform the same kinds of analysis on different texts. This is the biggest weakness of the current system and will be the primary goal of the new system to solve.

The next weakness that has become most apparent in my observations is the lack of organization that is built in to the system, although this can be teacher dependent, as they can organize the files in whatever way they want. However, as this is not forced by the system, it is something that the new system will have to address. This also links to the problem with finding old analyses once they have been finished with, as there is no searching feature in the current system.

The final big weakness I see in the current system is the lack of sharing and cooperation between the teachers, this will mean that teachers may often analyse the same texts without knowing, which wastes valuable time. To combat this, I will add a feature where teachers can share files with each other.

Addressing all of the issues I believe will allow the new system to work well for the client and allow for a large improvement over the old system.

## Limitations and Hardware Requirements

Currently the system will only be used within the school’s English department, and so there will be seven teachers using the system. All of the teachers have had training courses through the school on IT, so they are all computer literate and use computer systems within their jobs daily. However, I would still like my system to be as easy and user friendly as possible, which means a good user interface is key, but the user’s ability will not put a limit on the complexity and thoroughness of the system.

Nevertheless, there are some restraints on the system from other sources:

* Hardware: There is limited hardware that can be used by the teachers, this is of small concern whilst the teachers are using their own laptops or a school desktop as these modern machines can run most tasks with ease. However, hardware constraints will be of higher importance if the system gets a version, which works on phones or tablets. I have a wide range of hardware on which to design and code the system and many devices to test the system on, including ones almost identical to the ones that the teachers will be using when the system is in use.
* My ability: The problem cannot be too complex for me to solve using the resources that I have. The solution will be created using mainly Python3, primarily because this is the main language that I have learnt.
* Software: The school system has very tight rules on what sorts of software can be installed on the network and individual pcs on the network. This means that the project will have to be just an executable file installed manually on each system or a web application, which will be easier to get on every system and will not be constrained by system rules.
* Time constraints: The system and the documentation need to be completed by May.

## Objectives for the proposed system

After studying and observing the existing system, I have learned a lot about the drawbacks and weaknesses of the existing system and have consequently been able to see where my system will be able to improve on the current system. I have therefore been able to find what I see to be a complete list of features that will be needed in the new system in order to make it fit for use.

1. The user will need to able to create an account within the system, so that they can save any texts that they upload and so that other users can find them. The user then needs to be able to log into this account. To do this the user will have to enter a username, email and password
   1. The username must be longer than seven characters.
   2. The email must be in a valid format for an email e.g. [example@text.com](mailto:example@text.com)
   3. The password must contain at least one uppercase letter, one lowercase letter and a number, as well as being eight or more characters long.
2. The system must be designed with a user-friendly user interface. To do this I will need to make sure that my interface is:
   1. Easily navigable – Which will be achieved by making sure that the elements that the user will be interacting with are labeled and that the number of pages is small as possible.
   2. Provides Information – The system must give the user all the information that is required, as well as making this information easily findable and shown in a usable way.
3. The user must be able to input texts from any of their most used sources, these include text files, word documents, pdf files, raw text and from a website.
   1. Files – The user must be able to browse any file in their file system and upload it on to the system, the text must then be checked to make sure the text is extracted from it so that it can be tokenized and analysed.
   2. Raw Text – The raw text must be taken from the input, and sanitized to make sure that no code can be entered or injection attack can take place.
   3. Website – The user must be able to enter a URL, from this the text must be scraped and fed into the system.
4. The analysis must include these language features:
   1. Alliteration – Words that start with the same sounds will be highlighted, this will ignore ‘stop words’ (words that add no meaning, such as ‘and’ or ‘that’)
   2. Antithesis – Consecutive words that have contrasting meanings will be highlighted
   3. Juxtaposition – Consecutive sentences that contrast each other will both be highlighted
5. The analysis should be shown on the text that the user has inputted and phrases or words that contain the language features should be highlighted. From here the user should be able to choose which analysis that they want to see at any time, which should change the highlighting of the text to the new analysis that the user has selected
6. When the analysis takes place, the system must also find out statistics about the text, these will be displayed with the user’s text:
   1. Reading age – Using the Flesch-Kincaid Score, the system must work out the reading age of the text
   2. Sentiment – Sentiment will be calculated using the sentiment of the words within the text and then averaged out.
7. Most used words and punctuation from the text will also be found and these will be displayed as graphs on the page, which displays the analysed text.
8. Another feature that is to be found is the keywords of the text; these will be displayed alongside the analysed text.
9. The user then must be able to save that text along with the analysis and the statistics onto the server. The user must before this give the text a unique title, if the title is not unique the user must be prompted to enter another.
10. From the text that displays to the user the analysed texts, the user must be able to share the text with another user, this should be done by showing the user a list of the valid users. The user is valid if they do not have a text with the name of the text that you are sharing with them.
11. From the user’s profile page, they should be able to see their texts, sorted into texts that have been entered by them and ones that have been shared with them by other users.
12. Also form their profile page the user should be able to search for a text; the results should be displayed in the same, with a section for shared texts and one for owned texts.
    1. Search by Title – The user should be able to search for a string, if the title contains that string the text should be shown, this should not be limited to being the exact title and should not be case sensitive.
    2. Search by Category – The user should be given a list of the categories of texts that are in the system, from which they can select a category and all texts that are owned by them or that they have access to should be shown.
    3. Search by Keyword – The user should be able to search for a keyword in any of the texts. This could be expanded to be a smart search, so that when the user inputs a keyword, similar keywords are also searched for. The results should be ranked in order of how related the keywords are.
13. The user should have the ability from within the text view to delete texts that they do not need to be stored on the server anymore. It should delete the text from the database, both for the owner and the people that access to it, delete the keywords and the delete the file that held the analysed text
14. The system should include a download button in the text view that allows the user to download a copy of the plain text in a pdf file.

## Potential Solutions

### A semi-manual system

A semi manual system will allow the teachers to keep their current workflow. In this proposed solution, I would build an application that allows the teachers to select a file from their home drive and insert it into the application. The software would then analyse the text and show the user the analysis that have been completed, allowing the user to toggle the different results. The teacher could add these analyses on to their own copy before closing the application.

* This project is very feasible in terms of hardware and the software that I have available and it could be achieved within the period. However, the solution is unfeasible in terms of complexity and development.
* The main reason for this is that it only makes one improvement on the existing system; even though this is the most important improvement, it is still not good enough to make the new system feasible for use.
* All the storage and searching for the system would still have to be maintained by the teachers, which would again lead to bad organization and a system that I hard to search in.
* This system would cut out a lot of the time consuming repetitive tasks done by the teachers in the analysis; however, it still keeps the time-consuming task of having to copy and paste the texts into the system, which takes up a lot of time.
* Overall, although this system would save some time for the teacher, it still does not address all of the current systems weaknesses and does not meet all of the objectives, which I have proposed for the new system.

### Manual Analysis with Automated Storage

This system would be comprised of a system that would allow the user to enter the directory or URL of the text they would like to analyse. A word processor like window would then open up and allow the user to enter the analysis that they would like to enter. They could then store this based on a category and a title. This would later allow the user to view previously saved texts and would give the user to search theses texts.

* This solution is very close to the existing system in the fact that the teacher still does all of the analysis manually, the only part that will be computerized is the receiving and storage of the texts from the different sources.
* It would be feasible both in terms of development and my abilities, as well as in terms of the hardware and software available, but it would merely be an improvement as opposed to a solution.
* This solution does improve on the previous proposed solution in a few ways, including a higher level of automation in the storing and searching of texts, which is one of the big weaknesses in the current system.
* Nevertheless, this system does not have the automatic analysis and therefore is not a feasible system to implement.

### Desktop Application

The desktop application will be a fully functioning application that takes the directory or URL, automatically analyses the text, then displays the text to allow the user to toggle on and off the different text features that have been analysed. The texts will then be saved and will be available to access by the user, and will be able to be searched in order to find the all the features and texts that the user wants to find.

* This solution is a complete solution that will allow the majority of the objectives for the new system to be met as well as meeting or partially solving the weaknesses that I have discovered in the current system
* One of the drawbacks of using this system is that the files would only be able to be accessed locally; this then recreates the problem of texts not being easily accessible and text analysis needing to be redone. It would also be hard to allow teachers to share texts with each other without use of a file server and some form of application to control the files and the server, which would add to the time pressure of the project.
* The other drawback to this solution is that the application would have to be installed on every computer the teacher wanted to use the application on, adding to the work needed.
* In summary although this solution will meet the required specification of the project objective almost fully, it still has several large drawbacks, which do not make this solution optimal to implement

## Justification of Chosen Solution

The solution that I have chosen will potentially meet all of the objectives and remove all time-consuming manual work and processes from the system. After discussing the previous solutions with the user, we decided that none of these solutions could fully meet the user objectives or would use of the system easy. Therefore, we decided that an application, which has all the features of the last proposed solution, would be created, however instead of being a desktop application, it would be a web system. The user will enter texts on a website, these will then be sent to the server where the text will be processed and the analysed version stored. The user will also be able to register and log in to the system, where they will be able to store the texts so that they can retrieve and search them later.

* As long as all the objectives prove to be realistic and achievable, this is the most feasible potential solution. It meets the user’s needs as outlined by the objectives for the proposed system and it could be updated as time progresses to deal with changing text types and qualifications.
* The staff should not have a problem using the new system as they have experience using web systems and applications every day, so they will be used to using and interacting with the web based interface.
* The solution should be achievable given my skill level and the amount of time that I have to complete the project.
* As the system is online all the files will be stored on a server, this will allow the teachers to access their files from any computer. This means that no applications or software will have to be installed on any of the teacher’s devices.
* The web-based system also gives me the chance to create a responsive site that will allow the teacher to use the system on any device.

## Dataflow Diagrams

The key for the diagrams is:

* Oval – Input
* Rectangle – Function
* Document – File
* Cylinder – Database

### Text Analysis Diagram

This diagram shows the flow of data from when the user inserts the text, or link to the text, into the system, to when the text is displayed to the user and the text is saved to a file and to the database.

It shows how the text it categorized to be a file, a website or raw text where it will be extracted accordingly. The reader is then initialized, which is where the tokenized texts will be accessed from, and passed through to the analysis class. The analysis is then performed and a copy of the class is stored in a text file, the information about the text will be stored in the database and the text will be displayed from this class.

### Sign-Up/Login Diagram



This diagram shows the flow of data when the user wants to log in or sign-up. The top section of the diagram shows the process when the user wants to sign up. The details the user have entered are passed through to the validation function, which validates the details, checking them against current ones in the database. If these details are correct, the new user’s details are added to the database.

The second section of the diagram shows the login process. The details are checked against those in the database to make sure they have an account and that they have entered the correct password. If the details are correct, the user is logged in and the userID is passed to the Text and User\_Text table to find all of the texts that the user has access to. Once these texts are found they are passed to the show profile function where they will be rendered as links on the user’s screen.

## Entity Relationship Diagram



This diagram shows how the relational tables within the database used within my system will be linked together, through the use of foreign keys. The user table will store all of the information about the users. The user table is connected to the texts table, which stores all the information about the texts, by a line, which shows that one user can own many texts. This is different from the line to the User\_Text table which shows that each user can have access to many texts and as shown by the line from the Texts table, each text can have many users that have access to it. Each user can also have many notifications, which are stored in the Notifications table. The categories table stores all the different categories that the Texts can have, each category can have many texts. Finally, each text can have many keywords, but similarly each keyword can have many texts, this mean that a junction table is needed to hold this relationship, the table that I am using for this is word\_text, which within shows the which texts have which keywords.

# Documented Design

## Explanation of the solution

The solution that I have proposed will be a web-based Text Analysing and file organization system. The users will be able to submit texts from Text Files (.txt), PDF Files (.pdf), Word Files (.docx), the internet, by inputting a URL, and be able to type in their own raw text and submit it. The text will then be analysed showing these language features: Sentiment, Reading Age, Alliteration, Juxtaposition and Antithesis, as well as showing the most used words and punctuation in the text. The texts will then be stored on the system allowing for easy access. From here the text will be able to be searched by any of the features as well as by category, title and any keywords that the text contains. The user will also be able to share texts with any other user and download any texts they have saved on the system.

The back-end of the application will be written in Python3, using the Flask micro-framework. The design of the interface will use HTML, CSS, JavaScript and Jinja2. All the CSS stylings and any required JavaScript will be sourced from Bootstrap3.

## Overall System Design

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| Files  Raw Text  Web Addresses  Search Data  Login Data | Text Analysis  Text Searching  Text Extraction  Text Sharing  Text Download  Web Site Navigation | User Data  Texts and Text Data  Trained Document Frequency  Keyword graph | Analysed Texts  Search Results  Text PDF |

The main system features that I will have to design are:

* Web Page User Interfaces
* Data Structures
* Text Analysing Algorithms
* Database Structure and Interface

## Pre-Built Modules

### Flask Microframework

I will be using the Flask microframework to develop the system. Flask is a framework, which allows you to run the server, performing tasks such as serving pages and handling requests, within Python. Flask is not a MVC (Model-View-Control) framework as flask gives you freedom to choose how you implement the system; however, MVC is a good model to follow when designing a system anyway. Model is how the data is represented in the system, I will do this with a database, using SQLite, designing my own interface to the database rather than using a pre-built module, such as SQLAlchemy, I am doing this as I think it will give me more freedom as to how I structure my database and queries. View is the user interface of the system; I will be using Flask alongside Jinja2 to serve the pages from the templates designed in HTML. Flask will be primarily used in the Control part of the system, as I will be using it to handle form data and browser events, such as clicking on links or button, as well as using it to route the pages to the desired URLs.

### NLTK

NLTK is a module that is designed to perform a variety of natural language processing tasks. Within my program, I will be using NTLK, to process the text before it is analysed. The main way this will be done is through tokenizing of the texts. I will tokenize the text into words and sentences, allowing for easier analysis. Another way that I will be using NLTK is to remove the stop words from my text, these are words that do not add meaning to the sentences and so in some cases do not need to be analysed. The final way that I will be using NLTK is to have the huge corpus that it contains. These will be useful in testing my system as well as in training some of the algorithms that I will be implementing.

## File Structure

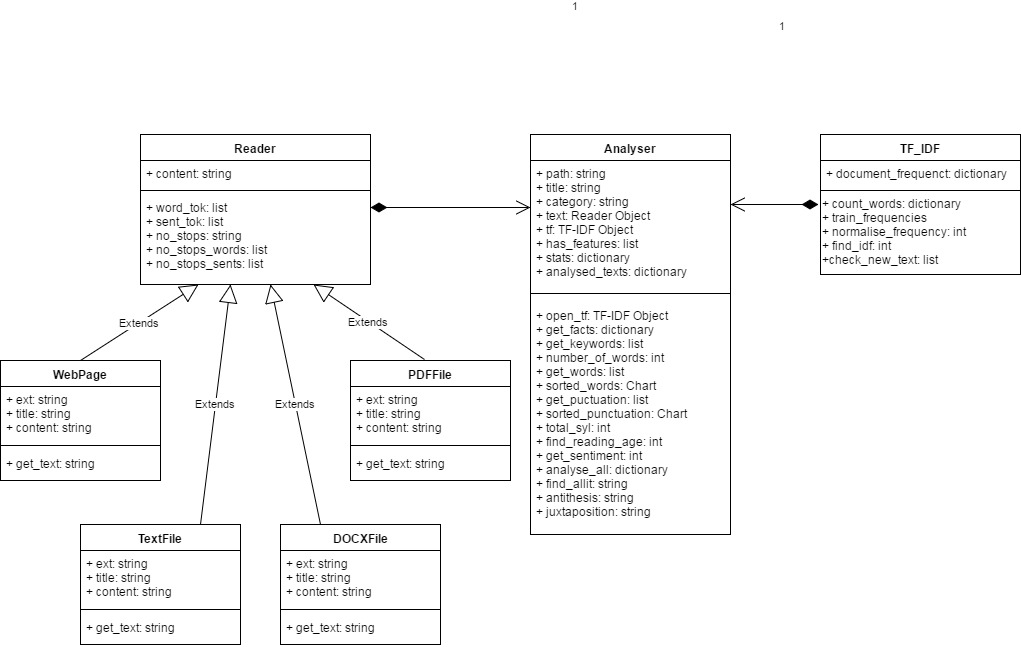
One of the challenges of this system is the large volume of files that the system will need to run, the files therefore will require organization, below is a diagram of the proposed file structure that I will implement in my system. Within each folder, I show an example of some of the files that will be contained there. The arrows from the top ‘\_\_init\_\_.py’ file shows how this will be the main file of the program that runs and which calls all of the other files, including templates and scripts.



## Class Design

Whilst designing my code I wrote class definitions for each class that I decided to use. In order to make my code is in the python format, I used docstrings to annotate these classes from the definitions that I originally wrote and these can be found below the header for each class in my code listing, see appendix A. Each of the docstrings provides a brief overview of the class, the properties that it has and the methods it uses.

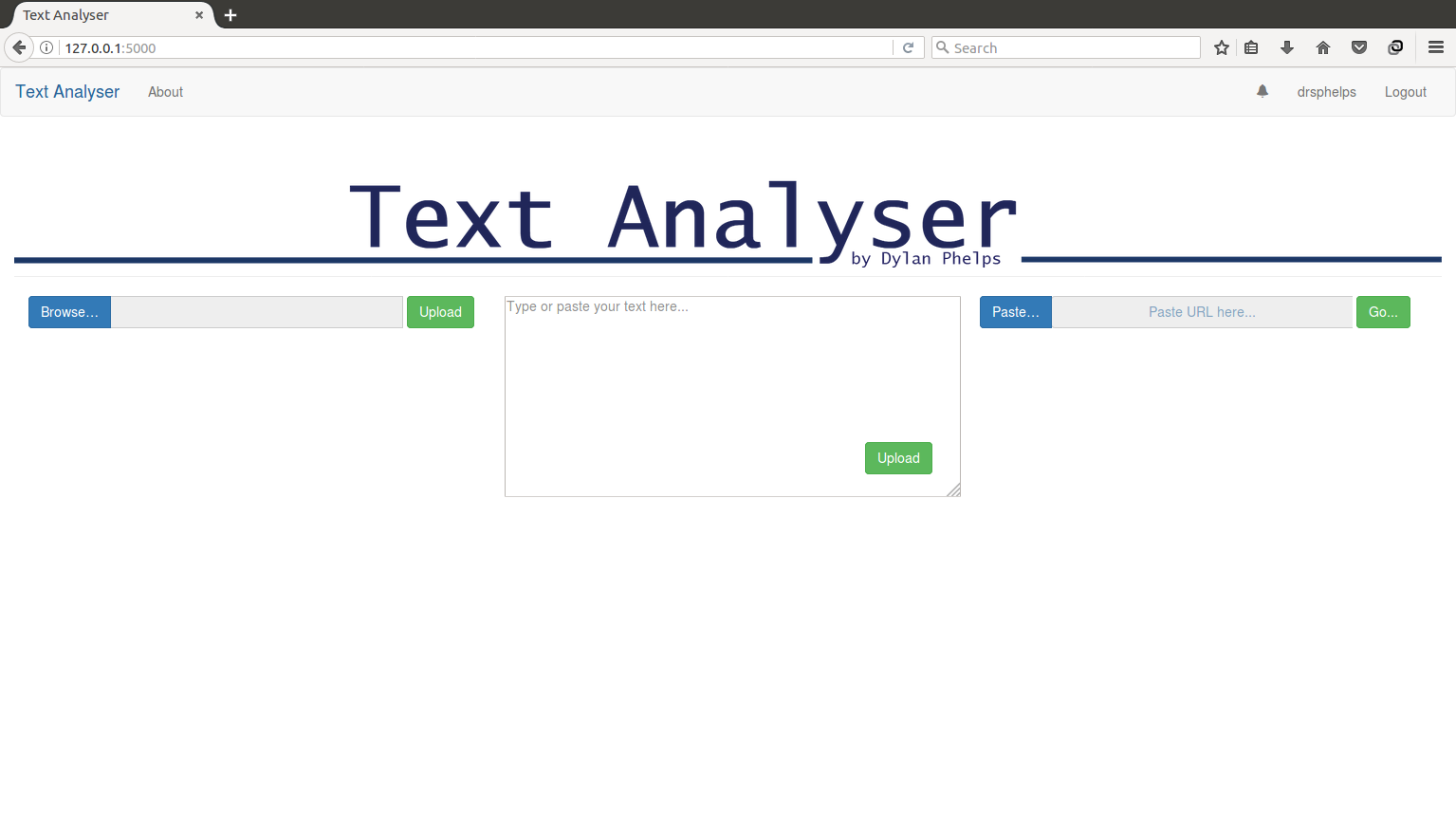
## Class Diagram



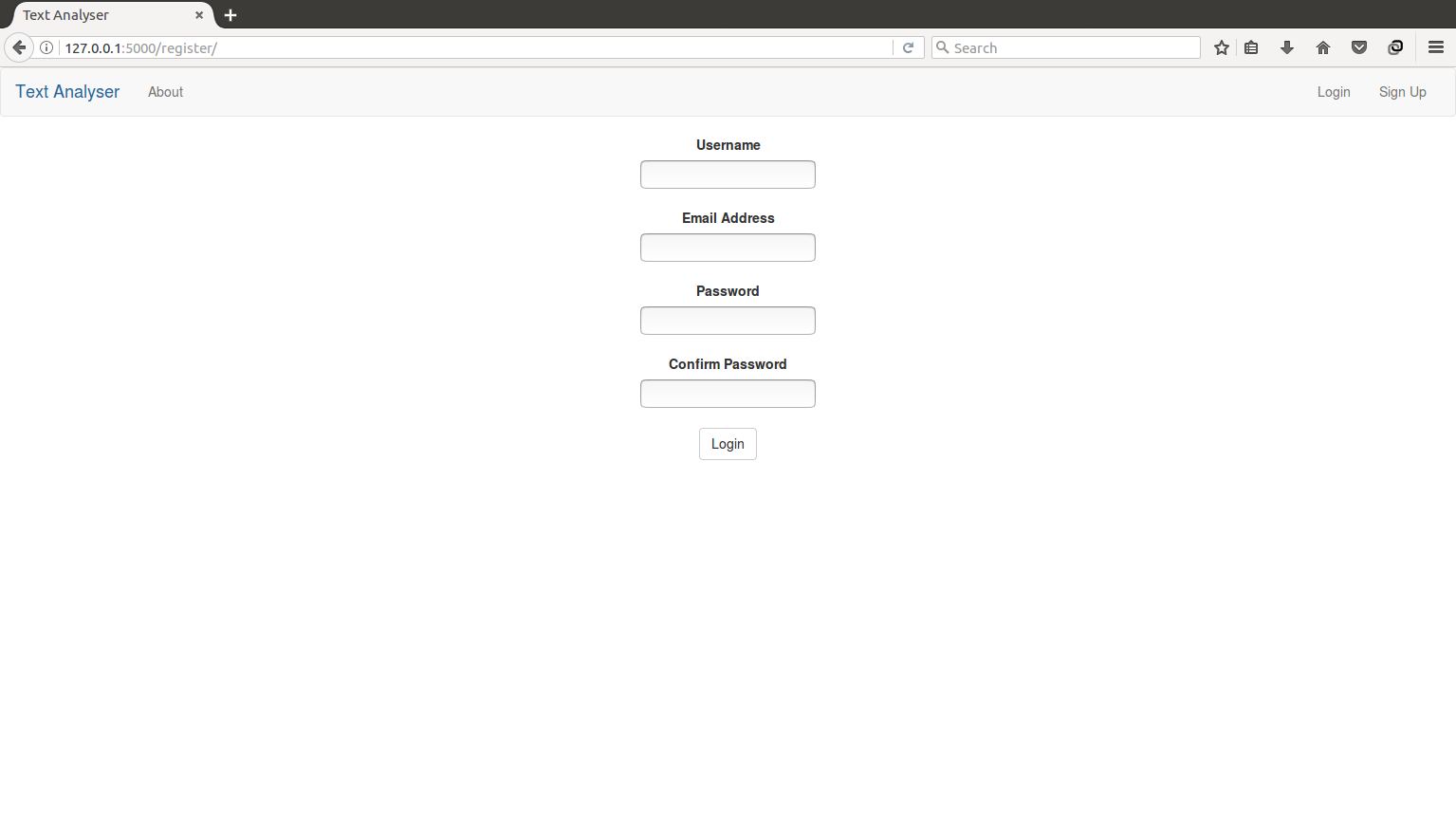
This diagram shows an outline of how the main classes within my program will be linked together, including the aggregation and inheritance that will take place. This diagram only includes the main classes that are used in the actual analysis of the texts. There are other classes used within the program, but as these do not have any aggregation or inheritance that needs to be explained so will be designed in the docstrings as specified in the previous section.

## C:\Users\Dylan\OneDrive - UVHS\Screenshots\LoggedOut.pngUser Interfaces

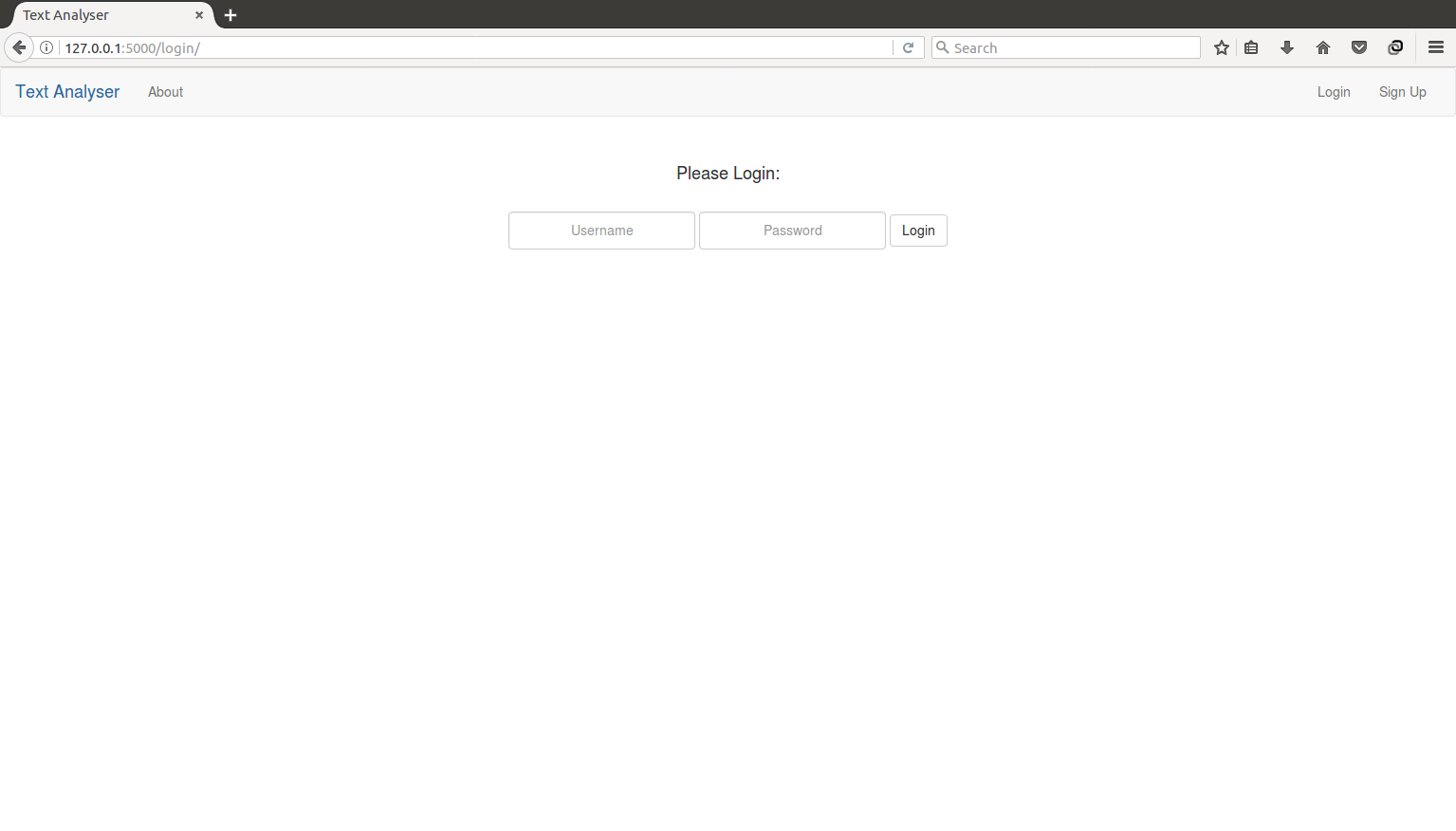
This is the first webpage the user will see when entering the site, from the header the user can see the about modal, get taken to the login page, or get taken to the sign-up page.



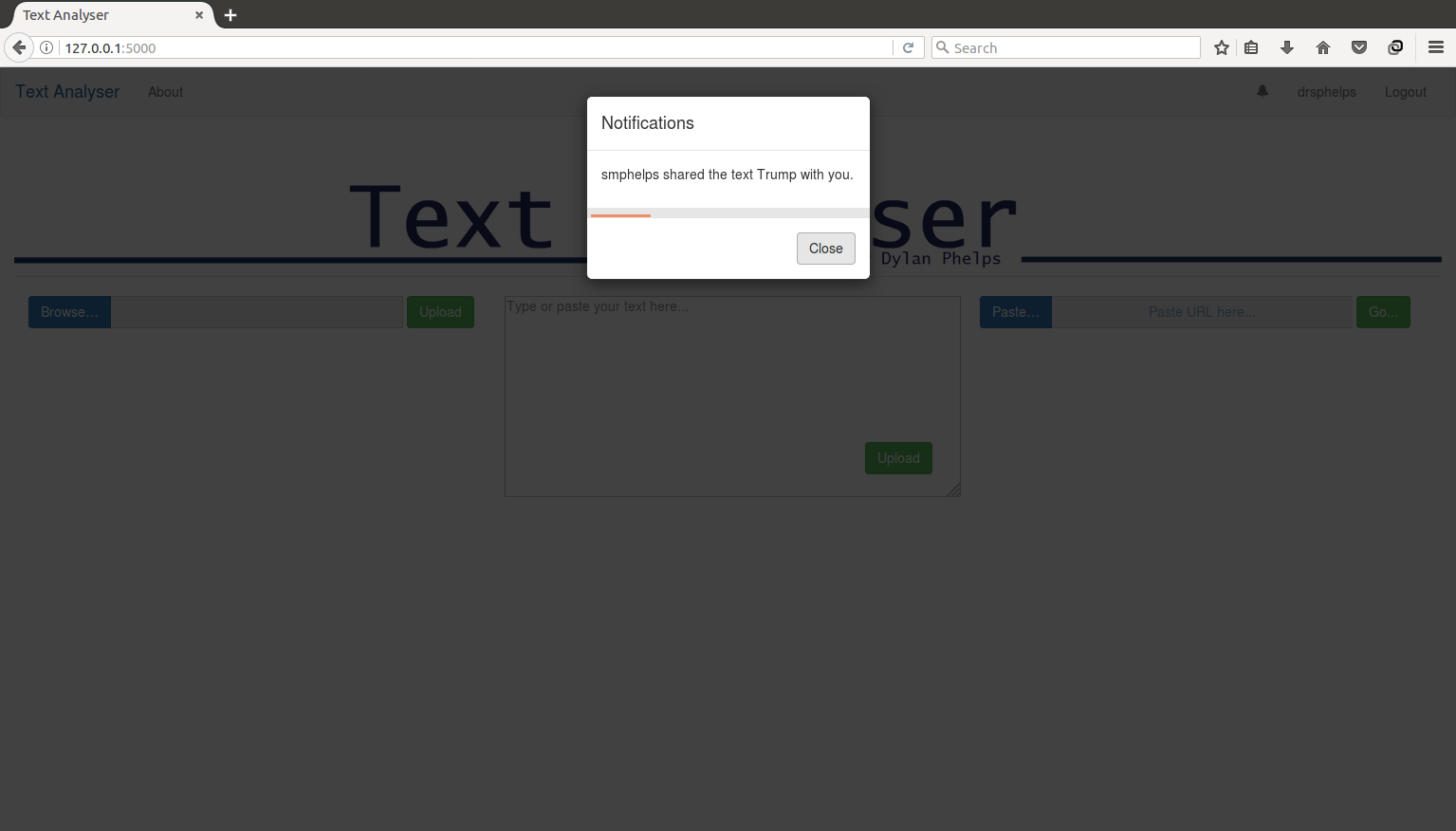
Once the user has logged in the header links change to the link for the user’s profile and the link to sign out, as well as the toggle for the notification modal.

The user can also enter their texts from one of the 3 text boxes. The left most box allows the user to browse their file system and upload a file. The center box allows the user to enter raw text and the right box allows the user to enter a URL.

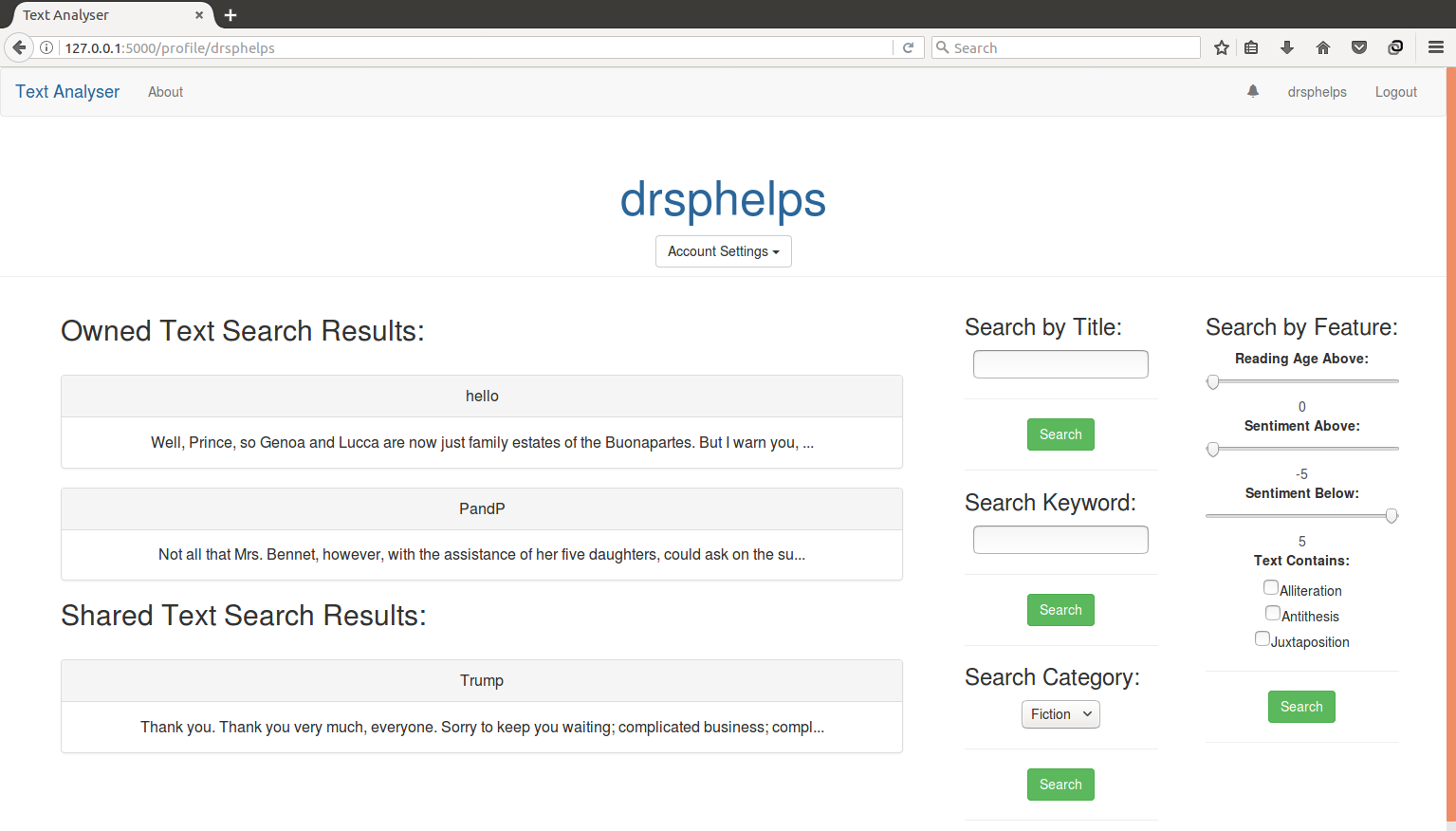
This interface allows the user to enter the details that will be stored about them on the system. It shows boxes for the username, email address, password and confirm password



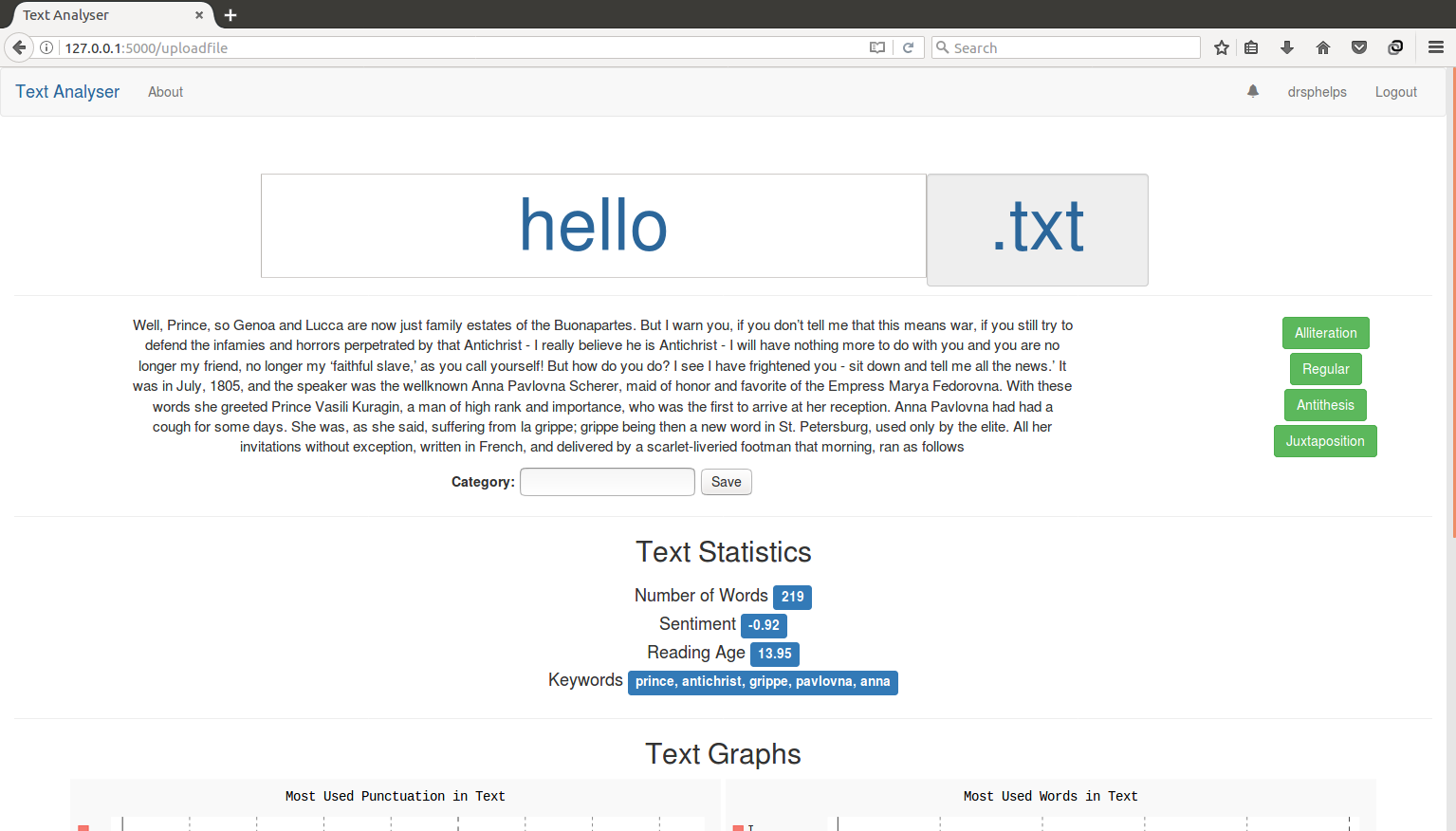
The login page shows just two boxes for the user to enter their username and password.



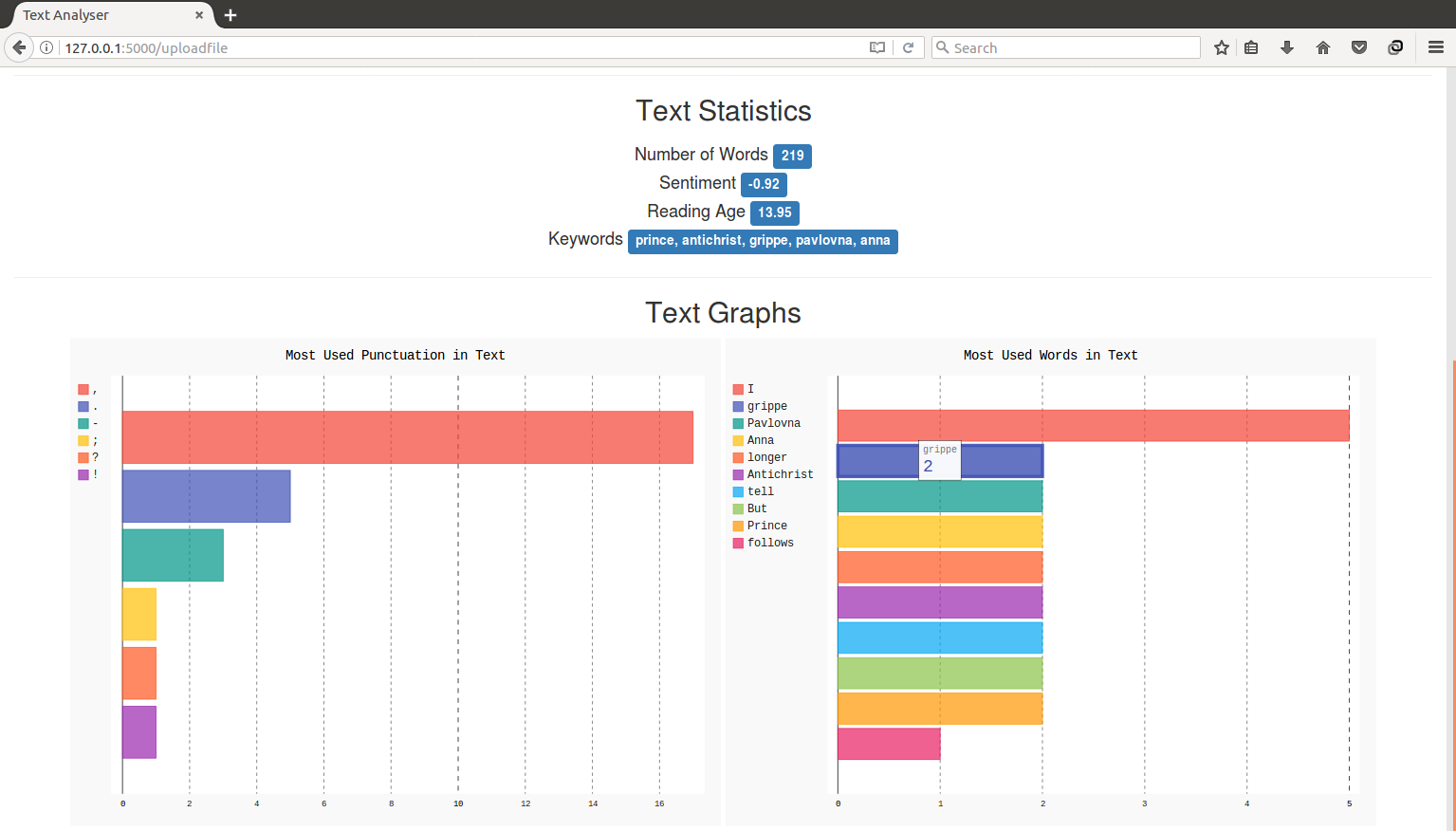
Once the user has logged in they will have access, from the header, to the notification modal, which displays messages such as “User has shared the text ‘Text’ with you”.



The user profile screen shows all the texts that the user has access to, these are organized into two categories, owned texts (that the user has entered) and shared texts (ones that have been shared with the user), each of the texts is represented by a box containing its name and the first 100 characters of the text, these can be clicked on to take the user to the viewing page for that text. From this page, the user can also search for texts. They can do this by entering a title or keyword in to the respective labeled boxes, choose a category from the dropdown menu, or enter which features they would like to be in the text and search for texts with those features.



This is the first half of the Text Display page, the page that the user will see when they input a text or when they choose to view it from their profile. This page contains the text title and the extension, as well as the text itself. Below this, there is the option to input a category to save the text into and the save button itself. Then we have the text statistics, this is all the information about the text such as sentiment, reading age and keywords. Finally, on the right hand side of the page are the buttons to choose which analysis to view, these can be selected and the specific analysis will be shown highlighted.



Below these elements we find the text graphs, these are graphs which display the most used words and punctuation. These graphs were created using the python library PyGal, this means that these graphs are interactive and bars can be toggled on and off from the key at the side.

## Data Structures

### Stack

I will be implementing a stack to perform operations on the words in the text, the stack will allow me to push appropriate words on to the stack and then retrieve them in reverse order, so that I can put the correct markings on them.

### Graph

I will be implementing a graph to be able to keep track of the keywords used in the texts and the edges and weights that link them together. The graph implemented is designed to be specific to the system that I am designing, specifically, the reduce edge feature is not a typical in a graph structure. I will be using this when I delete texts as it allows the weightings to be change, without the edge be deleted, as it will be used by another text.

|  |
| --- |
| Pseudo-code |
| Class Graph  CONSTRUCTOR  PROPERTY nodes 🡨 []  PROPERTY edges 🡨 {}  PROPERTY distances 🡨 {}  END CONSTRUCTOR  METHOD add\_node(value)  nodes.append(value)  END METHOD    METHOD add\_edge(from\_node, to\_node)  IF to\_node IN edge[from\_node]  distances[(from\_node, to\_node)] ++  ELSE  edges[from\_node].append(to\_node)  distances[(from\_node, to\_node)] 🡨 1  ENDIF  END METHOD  METHOD reduce\_edge(from\_node, to\_node)  IF distances[(from\_node, to\_node)] > 1  distances[(from\_node, to\_node)] --  ELSE  edges[from\_node].remove(to\_node)  IF LEN(edges[from\_node]) = 0  edges.REMOVE(from\_node)  ENDIF  distances.REMOVE((from\_node, to\_node))  ENDIF  END METHOD  END CLASS |

### Circular Queue

I will be implementing a circular queue to track the words while looking for differences in sentiment. With the circular queue, I will be able to choose a set number of words, and then use it to analyse the words at the ends of this set of a number of words. In my program, I will only be using it with queue size 2, but using a circular queue allows for the option in the future to look for differences in sentiments of words further apart. This could be useful, as it would allow for a greater depth in the analysis as we can look at the sentiment throughout a whole phrase and not just consecutive words.

|  |
| --- |
| Pseudocode |
| CLASS CircularQueue  CONSTRUCTOR(size)  PROPERTY max\_size 🡨 size  PROPERTY current\_size 🡨 0  PROPERTY queue 🡨 [None] \* max\_size  PROPERTY front\_pointer 🡨 0  PROPERTY rear\_pointer 🡨 -1  END CONSTRUCTOR    METHOD is\_empty  RETURN current\_size = 0  END METHOD    METHOD is\_full  RETURN current\_size = max\_size  END METHOD  METHOD peek\_at\_end  IF is\_empty()  RETURN False  ENDIF  RETURN queue[rear\_pointer]  END METHOD  METHOD dequeue  IF is\_empty()  RETURN False  ENDIF  item 🡨 queue[front\_pointer]  front\_pointer = (front\_pointer + 1) % max\_size  current\_size = current\_size – 1  RETURN item  END METHOD  METHOD enqueue(newItem)  IF is\_full()  RETURN False  ENDIF  rear\_pointer = (rear\_pointer + 1) % max\_size  queue[rear\_pointer] = newItem  current\_size += 1 |

## Algorithms

### Example Flask Function

1. @app.route('/register/', methods=["GET", "POST"])
2. **def** register\_page():
3. **try**:
4. **if** request.method == "POST":
5. form = reg\_form(request.form)
6. validation = form.validate()
7. **if** validation == "Passed":
8. password\_hash = pbkdf2\_sha256.encrypt(form.password.data, rounds=200000, salt\_size=16)
9. data = [form.username.data,
10. form.email.data, password\_hash]
11. with Database() as database:
12. database.addUser(data)
14. session['logged\_in'] = True
15. session['username'] = form.username.data
17. session['id'] = database.getID(session['username'])
19. app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \
20. session['username']
22. os.mkdir(app.config['UPLOAD\_FOLDER'])
23. filedir = app.config['UPLOAD\_FOLDER'] + '/files'
24. objdir = app.config['UPLOAD\_FOLDER'] + '/objects'
25. os.mkdir(filedir)
26. os.mkdir(objdir)
28. flash("You have been logged in")
29. **return** redirect(url\_for('index'))
30. **else**:
31. flash(validation)
32. **return** render\_template('register.html')
34. **return** render\_template('register.html')
36. **except** Exception as e:
37. flash(e)
38. **return** render\_template('register.html')

This is an example of a function written using the Flask framework, which combines the main features of the program. This function is what will run when the clients makes a request to the server along the path “/register/”, this could be by clicking a link on the web page or by typing in the URL. Line 1 contains the route within the website, “/register/” and the HTTP methods that can be user to access this webpage, “GET” and “POST”. Exception handling is used on line 3, and is within every flask function, this is to make sure that if anything goes wrong within the function the website does not crash, but simply serves the template to be rendered; the register page. If the pages is accessed using a “GET” request, the function will return the template for the page, allowing the user to see the page. However if the page is accessed using the “POST” request, this will usually (caught by the exception handling if not) be because the user has submitted the form on the webpage. When this happens, we pass the request form into the form reg\_form class. Within this, class the parts of the form: email, username, password and confirmed password, are retrieved from the form and stored within properties. Then on line 6 we validate the inputs that the user has given to make sure the user can sign up with these credentials. If the validation is passed we can has the password and add the data to the database. Line 14 to 18 change the data within the session variable. The session variable is a variable, which stores information about each concurrent user, similar to cookies in that they are stored on the client side, however they have an extra layer of cryptographic security, making them more secure than a normal cookie. Within the session variable, we store whether that client is logged in as well as the login username and user id. Config (line 19) is another variable created when the flask app is created. It stores information that is to be the same no matter where it is used within the app. A directory can then be created at this location on the server as well as object and files child directories, meaning separate upload folders are created for each user, allowing them to have a place for their own files. Line 28 shows off another feature of Flask, flash allows a message to be queued on the server so that when the next webpage is loaded the message is shown on the screen in an alert message box. Finally the Flask app redirects the user to the index page, redirect means the URL for that webpage specified will be loaded, not rendering a template at the original URL.

### Alliteration Checker

This algorithm is designed to check whether a text has alliteration in, and where the alliteration is. To do this I will be using a stack to hold the words as they are being checked. I have chosen to use the stack data structure as it allows me to check if the word is alliterative with the word already on the stack, using the peek function and add the word to the top of the stack. Then once there is no more alliteration, I can pop the items off the stack and know in what order the words are so that I can put the markers for alliteration in the right place. The actual checking for alliteration fill done using two functions, one to manage the stack and the formatting of the text and one to check whether the two words are alliterative. This algorithm has time complexity of O(n), this is calculated as the main find\_allit algorithm has one for loop which iterates through the n size array of words, within this loop only basic operations are performed, as well as a while loop, however this loop has a max size of 5 as this is the maximum size of the stack that it empties, leaving the complexity as O(5n) however as coefficients are ignored the overall complexity is O(n).

|  |
| --- |
| Pseudo-code |
| special\_case 🡨 [‘s’, ‘c’, ‘t’, ‘p’]  FUNCTION check\_alliteration(current\_word, top\_item)  IF current\_word first letter = top\_item first letter  IF the first letter is a special case  IF both words second letter is ‘h’  RETURN True  ELIF both words second letter is not ‘h’  RETURN True  ELSE  RETURN False  ENDIF  ELSE  RETURN True  ENDIF  ELSE  RETURN False  ENDIF  END FUNCTION  FUNCTION find\_allit  stack 🡨 stack()  FOR word IN text  IF word IS NOT punctuation OR NOT IN stopwords  current\_word 🡨 new\_node(word, index of word in text)  top\_item 🡨 stack.peek()  IF top\_item = False  stack.push(current\_word)  ELIF check\_allit(current\_word, top\_item)  stack.push(current\_word)  ELSE  last\_item 🡨 stack.pop()  WHILE stack is not empty  first\_item 🡨 stack.pop()  ENDWHILE  ADD mark at text[first\_item index]  ADD end mark at text[last\_item index]  ENDIF  ENDIF  ENDFOR  RETURN text  END FUNCTION |

### Sentiment

Three of the algorithms within my system will be based around sentiment. Sentiment is the score of how positive or negative a word is, based on a scale of -5 to 5. To find the sentiment scores of the words I will be using the AFINN-111 ranking, which is a collection of 2479 words whose sentiment have been hand scored. The three algorithms that I will be using with sentiment are ones to find: Antithesis, Juxtaposition and Overall Sentiment Score of the Text. The Overall Sentiment Algorithm will divide the total sentiment of all the words in the text by the number of words that have a sentiment score. Antithesis will see whether words have a difference in sentiment of above five between them. And Juxtaposition will look for difference of above 5 in the differences of the average sentiment of 2 consecutive sentences. All three of these algorithms have a time complexity of O(n) as they all only loop through each word given to them in the data.

|  |
| --- |
| Pseudo-code |
| FUNCTION total\_sentiment(text)  afinnwords 🡨 loads in dictionary of words and scores  sentiment 🡨 0  number\_of\_words 🡨 0  words 🡨 gets list of tokenized words in text  FOR word IN words  IF word IN afinnwords  sentiment 🡨 sentiment + afinnwords[word]  number\_of\_words ++  ENDIF  ENDFOR  IF number\_of\_words != 0  sentiment = sentiment / number\_of\_words  ENDIF  RETURN sentiment  END FUNCTION  FUNCTION antithesis  afinnwords 🡨 loads in dictionary of words and scores  words 🡨 list of tokenized words  comparison\_queue 🡨 sentiment\_queue()  FOR word AND index in words  IF word IS NOT punctuation OR stop\_word  IF word in afinnwords  extra\_info 🡨 {‘sentiment’: afinnwords[word]}  ELSE  extra\_info 🡨 {‘sentiment’: 0}  ENDIF  new\_word = Node(word, index, extra\_info)  comparison\_queue.enqueue(new\_word)  mark 🡨 comparison\_queue.compare\_sentiments()  IF mark = True  mark words in the words array  ENDIF  ENDIF  ENDFOR  result 🡨 join words array into string  RETURN result  END FUNCTION  FUNCTION juxtaposition  afinnwords 🡨 loads in dictionary of words and scores  sentences 🡨 tokenized sentences  sentiments 🡨 []  FOR sentence IN sentiments  sentiments APPEND total\_sentiment(sentence)  ENDFOR  FOR I 🡨 0 to I = len(sentiments)  IF ABSOLUTE(sentiments[I] – sentiments[I + 1]) > 5  mark sentences in sentences array  ENDIF  ENDFOR  result 🡨 join sentences from sentences array  RETURN result  END FUNCTION |

### TF-IDF

TF-IDF is an algorithm that identifies keywords in a text, based on the number of times that word occurs in other texts. I researched this algorithm from the source: *Manning et al. (2008) Introduction to Information Retrieval*. To achieve this the algorithm first needs to be ‘trained’ on a large corpus of text, to do this I will be using some of the corpora contained in the NLTK module, trying to get a good spread of the types of text that I expect my program to be used with. Specifically, I will be using the Brown Corpus, Gutenberg Corpus, Inaugural Address Corpus and Reuters Corpus, which cover Fiction, Speeches and News articles respectively. When each of these texts are passed in the document is analysed and each unique word it contains is identified, for each of these words the entry in the dictionary with that index is incremented by one, showing that it is contained in one more document. Then when a new text is passed in to the method the TF (Text Frequency) in the text is calculated and normalized using the formula , where is the number of times the word appears in that document, is the highest frequency of any word in that document and is a *smoothing* term, this is usually set to , which makes the range of the results smaller. The IDF (Inverse Document Frequency) is then calculated using the formula , where is the total number of documents it has been trained with and is the number of documents that contain that word. The TF-IDF for that word is then calculated using. The words are then ranked from highest TF-IDF to the lowest, allowing me to find the top five most important words, these are the keywords for that text. From the texts entered by the user, the algorithm can be trained further, from texts that will be typical for the system, because they are the texts actually being used. Therefore, over time the algorithm will be able to better analyse for keywords in texts the user enters, making this a machine learning algorithm.

|  |
| --- |
| Pseudo-code |
| CLASS TF\_IDF  CONSTRUCTOR  PROPERTY document\_frequency 🡨 {}  PROPERTY max\_occurences 🡨 1  END CONSTRUCTOR    METHOD count\_words(text)  words 🡨 tokenise text  text\_words 🡨 {}  FOR word IN words  IF word IS NOT stop word OR word IS NOT punctuation  current\_words 🡨 keys from text\_words dictionary  IF word IN current\_words  text\_words[word] ++  ELSE  text\_words[word] = 1  ENDIF  ENDIF  ENDFOR  END METHOD  METHOD train\_frequencies(text, title)  text\_words 🡨 count\_words(text)  document\_frequency[title] 🡨 text\_words  END METHOD    METHOD normalize\_frequency(word)  word\_count = count\_words(text)  A = 0.4  RETURN A +((1 - A) \*(word\_count[word]/highest in word\_count))  END METHOD  METHOD Find\_IDF(word)  no\_of\_docs = LEN(document\_frequency)  docs\_containing = 0  FOR document IN document\_frequency  IF word IN document  documents\_containing ++  ENDIF  ENDFOR  RETURN log(no\_of\_docs / 1 + documents\_containing)  END METHOD  METHOD check\_new\_text(text)  words 🡨 tokenise text  tf\_idfs 🡨 {}  FOR word IN words  IF word NOT a stop word AND word NOT punctuation  ntf = normalize\_frequency(word)  idf = find\_idf(word)  tf\_idfs[word] 🡨 ntf \* idf  ENDIF  ENDFOR  RETURN last five items of radix sorted list of tf\_idfs  END METHOD  END CLASS |

### Keyword Graphing

When a user enters a keyword in the keyword search, texts that have similar keywords will also show. To achieve this I will use a weighted graph, which represents how the words are linked. Each keyword will be a node in the graph an edge will be created when the two words are keywords for the same text, then for each time they are keywords in the same text, the weight of the edge will increase by 1. When a text is deleted, each of the edges that represented the keywords having a shared text, will have its weight reduced by 1, if the weight goes to 0 the edge will be deleted. When the user enters a keyword in the keyword search, a modified Dijkstra’s algorithm will be run, with the start node being the keyword that has been entered by the user. The initial value of the traversal will be set to 10, then as it traverses through the connected edges a value equal to the inverse of the weight of that node will be taken away from initial value. Therefore, the higher the value the better, so my modified Dijkstra’s algorithm optimizes the maximum distance between the two nodes. Within my system the depth of the traversal will be set to one node away, so only keywords that are directly connected to the initial node will be retrieved, however I will write the code so that as the number of texts increases, the depth of the traversal can be increased, as the words further away will become more relevant once there are more texts and the weightings increase. This implementation will use the graph that I defined in the previous data structures section. The following Pseudocode therefore just contains the modified Dijkstra’s algorithm.

|  |
| --- |
| Pseudo-code |
| FUNCTION dijkstra(graph, initial)  visited 🡨 {}  nodes 🡨 graph.nodes  maxWeight 🡨 MAX(graph.distances)  minWeight 🡨 MIN(graph.distances)  nodesToExplore 🡨 LEN(graph.edges[initial]) + 2  explored 🡨 1  WHILE explored < nodesToExplore  minNode = None  FOR node IN nodes  IF node IN visited  IF minNode IS None  minNode = node  ELIF visited[node] > visited[minNode]:  minNode = node  ENDIF  ENDIF  ENDFOR    IF minNode IS None  BREAK  ENDIF  nodes.remove(minNode)  currentWeight = visited[minNode]    FOR edge IN graph.edges[minNode]  weight 🡨 graph.distances[(minNode, edge)]  IF maxWeight – minWeight = 0  normalized 🡨 1  ELSE  Normalized 🡨 ((weight-min\_weight)/(max\_weight-min\_weight))+1  ENDIF  weight 🡨 current\_weight - (1 / normalised)  IF edge NOT IN visited OR weight > visited[edge]:  visited[edge] = weight  ENDIF  ENDFOR  explored 🡨 explored + 1  ENDWHILE  RETURN visited |

### Radix Sort

Radix sort is non-comparative sort that I will be using to rank the words found in the texts, in order to finds the 10 most used words in the text. I chose to use this text because of its high time efficiency. The radix sort has a time efficiency of where is the length of integer being sorted and is the number of integers to be sorted. This makes the radix sort more efficient (worse case) than other sorts, e.g. merge sort, that have efficiency, when is sufficiently large enough, so that, and as I will be using and will usually be greater than 100, this makes the radix sort more efficient.

|  |
| --- |
| Pseudo-code |
| FUNCTION radix\_sort(numbers\_to\_sort)  list\_length 🡨 LEN(numbers\_to\_sort)  number\_length 🡨 LEN(numbers\_to\_sort[0])  WHILE number\_length > 0  FOR x 🡨 0 TO number\_length  buckets = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  FOR number IN numbers\_to\_sort  buckets[number[x]] 🡨 buckets[number[x]] + 1  ENDFOR  FOR x 🡨 1 TO 10  buckets[x] 🡨 buckets[x] + buckets[x – 1]  ENDFOR  sort\_list 🡨 []  FOR x 🡨 0 TO list\_length  sort\_list.append(0)  ENDFOR  FOR number IN reversed(numbers\_to\_sort)  buckets[number[number\_length – 1]] 🡨 -1  sort\_list[buckets[number[number\_length – 1]]] = number  ENDFOR  length 🡨 length – 1  numbers\_to\_sort 🡨 sort\_list  ENDWHILE  RETURN numbers\_to\_sort  END FUNCTION |

## Database Structure

USER(UID, Username, Email, Password)

TEXT(TID, , , TextTitle, FHC, ReadingAge, Sentiment, Alliteration, Antithesis, Juxtaposition)

USER\_TEXT(, )

USER\_TEXT(CID, Category)

NOTIFICATION(, Message)

KEYWORD(WID, Word)

WORD\_TEXT(, , Score)

These tables will be created using these DDL scripts:

|  |  |
| --- | --- |
| TEXT | CREATE TABLE Text (  TID INTEGER PRIMARY KEY AUTOINCREMENT,  UID INTEGER,  TextTitle VARCHAR(100) NOT NULL,  FHC VARCHAR(100),  CID INTEGER,  ReadingAge REAL,  Sentiment REAL,  Alliteration INTEGER(1),  Antithesis INTEGER(1),  Juxtaposition INTEGER(1),  FOREIGN KEY(UID) REFERENCES User(UID)  FOREIGN KEY(CID) REFERENCES Categories(CID)); |
| USER | CREATE TABLE User(  UID INTEGER PRIMARY KEY AUTOINCREMENT,  UserName VARCHAR(30) NOT NULL,  Email VARCHAR(50),  Password VARCHAR(100) NOT NULL); |
| USER\_TEXT | CREATE TABLE User\_Text(  UID INTEGER,  TID INTEGER,  FOREIGN KEY(UID) REFERENCES User(UID),  FOREIGN KEY(TID) REFERENCES Text(TID),  PRIMARY KEY (UID, TID)); |
| CATEGORIES | CREATE TABLE Categories(  CID INTEGER PRIMARY KEY AUTOINCREMENT,  Cat VARCHAR(50)); |
| NOTIFICATIONS | CREATE TABLE Notifications(  UID INTEGER,  MESSAGE VARCHAR(140),  FOREIGN KEY(UID) REFERENCES User(UID)); |
| KEYWORDS | CREATE TABLE Keywords(  INTEGER PRIMARY KEY AUTOINCREMENT,  Word VARCHAR(50)); |
| WORD\_TEXT | CREATE TABLE text\_word(  TID INTEGER,  WID INTEGER,  Score INTEGER,  FOREIGN KEY(TID) REFERENCES Text(TID),  FOREIGN KEY(WID) REFERENCES Keywords(WID),  PRIMARY KEY (WID, TID)); |

## SQL Notation

Throughout my program I will be using SQL to access the database to retrieve information about the users and their texts. To do this I will have to use cross table parameterized SQL statements as well as some operations which will require multiple queries to work as necessary. The parameterised queries stop any sql injection attacks as it does not allow the user to end the query and write their own. Within the SQL statements, the parameters are represented by question marks.

### Search by Category

This set of two statements will search the database for all of the user’s texts that contain a certain keyword, both ones that have been shared with the user and ones that the user has inputted themselves and owns.

|  |  |
| --- | --- |
| SELECT Text.TextTitle, Text.FHC  FROM Text, Keywords, text\_word  WHERE Keywords.Word = ?  AND Keywords.WID = text\_word.WID  AND text\_word.TID = Text.TID  AND Text.UID = ? | SELECT Text.TextTitle, Text.FHC  FROM Text, Keywords, text\_word, User\_Text  WHERE Keywords.Word = ?  AND Keywords.WID = text\_word.WID  AND text\_word.TID = User\_Text.TID  AND User\_Text.TID = Text.TID  AND User\_Text.UID = ?; |

The first SQL query gets the text title and first hundred characters of all the texts that the user owns, that contain a certain keyword. To do this it finds the Word ID of the keyword that was inputted, then using the text word table, which contains the information as to which text has which keyword, it fins all the texts that have the keyword, and finally matches it to the text by finding the matching Text ID in the Text table. The second SQL query does the same task as the first, but for shared texts not owned texts. TO do this an extra step is inserted by matching the Text ID from the text word table to the Text ID in the user text table, where the user ID is the same as the one supplied, then this Text ID can be matched from the Text table, allowing the information to be retrieved. The results of these two queries can then be joined to find the full list of found texts.

### Add Text

As well as finding the texts the users have access to and searching the database, the database will also be used to manage the texts in the system and information about the texts.

|  |
| --- |
| INSERT INTO Text(UID, TextTitle, FHC, CID, ReadingAge, Sentiment, Alliteration, Antithesis, Juxtaposition)  VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?) |

### Find Text Owner

|  |
| --- |
| SELECT User.UserName  FROM User, Text, User\_Text  WHERE Text.UID = User.UID  AND TextTitle = ?  AND User\_Text.TID = Text.TID  AND User\_Text.UID = ?; |

This query allows the system to find the owner of a text, by retrieving their user ID. We are capable of doing this as we know that the user has access to this text and we know this will alwayss be the only text that this user has access to with this title. So we can look for all the texts the user has access to, then find the text that has the title specified, and from there find the owners ID, as stored in the user table.

### Delete User

To make sure that all the code independent, I have seperated some of the more complex operations into multiple queries, for example the following 3 queries are called when the user wants to delete their profile. The queries each do a separate thing, the first deletes the user notifications, the second, the records that show a text has been shared with that user, and the final one which deletes the users record from the user table.

|  |
| --- |
| DELETE FROM Notifications  WHERE UID = ?; |
| DELETE FROM User\_Text  WHERE UID = ?; |
| DELETE FROM User  WHERE UID = ?; |

## Validation and Exception Handling

Throughout the system the user will have the opportunity to input various types of data into the program. When the user submits file, the accepted files, will be pdf files, text files or word documents. When the user inputs a text, the file extension will be checked, if the file extension is not in the list of accepted file types then the file will be rejected and the user will be shown a message. The user can also input raw text. Because of the way that the marking of analysis works, the text is change in to markup, meaning that if the user inputted any html tags they would be interpreted and run. To stop the user inputting malicious scripts, I will find any html tags, or any text surrounded by <> tags, and remove them using regular expressions.

Validation will also be done when the user signs up to the system. The user will have to enter a username, password and email. The username will be checked that it has a length of 4 and 21. The password will be checked with the following regular expression: ^(?=.\*?\d)(?=.\*?[a-z])(?=.\*?[A-Z])[a-zA-Z\d]{6,}$. This checks that the password has at least one number, one lower case letter and one upper case letter, as well as having at least 6 characters. The email will also be checked to make sure it is in the correct form: ^[a-zA-Z0-9\_.]+@[a-zA-Z0-9]+\.[a-zA-z0-9]+$.

Finally, validation will be needed in the entry of keywords and titles for searching. Validation is needed here as these will be put in to queries for the database, any vulnerability could allow someone to perform an SQL injection attack. To avoid this, I will be using prepared SQL statements with parameterized queries, so instead of constructing a query after the user has inputted their data, I will be using queries that I have already written that just accept the users input as parameters, not allowing for any chance of an attack.

Within my system I have used exception handling in order to make sure that the server can never go down. The main way that I have done this is to add a try except block around every flask function, this it to make sure that any errors that occur during any part of the program are caught and handled, this is usually done by returning the user to the home page and flashing a message which explains that something has gone wrong. Any exception handling other than this should not be needed as no data which causes errors should be able to be entered into the program, due to the input nvalidation that has previously been described.

## Prototype

In order to get an idea of how my project is progressing I have decided to get some feedback from the user from an early build of the system. This build has the web interface and part of the main analysing algorithm implemented the latter of which takes a text file from the user, extracts the text, tokenizes it and finds the reading age and any alliteration. Below are some screenshots of what output the user would have seen while using the system at this stage.

### Initial User Feedback

After giving the user the prototype and allowing them to use it I interviewed the user again, to see his feelings on the project so far.

**How successful do you feel the current system is at meeting the objectives you had in mind?**

MM: The current system is a good starting point, it correctly took the text out of the file, the reading age score was approximately the reading age I would have assessed the text to be. The alliteration checker found most of the alliteration, however I think a slight change is needed catch edge cases such as words beginning with “Eu” and “U”.

**Is the UI for entering texts suitable for the system?**

MM: The interface contains all the necessary features that I hoped would be in the program, features such as the task bar at the top allow for easy navigation and it is made clear how to enter the texts, although some more labels would be appreciated so I can tell which input does what.

**Does the highlighting of the features make it obvious where features are?**

MM: The features can be easily seen with the highlighting. The statistics about the text, in this case the reading age, would be more easily readable if they were highlighted or more colourfully labeled as they tend to blend in to the background a bit when in black and white.

**Overall, do you think the project is on track to meet your expectations and needs?**

MM: I do believe that the project is definitely on track at the moment and will meet my needs when all of the previously discussed objectives have been met.

### Changes to the System

After meeting with the user I decided to implement the changes that he had proposed I make to prototype and do my own testing to make sure that they had been successful. The changes that I made were the following:

1. Changed the Alliteration algorithm to find the special cases of alliteration:
   1. ‘N’, ‘Kn’ and ‘Gn’
   2. ‘Eu’ and ‘U’
   3. ‘F’ and ‘Ph’
   4. ‘S’, ‘Ce’, ‘Ci’ and ‘Cy’
   5. ‘R’ and ‘Wr’
2. Add labels to the text entry fields that the user uses to enter files, raw text or a web address.
3. Add better labels to the statistics show about the texts as well as adding a coloured background to the figures so that they stand out more and can be more easily seen.

### Evaluation of changes

After making these changes, I personally tested my program to make sure that the objectives set by the user were met, to test objective one I wrote a file that contained all of the new special case types of alliteration. For objectives, two and three I observed the output that the program gave and made sure it met the user’s specification. Below are screenshots of the observed program output:

The first objective has been met as the program now correctly highlights alliteration in all of the special cases. Alongside this we can see that the new format for displaying the statistics, as defined in objective 3, makes the statistics as they now have their own labeled are of the display page and have a coloured background. Finally, objective 2 has been met as the input fields on the home page have now been labeled.

Overall, I now believe that all of the new objectives have been met, allowing me to move on with the rest of the system.

## Optimisation

While programming my project I discovered that the complexity of one of my algorithms was O(n3), this was hugely in efficient and was slowing my system down significantly. Specifically, this algorithm was the tf-idf algorithm, the one used to specify the keywords of the text. While running my program I attempted to input a large text consisting of 44000 words, when the algorithms were run the total time to complete them was 30 minutes. In order to reduce this time I decided to change the design of this algorithm to reduce its complexity to O(n). The original Pseudocode can be found on page 29, below is the pseudocode for the new more efficient version of the algorithm.

|  |
| --- |
| Pseudo-code |
| CLASS TF\_IDF  CONSTRUCTOR  PROPERTY document\_frequency 🡨 {}  PROPERTY number\_of\_documents 🡨 0  END CONSTRUCTOR    METHOD count\_words(text)  words 🡨 tokenise text  text\_words 🡨 {}  FOR word IN words  IF word IS NOT stop word OR word IS NOT punctuation  current\_words 🡨 keys from text\_words dictionary  IF word IN current\_words  text\_words[word] ++  ELSE  text\_words[word] = 1  ENDIF  ENDIF  ENDFOR  END METHOD  METHOD train\_frequencies(text)  text\_words 🡨 words tokenized text  for word in text\_words  if word is not a stop word or punctuation  document\_frequency[word] ++  number\_of\_documents ++  END METHOD    METHOD normalize\_frequency(word\_count, highest)  A = 0.4  RETURN A +((1 - A) \*(word\_count/highest))  END METHOD  METHOD Find\_IDF(word)  TRY  documents\_containing 🡨 document\_frequency[word] + 1  EXCEPT  documents\_containing 🡨 1  RETURN log(number\_of\_documents / documents\_containing)  END METHOD  METHOD check\_new\_text(text)  words 🡨 tokenise text  word\_count 🡨 count\_words(text)  highest 🡨 highest in word\_count  tf\_idfs 🡨 {}  FOR word IN words  IF word NOT a stop word AND word NOT punctuation  ntf = normalize\_frequency(word, highest)  idf = find\_idf(word)  tf\_idfs[word] 🡨 ntf \* idf  ENDIF  ENDFOR  RETURN last five items of radix sorted list of tf\_idfs  END METHOD  END CLASS |

After changing this algorithm the checking of the new text is now just O(n) because of the one for loop that loops around each of the words in the text. This change has caused a massive decrease in processing time, the same text which previous took 30 minutes to process now only takes around 30 seconds, an increase in efficiency of 60x. From these results I decidced to implement this code in to the final version of the system.

## Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No. | Objective being Tested | Test Description | Test Type | Test Data | Expected Output |
| 1 | 1 | Test the user can sign up, given valid data | Normal | drsphelps  [dp@gmail.com](mailto:dp@gmail.com)  Qwerty123  Qwerty123 | “You have been logged in” |
| 2 | 1 | Test to give an error if the user gives invalid data | Erroneous | drsphelps  EIDGEIAIEDB  qwerty123  Qwer12 | “Not a valid email address”  “Password must contain an upper/lower case letter and a number”  “Passwords do not match” |
| 3 | 1 | Checks the user can login to their account with the correct data | Normal | drsphelps  Qwerty123 | “You have been logged in” |
| 4 | 1 | Test to give an error if the user gives data that does not match an account | Erroneous | drphelp  qwerty12 | “Invalid login credentials” |
| 5 | 3a | Checks the user can input a supported file | Normal | Any docx, text or pdf file | Go to text view page |
| 6 | 3a | Check for an error if the filetype is not supported | Erroneous | Any file that is not a docx, text or pdf file | “File type not supported” |
| 7 | 3b | Test to see if the user can input raw text | Normal | “Hello, World!” | Text Display page is shown |
| 8 | 3b | Test to see if, if the user puts in any html tags then the tags will be removed | Erroneous | “<script> ” | “script” |
| 9 | 3c | Test that user can input a web address to be scraped | Normal | [www.bbc.co.uk/headline](http://www.bbc.co.uk/headline) | Shows the text from the webpage |
| 10 | 3c | If an invalid web address is entered test for an error | Erroneous | [www.bbc/headline](http://www.bbc/headline) | “Web address is not recognised” |
| 11 | 4a | Test that basic alliteration is highlighted | Normal | “Brian Bensons Building Buddies” | Alliteration will be highlighted |
| 12 | 4a | Test that more complex alliteration is highlighted | Extreme | “European Utopia” | Alliteration will be highlighted |
| 13 | 4a | Test that if there is no alliteration it is not highlighted | Erroneous | “Church Cats” | Alliteration is not highlighted |
| 14 | 4b | Test that any antithesis is highlighted | Normal | “Breathtakingly Antichrist” | Antithesis Highlighted |
| 15 | 4b | Test that words which have a sentiment difference on the very edge of the accepted limit is highlighted | Extreme | “Adore Anger” | Antithesis Highlighted |
| 16 | 4b | Test that if there is no antithesis it is not highlighted | Erroneous | “Church Cats” | Antithesis not Highlighted |
| 17 | 4c | Test that any juxtaposition is highlighted | Normal | “Breathtakingly. Antichrist.” | Antithesis Highlighted |
| 18 | 4c | Test that sentences which have a sentiment difference on the very edge of the accepted limit is highlighted | Extreme | “Adore. Anger.” | Juxtaposition Highlighted |
| 19 | 4c | Test that if there is no juxtaposition it is not highlighted | Erroneous | “Church. Cats.” | Juxtaposition not Highlighted |
| 20 | 5 | Test that by clicking on the different buttons displayed that the different analyses show | Normal | Selecting the on screen buttons | Changes to show the selected analysis |
| 21 | 6a | Test that the reading age calculated is close to the actual reading age. The actual will be found using another online reading age tool | Normal | “The Australian platypus is seemingly a hybrid of a mammal and reptilian creature” | The algorithm will be judged to be successful if the given reading age is within 15%. The required range is therefore 14.365 – 19.435 |
| 22 | 6b | Test that the sentiment of the text is correctly calculated | Normal | Positive Review from Amazon.com: “Really excellent little machine and so versatile, I just used one to bring an old PlayStation back to life, loads of fun.” | Positive Sentiment in range 2-5 |
| 23 | 6b | Test that if there are no words with sentiment scores, that the sentiment is just given as 0 | Extreme | “Hello World” | Sentiment: 0 |
| 24 | 7 | Test that a graph can be generated showing the most used words | Normal | Large Body of Text | Graph ranking the top ten most user words in order |
| 25 | 7 | Test that if no words are in the text that the graph shows no data | Extreme | “.” | Graph shown with no data |
| 26 | 7 | Test that a graph can be generated showing the most used punctuation | Normal | Large Body of Text | Graph ranking the most used punctuation in the text |
| 27 | 7 | Test that if there is no punctuation in the text that the graph shows no data | Extreme | “Hello” | Graph shown with no data |
| 28 | 8 | Test that the correct keywords are selected from the text | Normal | First page of “Harry Potter and the Philosopher’s Stone” | From reading the text I have found the keywords to be: “Dudley”, “Dursley”, “Potters”. I will judge the algorithm to be successful if 3 of the keywords match. |
| 29 | 8 | Test that if there are no words that an empty keyword box is shown | Extreme | “.” | Empty Keyword box |
| 30 | 9 | Test that the user can save a text to their profile | Normal | “test” | Save the text and make it accessible on the users profile |
| 31 | 9 | Test that the user cannot save two texts with the same name | Erroneous | Try to save another text with the same title “text” | “A file with this name already exists.” |
| 32 | 10 | Test that all the users are shown so that a text can be shared with them | Normal | All user have no texts with the same title as the text to be shared | All users displayed |
| 33 | 10 | Test that if all users already have a text of the same name it will not be shown | Extreme | All users have texts with the same name as the text to be shared | No users displayed |
| 34 | 10 | Test to see that the text is shared with the user if they are selected | Normal | Username to share with is clicked | Text is added to their profile under the shared section |
| 35 | 11 | Test that all texts are shown in the users profile | Normal | User clicks to enter profile page, while they have both owned and shared texts on their account | Texts are shown sorted into shared and owned texts |
| 36 | 11 | Test that even if there are no texts in the users profile, the profile still shows | Extreme | User clicks to enter the profile while they have no texts on their account | Sections are shown, but display “No texts found.” |
| 37 | 12a | Test that when a title is searched all texts containing that word in the title are shown | Normal | “test” | All texts with the string of characters “test” in are displayed, in both the owned and shared sections |
| 38 | 12a | Test that when a title is searched that is not in that users texts that no texts show | Extreme | “hello” when no texts on the server contain the string “hello” | Sections are shown, but display “No texts found.” |
| 39 | 12b | Test that the user can select a category and texts from that category will show | Normal | User selects a category from the dropdown menu, category selected has texts in on that account | Texts in that category are displayed in the owned and shared sections |
| 40 | 12b | Test that if the user has no texts in the category that an appropriate message is displayed | Erroneous | User selects a category from the dropdown menu, category selected does not texts in on that account | Sections are shown, but display “No texts found.” |
| 41 | 12c | Test that the user can input a keyword and all texts with that keyword will be shown | Normal | User enters a keyword | Texts with that keyword are shown` |
| 42 | 12c | Test that if no texts with the keyword are found that it will display an appropriate message | Erroneous | User enters a keyword | Sections are shown, but display “No texts found.” |
| 43 | 12c | Test that the user can input a keyword and all texts with that keyword will be shown as well as texts that contain similar keywords. Any words that share a common text are found to be similar | Normal | User enters a keyword | All texts that contain that keyword and any other keyword, which have a common text with the original keyword, are shown. |
| 44 | 13 | Test that the text will be deleted when the user presses the button to delete the text |  | User clicks delete on the text display | Text is deleted from server and not shown on the users profile or on any users profiles that the text had been shared with. |
| 45 | 14 | Test that a pdf containing the text will be downloaded when the user clicks download |  | User clicks download on the text display | Text is put into a pdf and downloaded to the user’s computer. |

Objective two will not be tested in the formal testing section as this is based on the design of the user interface; however, this objective has been tested in the previous prototype section when the user interface was shown to the end user.

# Code Overview

To give a better understanding of some of the more difficult to understand code sections, I have compiled a range of example from throughout my code base and provided annotation to help in the understanding of these sections.

## URL Structure

However, to first get an understanding of which URLs to use to find any of the information below is a table showing all of the URLs and what the appears on that page as well as which pages you can access to by links from that page. Every page contains the header which when not logged in give the user a link back to the homepage, the login page and the sign up page, and when logged in gives links to the homepage, profile page and the logout button. In the URL column sections surrounded by <> are parameters and allow for variable web addresses.

Each of the URLs is given from the root directory, which is the web address of the site, for all testing purposes I hosted the website locally, so to run the website the \_\_init\_\_.py file in the root directory needs to be run.

|  |  |  |
| --- | --- | --- |
| URL | Description | Links to… |
| / | Home page of the site, contains the 3 boxes for the user to input data (files, raw text and web addresses). | /uploadfile  /raw\_text  /webtext |
| /login/ | Login page, contains two boxes which allow the user to input their username and password and a button which submits the form. | / - When the user successfully logs in |
| /register/ | Register page, contains boxes for the user to input a username, email, password and to confirm their password. | / - When the user successfully registers |
| /uploadfile | Text Display, shows the text and analyses when the user uploads a file. | /save\_text  /changeview |
| /raw\_text | Text Display, shows the text and analyses when the user enters raw text. | /save\_text  /changeview |
| /webtext | Text Display, shows the text and analyses when the user enters a web address. | /save\_text  /changeview |
| /changeview/<analysis>  Analysis is the type of analysis to be shown | Analysis Display, shows analysis that the user selects when they are uploading a text | /save\_text  /changeview |
| /textdisplay/<textTitle>  /<analysis>  <textTitle> is the title of the text to be displayed  <analysis> is the analysed version of the text you want to see | Shows the user the text and all the statistics about it, buttons to allow the user change the analysis of the text that they wish to see | /download  /share  /deletetext |
| /save\_text | Doesn’t display anything, but through this link the users text is uploaded and stored onto the server and added to the database | /profile/<username> |
| /profile/<username>  <username> is the username of the persons profile you wish to view | Shows the user the texts that they own, and the texts that have been shared with them. Also has search boxes and sliders that allows them to search through their texts, a button that allows them to delete their account and one that allows them to change their password. | /textdisplay  /search\_titles  /search\_keyword  /search\_categories  /search\_values  /changepassword  /deleteaccount |
| /download | Downloads the text in a pdf |  |
| /share | Shows a list of the users that the text can be shared with | /share\_text |
| /share\_text | Adds the text to the user’s profile who you chose to share the text with | / |
| /deletetext | Removes the text from the system | /profile – redirects the user once the text is deleted |
| /search\_title | Displays the same page as /profile/, but only showing the texts that have a specific string in the title | Same as profile |
| /search\_keyword | Displays the profile, but only showing texts with a specific keyword or related keywords | Same as profile |
| /search\_categories | Displays profile, but only texts that are in a certain category | Same as profile |
| /search\_values | Displays profile, but only with texts that meet the language feature value requirements | Same as profile |
| /changepassword | Gives the user boxes to enter their old password, new password and to confirm the new password | /profile – redirects once the password has been changed |
| /deleteaccount | Lets the user delete their account | / - redirects once the account is deleted |

## Flask Control Function

The main control of my system takes place within the flask control functions. These functions take requests to a specific URL, process any data received and return the templates for the browser to render. Within these functions, the main processing algorithms and database queries are called. Flask functions have been discussed before in the design section of this write up, however this gives a more exhaustive overview of a specific function and how it links to the other sections of the program.

1. @app.route('/profile/<username>/search\_keywords', methods=["POST", "GET"])
2. **def** search\_keywords(username):
3. """
4. Looks for texts with a certain keyword and similar keywords
5. """
6. **try**:
7. with Database() as database:
8. user = database.checkForUser(username)
9. **if** user == session['username']:
10. **if** session['username'] == username:
11. session['id'] = database.getID(session['username'])
12. search\_string = request.form['keyword']
13. # Gets the graph from a file
14. with open("word\_graph.txt", "rb") as f:
15. G = pickle.load(f)
16. owned\_texts = []
17. shared\_texts = []
18. **if** search\_string **in** G.nodes:
19. # Gets all nodes that are connected to your current word
20. keywords = dijsktra(G, search\_string)
21. owned\_texts = []
22. shared\_texts = []
23. owned\_titles = []
24. **for** keyword, score **in** keywords.items():
25. owned, shared = database.searchKeyword(keyword, session['id'])
26. to\_format\_owned = []
27. to\_format\_shared = []
28. # Checks the text has not already been loaded
29. **for** text **in** owned:
30. **if** text[0] **not** **in** owned\_titles:
31. to\_format\_owned.append(text)
32. owned\_titles.append(text[0])
33. **for** text **in** shared:
34. **if** text[0] **not** **in** shared\_titles:
35. to\_format\_shared.append(text)
36. shared\_titles.append(text[0])
37. owned, shared = formatTexts(to\_format\_owned, to\_format\_shared
38. owned\_texts += owned
39. shared\_texts += shared
40. # Makes sure there is only one false statement in array
41. owned\_texts = [text **for** text **in** owned\_texts **if** text **is** **not** False]
42. **if** len(owned\_texts) == 0:
43. owned\_texts.append(False)
44. shared\_texts = [text **for** text **in** shared\_texts **if** text **is** **not** False]
45. **if** len(shared\_texts) == 0:
46. shared\_texts.append(False)
47. categories = database.loadCategories()
48. **return** render\_template('profile.html',
49. owned\_texts=owned\_texts,
50. shared\_texts=shared\_texts,
51. username=username,
52. categories=categories)
53. **else**:
54. flash("Keyword not found")
55. **return** redirect(request.referrer)
56. flash("You cannot view other users profiles")
57. **return** redirect(url\_for('index'))
58. flash("User %s not found" % username)
59. **return** redirect(url\_for('index'))
60. **except** Exception as e:
61. flash("Something went wrong, please try again")
62. **return** redirect(url\_for('index'))

The example function that I have chosen to use is the function that is run when the user tries to search for a keyword in their profile. The first part of the URL “profile” is the model as it defines the type of function we are using, this function is one of many profile functions, all of which load in texts and then render the users profile. The second part “<username>” is a parameter passed into the address when it is called, this is represented with the “<>” around the parameter name. This allows for content to be loaded dynamically based on the user who is accessing the page. Finally, “search\_keyword” is the method of the address as this is the actual processing that is going to be run.

Line 2 defines the subroutine that will be run when the user accesses the URL and again shows the parameter which will be accepted into that function. Lines 3, 4 and 5 are the docstrings, which are contained at the start of every function in the system. These give a brief overview of what the function does.

All the code within is contained in a try-except block as this means that the website will stay up no matter what the error is or when it occurs.

|  |
| --- |
| def checkForUser(self, username):  query = """SELECT UserName  FROM User  WHERE UserName = ?;"""  matched\_user = self.cursor.execute(query,[username]).fetchall()[0][0]  return matched\_user |

The database is then opened and the first call to the database is made, this query searches the database for a user with a specific user name and returns that users name, if it is not found this would return a None value. The function can then check for this None value to make sure the user exists.

The session variable is then used to find the user name of the person logged in on that client. This is to make sure that users can only access their own profiles. Data from the form that was submitted to access the page is used retrieved using “request.form[‘keyword’]”. The pickle module is then used to load a python object from the file “word\_graph.txt”, this is the file that contains the graph that shows the connection between the keyword, depending on the texts they have been used in. This graph is then used with Dijkstra’s algorithm to find all the connected keywords to the one inputted by the user.

Each one of the keywords return from the Dijkstra’s algorithm is the searched for in the database using two queries.

|  |  |
| --- | --- |
| SELECT Text.TextTitle, Text.FHC, text\_word.Score  FROM Text, Keywords, text\_word  WHERE Keywords.Word = ?  AND Keywords.WID = text\_word.WID  AND text\_word.TID = Text.TID  AND Text.UID = ?; | SELECT Text.TextTitle, Text.FHC, text\_word.Score  FROM Text, Keywords, text\_word, User\_Text  WHERE Keywords.Word = ?  AND Keywords.WID = text\_word.WID  AND text\_word.TID = User\_Text.TID  AND User\_Text.TID = Text.TID  AND User\_Text.UID = ?; |

The first of which finds all the texts the user owns that contain the keyword and the second which find all the texts that the user has access to that contains the keyword.

These texts are then returned to the function which adds all the texts to an array if they have not already been added which can be checked as each title is unique to that user. The texts are then formatted, so that they can be displayed in the right way. If the array containing the texts is empty “False” is added as a value so that the interface can display the message “No texts found”. The categories are then loaded in from the database.

The final clauses of every branch of the subroutine are to return a render template of specific html file, as well as sometimes to “flash” something. “Flash” is a method which adds a message to a session specific variable which can be looped through within each HTML document to display the message that have been added, meaning that messages can be displayed to the user on any webpage. The render template method returns the contents of the HTML file, and passing through certain variables, which are then rendered in the client’s web browser.

## Algorithm Training

As previously mentioned, I will be using the TFIDF algorithm which requires training before it can be used. The training it requires is a corpus of texts, so that when a new text is entered it can refer to the corpus of texts to find how common a word is. In order to be trained in the best way a wide range of texts from the genres that will be mostly commonly analysed will be needed. To do this I used the NLTK corpus module and selected texts from the brown, gutenburg, reuters and inaugural address corpora. These corpora together total the thousand texts containg over 3 million words. These will provide a basis for the algorithm to work on originally. As time goes on and more texts are entered in to the program the corous of available texts will grow as the texts entered are added. This will allow the keywords to be analysed more accurately, as they can be compared against the user entered texts as well as the initial corpora.

# Testing

Below you will find the result from the tests planned in the design section of this document. For each of these test there is an input and output screenshot, whether the test has passed or failed. If the test fails details of corrections and further testing can be found below.

|  |
| --- |
| Test 1 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test1Input.png |
| Test 1 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test1Output.png |
| Test 1 Success |
| Test 2 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test2Input.png |
| Test 2 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test2Output.png |
| Test 2 Success |
| Test 3 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test3Input.png |
| Test 3 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test1Output.png |
| Test 3 Success |
| Test 4 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test4Input.png |
| Test 4 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test4Output.png |
| Test 4 Success |
| Test 5 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test5Input.png |
| Test 5 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test5Ouptput.png |
| Test 5 Success |
| Test 6 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test6Input.png |
| Test 6 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test6Output.png |
| Test 6 Success |
| Test 7 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test7Input.png |
| Test 7 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test7Output.png |
| Test 7 Success |
| Test 8 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test8Input.png |
| Test 8 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test8Output.png |
| Test 8 Success |
| Test 9 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test9Input.png |
| Test 9 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test9Output.png |
| Test 9 Success |
| Test 10 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test10Input.png |
| Test 10 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test10Output.png |
| Test 10 Success |
| Test 11 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test11Input.png |
| Test 11 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test11OutputFailed.png |
| Test 11 Failed |
| Test 12 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test12Input.png |
| Test 12 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test12Output.png |
| Test 12 Success |
| Test 13 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test13Input.png |
| Test 13 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test13Output.png |
| Test 13 Success |
| Test 14 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test14input.png |
| Test 14 Output |
|  |
| Test 14 Success |
| Test 15 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test15Input.png |
| Test 15 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test15Output.png |
| Test 15 Success |
| Test 16 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test16Input.png |
| Test 16 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test16Ouput.png |
| Test 16 Success |
| Test 17 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test17Input.png |
| Test 17 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test17output.png |
| Test 17 Success |
| Test 18 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test18Input.png |
| Test 18 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test18Output.png |
| Test 18 Success |
| Test 19 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test19Input.png |
| Test 19 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test19Output.png |
| Test 19 Success |
| Test 20 Regular Selected |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test20Regular.png |
| Test 20 Alliteration Selected |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test20Alliteration.png |
| Test 20 Antithesis Selected |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test20Antithesis.png |
| Test 20 Juxtaposition Selected |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test20Juxtaposition.png |
| Test 20 Success |
| Test 21 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test21Input.png |
| Test 21 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test21Ouput.png |
| Test 21 Success |
| Test 22 Input |
| C:\Users\Dylan\OneDrive - UVHS\Year 13\FinalScreenshots\Test22Input.png |
| Test 22 Output |
| C:\Users\Dylan\OneDrive - UVHS\Year 13\FinalScreenshots\Test22Output.png |
| Test 22 Success |
| Test 23 Input |
| C:\Users\Dylan\OneDrive - UVHS\Year 13\FinalScreenshots\Test23Input.png |
| Test 23 Output |
| C:\Users\Dylan\OneDrive - UVHS\Year 13\FinalScreenshots\Test23Output.png |
| Test 23 Success |
| Test 24 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test2426Input.png |
| Test 24 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test24Output.png |
| Test 24 Success |
| Test 25 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test25Input.png |
| Test 25 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test25Output.png |
| Test 25 Success |
| Test 26 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test2426Input.png |
| Test 26 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test26Output.png |
| Test 26 Success |
| Test 27 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test27Input.png |
| Test 27 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test27Output.png |
| Test 27 Success |
| Test 28 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test28Input.png |
| Test 28 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test28Output.png |
| Test 28 Success |
| Test 29 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test29Input.png |
| Test 29 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test29Output.png |
| Test 29 Success |
| Test 30 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test30Input.png |
| Test 30 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test30Output.png |
| Test 30 Success |
| Test 31 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test31Input.png |
| Test 31 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test31Output.png |
| Test 31 Success |
| Test 32 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test32Output.png |
| Test 32 Success |
| Test 33 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test33Output.png |
| Test 33 Success |
| Test 34 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test34Output.png |
| Test 34 Success |
| Test 35 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test35Output.png |
| Test 35 Success |
| Test 36 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test36Output.png |
| Test 36 Success |
| Test 37 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test37Input.png |
| Test 37 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test37Output.png |
| Test 37 Success |
| Test 38 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test38Input.png |
| Test 38 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test36Output.png |
| Test 38 Success |
| Test 39 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test39Input.png |
| Test 39 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test39Output.png |
| Test 39 Success |
| Test 40 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test40Input.png |
| Test 40 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test40Output.png |
| Test 40 Success |
| Test 41 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test41Input.png |
| Test 41 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test41Output.png |
| Test 41 Success |
| Test 42 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test42Input.png |
| Test 42 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test42Output.png |
| Test 42 Success |
| Test 43 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test43Input.png |
| Test 43 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test43Output.png |
| Test 43 Success |
| Test 44 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test44Output.png |
| Test 44 Success |
| Test 45 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test45Input.png |
| Test 45 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test45Output.png |
| Test 45 Success |

## Test Corrections

|  |
| --- |
|  |

The only test to fail in my testing was test 11, this test was meant to find basic alliteration in the text and was given the test string “Brian Benson’s building buddies” which is a string of four alliterative words. After investigating this error further, I found that it was a result of the alliteration being at the end of the text. This was caused by the algorithm to fin alliteration not emptying the stack when the pointer reached the end of the array of words, meaning that any alliteration at the end of the text would be left on the stack and not marked in the text display. To fix this I added some code, which checked if the pointer had reached the end of the array and if the end had been reached to empty the stack and add marks where necessary. After implementing this new code I decided to re do the test that was failed in the original testing and make sure that this time around that the test was passed.

|  |
| --- |
| Test 11 Input |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test11Input.png |
| Test 11 Output |
| C:\Users\Dylan\Documents\Testing Screenshots\TestingScreenshots\Test11OutputPassed.png |
| Test 11 Success |

# Evaluation

|  |  |  |
| --- | --- | --- |
| Number | Objectives | Objective Met? |
| 1 | The user will need to able to create an account within the system, so that they can save any texts that they upload and so that other users can find them. The user then needs to be able to log into this account. To do this the user will have to enter a username, email and password   1. The username must be longer than seven characters. 2. The email must be in a valid format for an email e.g. [example@text.com](mailto:example@text.com) 3. The password must contain at least one uppercase letter, one lowercase letter and a number, as well as being eight or more characters long. | Yes, the user can register an account which they can log in to at a later time. |
| 2 | The system must be designed with a user-friendly user interface. To do this I will need to make sure that my interface is:   1. Easily navigable – Which will be achieved by making sure that the elements that the user will be interacting with are labelled and that the number of pages is small as possible. 2. Provides Information – The system must give the user all the information that is required, as well as making this information easily findable and shown in a usable way. | Yes, feedback from the user has shown that they think the user interface will meet their requirements. |
| 3 | The user must be able to input texts from any of their most used sources, these include text files, word documents, pdf files, raw text and from a website.   1. Files – The user must be able to browse any file in their file system and upload it on to the system, the text must then be checked to make sure the text is extracted from it so that it can be tokenized and analysed. 2. Raw Text – The raw text must be taken from the input, and sanitized to make sure that no code can be entered or injection attack can take place. 3. Website – The user must be able to enter a URL, from this the text must be scraped and fed into the system. | Yes, from the home page the user has the option to input a file, text or web address. The text is then taken from these sources and stored. |
| 4 | The analysis must include these language features:   1. Alliteration – Words that start with the same sounds will be highlighted, this will ignore ‘stop words’ (words that add no meaning, such as ‘and’ or ‘that’) 2. Antithesis – Consecutive words that have contrasting meanings will be highlighted 3. Juxtaposition – Consecutive sentences that contrast each other will both be highlighted | Yes, analysis is done and then stored within a python object. |
| 5 | The analysis should be shown on the text that the user has inputted and phrases or words that contain the language features should be highlighted. From here the user should be able to choose which analysis that they want to see at any time, which should change the highlighting of the text to the new analysis that the user has selected | Yes, analysis is highlighted on the text display page, the user has option to press a specific button to change the analysis. |
| 6 | When the analysis takes place, the system must also find out statistics about the text, these will be displayed with the user’s text:   1. Reading age – Using the Flesch-Kincaid Score, the system must work out the reading age of the text 2. Sentiment – Sentiment will be calculated using the sentiment of the words within the text and then averaged out. | Yes, statistics displayed underneath the text. Values calculated are very similar to the ones using other dedicated tools. |
| 7 | Most used words and punctuation from the text will also be found and these will be displayed as graphs on the page which displays the analysed text. | Yes, graphs are displayed at the bottom the page. |
| 8 | Another feature that is to be found is the keywords of the text, these will be displayed alongside the analysed text. | Yes, keywords are displayed with the text. |
| 9 | The user then must be able to save that text along with the analysis and the statistics onto the server. The user must before this give the text a unique title, if the title is not unique the user must be prompted to enter another. | Yes, the user can input a title and save the text, an error is shown if the title is not unique. |
| 10 | From the text that displays to the user the analysed texts, the user must be able to share the text with another user, this should be done by showing the user a list of the valid users. The user is valid if they do not have a text with the name of the text that you are sharing with them. | Yes, names of other users are shown and the text can be shared. |
| 11 | From the user’s profile page, they should be able to see their texts, sorted into texts that have been entered by them and ones that have been shared with them by other users. | Yes, texts are displayed on the profile page. |
| 12 | Also form their profile page the user should be able to search for a text; the results should be displayed in the same, with a section for shared texts and one for owned texts.   1. Search by Title – The user should be able to search for a string, if the title contains that string the text should be shown, this should not be limited to being the exact title and should not be case sensitive. 2. Search by Category – The user should be given a list of the categories of texts that are in the system, from which they can select a category and all texts that are owned by them or that they have access to should be shown. 3. Search by Keyword – The user should be able to search for a keyword in any of the texts. This could be expanded to be a smart search, so that when the user inputs a keyword, similar keywords are also searched for. The results should be ranked in order of how related the keywords are. | Partially Met, user can search by category and title. Keyword search also works and ‘smart search’ is implemented. However the texts are not ranked in order of the similarity. |
| 13 | The user should have the ability from within the text view to delete texts that they do not need to be stored on the server anymore. It should delete the text from the database, both for the owner and the people that access to it, delete the keywords and the delete the file that held the analysed text | Yes, a delete button is displayed, which when clicked deletes the text from the users account. |
| 14 | The system should include a download button in the text view that allows the user to download a copy of the plain text in a pdf file. | Yes, a delete button is displayed, which when clicked downloads the text as a pdf. |

## User Feedback

To make sure that my system is fit for purpose and meets all of the requirements/objectives for the project. I have shown my completed system to the end user and got their feedback. On top of this, I also have the feedback that I received during the prototyping phase of the project. From this feedback, I have already concluded that the basic analysis features and the user interface meet the requirements of the end user.

**Are text analysis features accurate and do they meet the requirements of your system?**

MM: The text analysis features seem to be very accurate. I was particularly impressed with the alliteration detection; it caught a wide variety of different cases not just words, which start with the same letter. They also meet the requirements of the system, as all the features that I wanted to be analysed are included. The sentiment and reading age values appear to very accurate as they very close what I expected and to other values that I have found online.

**Are the texts easily accessible and do you feel they are organised in a satisfactory way?**

MM: While using the system I found it very easy to access the texts as they are displayed very well on the profile page. The search tools were also very useful while trying to find texts. The sliders and checkboxes also helped to find texts that met certain requirements for texts.

**Did the sharing feature help the teachers collaborate on texts?**

MM: The sharing feature has helped immensely as individual teaches do not have to add texts, which they know that other teachers have already added into the system. This has saved lots of time and allowed teachers to cooperate a lot more on the texts that they use in their classes.

**Are there any expansions to the system that you think will be helpful?**

MM: Expansions such as adding more language features to the text analysis would be helpful to add scope to the system. The only other feature would be some sort of group feature that would allow teachers to upload texts to the group and not have to share every text that they want to be used in multiple classes.

**Overall, does the system meet the requirements and will you use the system as part of the department?**

MM: The system meets all of the requirements that I initially wanted in the system and is very easy to use. I can see the system being used in the department to store and analyse all the texts we want to use.

## Final System Evaluation

Although my project has been successful, I can see many expansions to the system that could be made in the future. The first of these is to complete objective 12 and have the keyword search by ranked in order of how related the words are. The second, as given in my user feedback, is to add a groups system, where all members of the same group have access to a common set of texts and analysis. Another feature that was discussed in the user feedback is an expansion to the language features that are analysed when the text is uploaded, this is easily expandable as I have built this system to be easily expandable, the algorithms need to be written and then the result needs to be added to the analysis dictionary which is easily done. There are also some longer-term expansions that I think could be made such as implementing the algorithms in to an app for android or iOS.

Overall, I think, through my talk with the end user and accessing the system myself against the objectives, that the system has been successful and has met nearly all of the objectives and will be a system that can be used within any English department or by anybody who has a need to analyse texts.

# Code Appendix

/\_\_init\_\_.py

|  |  |
| --- | --- |
| 1 | # IMPORT file dependancies |
| 2 | from flask import render\_template, request, url\_for, Flask, redirect, session, flash, Markup, make\_response |
| 3 | from static.scripts.sorts import quick\_sort, radix\_sort |
| 4 | from static.scripts.form import Field, reg\_form |
| 5 | from werkzeug.utils import secure\_filename |
| 6 | from static.db.dbclass import Database |
| 7 | from passlib.hash import pbkdf2\_sha256 |
| 8 | from static.scripts.main import TF\_IDF |
| 9 | from collections import defaultdict |
| 10 | from static.scripts import main |
| 11 | from functools import wraps |
| 12 | from math import log |
| 13 | import sqlite3 |
| 14 | import pdfkit |
| 15 | import shutil |
| 16 | import pickle |
| 17 | import os |
| 18 | import re |
| 19 |  |
| 20 | # Defualt upload directory |
| 21 | UPLOAD\_FOLDER = "./static/userfiles/" |
| 22 | # Filetypes the program allows |
| 23 | allowed\_extensions = ("pdf", "txt", "docx") |
| 24 | # File currently being used |
| 25 | current\_file = None |
| 26 |  |
| 27 | # Starting flask application |
| 28 | app = Flask(\_\_name\_\_) |
| 29 | # Setting specific upload folder - changes later based on session user |
| 30 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER |
| 31 |  |
| 32 |  |
| 33 | def TextDelete(texttitle): |
| 34 | """ |
| 35 | Deletes text object file, database records and any keywords in the graph. |
| 36 | """ |
| 37 |  |
| 38 | path = app.config['UPLOAD\_FOLDER'] + \ |
| 39 | '/objects/' + texttitle + '.txt' |
| 40 | with Database() as database: |
| 41 | database.deleteText(texttitle, session['id']) |
| 42 |  |
| 43 | # Loads in the file to be deleted and the keyword graph |
| 44 | with open(path, "rb") as objectfile: |
| 45 | current\_file = pickle.load(objectfile) |
| 46 | keywords = current\_file.stats['Key Words'] |
| 47 | with open("word\_graph.txt", "rb") as graphfile: |
| 48 | word\_graph = pickle.load(graphfile) |
| 49 |  |
| 50 | # Reduces each edge connected to the current file keywords |
| 51 | for keyword in keywords: |
| 52 | word\_graph.add\_node(keyword[0]) |
| 53 | for k in keywords: |
| 54 | if k[0] != keyword[0]: |
| 55 | word\_graph.reduce\_edge(keyword[0], k[0]) |
| 56 |  |
| 57 | # Rewrites the graph object file |
| 58 | with open("word\_graph.txt", "wb") as graphfile: |
| 59 | pickle.dump(word\_graph, graphfile) |
| 60 |  |
| 61 | # Deletes the object file |
| 62 | os.remove(path) |
| 63 |  |
| 64 |  |
| 65 | def formatTexts(owned, shared): |
| 66 | """ |
| 67 | Formats the texts recived from a database search to the format used to |
| 68 | display them in the users profile page. |
| 69 | """ |
| 70 | owned\_texts = [] |
| 71 | shared\_texts = [] |
| 72 | # Catches error if there is no score from the databse search |
| 73 | try: |
| 74 | for text in range(len(owned)): |
| 75 | owned\_texts.append( |
| 76 | {'title': owned[text][0], 'body': owned[text][1], 'score': owned[text][2]}) |
| 77 | for text in range(len(shared)): |
| 78 | shared\_texts.append( |
| 79 | {'title': shared[text][0], 'body': shared[text][1], 'score': shared[text][2]}) |
| 80 | except: |
| 81 | for text in range(len(owned)): |
| 82 | owned\_texts.append( |
| 83 | {'title': owned[text][0], 'body': owned[text][1]}) |
| 84 | for text in range(len(shared)): |
| 85 | shared\_texts.append( |
| 86 | {'title': shared[text][0], 'body': shared[text][1]}) |
| 87 | # Adds False if the either of the text arrays are empty |
| 88 | if len(owned\_texts) == 0: |
| 89 | owned\_texts.append(False) |
| 90 | if len(shared\_texts) == 0: |
| 91 | shared\_texts.append(False) |
| 92 | return owned\_texts, shared\_texts |
| 93 |  |
| 94 |  |
| 95 | def check\_extension(f): |
| 96 | """ |
| 97 | Gets the extension of the submitted file |
| 98 | """ |
| 99 | parts = f.split('.') |
| 100 | last = parts[len(parts) - 1] |
| 101 | return last in allowed\_extensions |
| 102 |  |
| 103 |  |
| 104 | def login\_required(f): |
| 105 | """ |
| 106 | Wrapper for function that require the user to be logged in to access the function |
| 107 | """ |
| 108 | @wraps(f) |
| 109 | def wrap(\*args, \*\*kwargs): |
| 110 | if 'logged\_in' in session: |
| 111 | return f(\*args, \*\*kwargs) |
| 112 | else: |
| 113 | flash("You need to login first") |
| 114 | return redirect(url\_for('login\_page')) |
| 115 | return wrap |
| 116 |  |
| 117 |  |
| 118 | def check\_pass(password, confirmed): |
| 119 | """ |
| 120 | Checks the passwords entered are the same |
| 121 | Checks they meet the complexity criteria |
| 122 | """ |
| 123 | # Requires at least one digit, a lower case letter, |
| 124 | # an upper case letter and has at least 6 characters |
| 125 | password\_regex = r"^(?=.\*?\d)(?=.\*?[a-z])(?=.\*?[A-Z])[a-zA-Z\d]{6,}$" |
| 126 | regex = re.compile(password\_regex) |
| 127 | if password == confirmed: |
| 128 | if regex.match(pas): |
| 129 | return "Passed" |
| 130 | else: |
| 131 | return "Password needs to have an upper and lowercase letter and a number." |
| 132 | else: |
| 133 | return "Passwords do not match." |
| 134 |  |
| 135 |  |
| 136 | class Graph: |
| 137 | """ |
| 138 | Graph data structure used to map links between keywords |
| 139 |  |
| 140 | Properties: |
| 141 | nodes(set) - set of nodes in the Graph |
| 142 | edges(dict: values are set to be lists) - connection between nodes |
| 143 | distances(dict) - stores the weightings for each edge |
| 144 |  |
| 145 | Methods: |
| 146 | add\_node - adds another node to the node set |
| 147 | add\_edge - if edge exists. increases the weight by 1, otherwise adds |
| 148 | edge and sets weight to 1 |
| 149 | reduce edge - reduces a edges weight by 1, if weight is then 0, |
| 150 | removes edge |
| 151 | """ |
| 152 |  |
| 153 | def \_\_init\_\_(self): |
| 154 | self.nodes = set() |
| 155 | self.edges = defaultdict(list) |
| 156 | self.distances = {} |
| 157 |  |
| 158 | def add\_node(self, value): |
| 159 | self.nodes.add(value) |
| 160 |  |
| 161 | def add\_edge(self, from\_node, to\_node): |
| 162 | if to\_node in self.edges[from\_node]: |
| 163 | self.distances[(from\_node, to\_node)] += 1 |
| 164 | else: |
| 165 | self.edges[from\_node].append(to\_node) |
| 166 | self.distances[(from\_node, to\_node)] = 1 |
| 167 |  |
| 168 | def reduce\_edge(self, from\_node, to\_node): |
| 169 | if self.distances[(from\_node, to\_node)] > 1: |
| 170 | self.distances[(from\_node, to\_node)] -= 1 |
| 171 | else: |
| 172 | self.edges[from\_node].remove(to\_node) |
| 173 | if len(self.edges[from\_node]) == 0: |
| 174 | self.edges.pop(from\_node) |
| 175 | self.distances.pop((from\_node, to\_node)) |
| 176 |  |
| 177 |  |
| 178 | def dijsktra(graph, initial): |
| 179 | """ |
| 180 | Performs dijsktras algorithm from a certain node |
| 181 | """ |
| 182 | # Sets initial node score to 10 |
| 183 | visited = {initial: 10} |
| 184 |  |
| 185 | nodes = set(graph.nodes) |
| 186 | max\_weight = graph.distances[max(graph.distances, key=graph.distances.get)] |
| 187 | min\_weight = graph.distances[min(graph.distances, key=graph.distances.get)] |
| 188 |  |
| 189 | # Defines the number of nodes to explore as the number of conected nodes |
| 190 | nodes\_to\_explore = len(graph.edges[initial]) + 2 |
| 191 | explored = 1 |
| 192 | while explored < nodes\_to\_explore: |
| 193 | # Finds nodes with maximum value that has not yet been explored |
| 194 | max\_node = None |
| 195 | for node in nodes: |
| 196 | if node in visited: |
| 197 | if max\_node is None: |
| 198 | max\_node = node |
| 199 | elif visited[node] > visited[max\_node]: |
| 200 | max\_node = node |
| 201 |  |
| 202 | if max\_node is None: |
| 203 | break |
| 204 |  |
| 205 | nodes.remove(max\_node) |
| 206 | current\_weight = visited[max\_node] |
| 207 |  |
| 208 | # Finds score of the next node if node has already been visited |
| 209 | # changes score if it is greater |
| 210 | for edge in graph.edges[max\_node]: |
| 211 | weight = graph.distances[(max\_node, edge)] |
| 212 | if max\_weight - min\_weight == 0: |
| 213 | normalised = 1 |
| 214 | else: |
| 215 | normalised = ((weight - min\_weight) / (max\_weight - min\_weight)) + 1 |
| 216 | weight = current\_weight - (1 / normalised) |
| 217 | if edge not in visited or weight > visited[edge]: |
| 218 | visited[edge] = round(weight, 2) |
| 219 | explored += 1 |
| 220 |  |
| 221 | return visited |
| 222 |  |
| 223 |  |
| 224 | # Error Handler Methods |
| 225 | @app.errorhandler(404) |
| 226 | def page\_not\_found(e): |
| 227 | """ |
| 228 | Serves '404:Page Not Found' error page |
| 229 | """ |
| 230 | return render\_template('404.html'), 404 |
| 231 |  |
| 232 |  |
| 233 | @app.errorhandler(405) |
| 234 | def method\_not\_allowed(e): |
| 235 | """ |
| 236 | Serves '405:Method Not Allowed' error page |
| 237 | """ |
| 238 | return render\_template('405.html'), 405 |
| 239 |  |
| 240 |  |
| 241 | @app.errorhandler(500) |
| 242 | def internal\_server\_error(e): |
| 243 | """ |
| 244 | Serves '500:Internal Server Error' error page |
| 245 | """ |
| 246 | return render\_template('500.html'), 500 |
| 247 |  |
| 248 |  |
| 249 | @app.route('/') |
| 250 | def index(): |
| 251 | """ |
| 252 | Serves the home page at the '/' route |
| 253 | Loads notifications for the logged in user |
| 254 | """ |
| 255 | try: |
| 256 | with Database() as db: |
| 257 | notifs = db.getNotifs(session['username']) |
| 258 | b\_notifs = [] |
| 259 | for i in range(len(notifs) - 1, -1, -1): |
| 260 | b\_notifs.append(notifs[i]) |
| 261 | session['notifs'] = b\_notifs |
| 262 | return render\_template( |
| 263 | 'index.html', |
| 264 | notifs=notifs |
| 265 | ) |
| 266 | except: |
| 267 | return render\_template( |
| 268 | 'index.html') |
| 269 |  |
| 270 |  |
| 271 | @app.route('/uploadfile', methods=["POST", "GET"]) |
| 272 | @login\_required |
| 273 | def upload\_file(): |
| 274 | """ |
| 275 | Upload link for files |
| 276 | Creates analyser object |
| 277 | Serves upload text display page. |
| 278 | """ |
| 279 | try: |
| 280 | global current\_file |
| 281 | if request.method == "POST": |
| 282 | # Validates a file has been uploaded |
| 283 | if 'file' not in request.files: |
| 284 | flash("No file submitted") |
| 285 | return redirect(url\_for('index')) |
| 286 |  |
| 287 | f = request.files['file'] |
| 288 | if f.filename == '': |
| 289 | flash("No file submitted") |
| 290 | return redirect(url\_for('index')) |
| 291 |  |
| 292 | if app.config['UPLOAD\_FOLDER'] == UPLOAD\_FOLDER: |
| 293 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 294 | session['username'] |
| 295 |  |
| 296 | if check\_extension(f.filename): |
| 297 | # Makes sure filename is safe |
| 298 | filename = secure\_filename(f.filename) |
| 299 | filepath = app.config['UPLOAD\_FOLDER'] + '/files/' + filename |
| 300 | # Saves the uploaded file |
| 301 | f.save(filepath) |
| 302 | # Removes extension from filename |
| 303 | filename = filename.replace('.txt', '') |
| 304 | filename = filename.replace('.pdf', '') |
| 305 | filename = filename.replace('.docx', '') |
| 306 |  |
| 307 | current\_file = main.Analyser(filepath, filename) |
| 308 | analysed\_texts = current\_file.analysed\_texts |
| 309 | text\_facts = current\_file.stats |
| 310 | with Database() as db: |
| 311 | categories = db.loadCategories() |
| 312 | keywords = '' |
| 313 | for word in text\_facts['Key Words']: |
| 314 | keywords += word[0] + ", " |
| 315 | keywords = keywords[:-2] |
| 316 | return render\_template('textdisplay.html', |
| 317 | title=current\_file.title, |
| 318 | texts=analysed\_texts, |
| 319 | text=analysed\_texts['Regular'], |
| 320 | facts=text\_facts, |
| 321 | ext=current\_file.text.ext, |
| 322 | categories=categories, |
| 323 | keywords=keywords, |
| 324 | upload=True) |
| 325 |  |
| 326 | else: |
| 327 | flash("File type not allowed") |
| 328 | return redirect(url\_for('index')) |
| 329 |  |
| 330 | else: |
| 331 | return redirect(url\_for('index')) |
| 332 | except Exception as e: |
| 333 | flash("Something went wrong, please try again") |
| 334 | return redirect(url\_for('index')) |
| 335 |  |
| 336 |  |
| 337 | @app.route('/raw\_text/', methods=["POST", "GET"]) |
| 338 | @login\_required |
| 339 | def raw\_text\_upload(): |
| 340 | """ |
| 341 | Upload path for raw text, creates a text file with the text in |
| 342 | Creates analyser object |
| 343 | """ |
| 344 | try: |
| 345 | global current\_file |
| 346 | if request.method == "POST": |
| 347 | raw\_text = request.form['raw\_text'] |
| 348 | # Checks text is not empty |
| 349 | raw\_text = raw\_text.strip('<>') |
| 350 | if raw\_text != '': |
| 351 | if app.config['UPLOAD\_FOLDER'] == UPLOAD\_FOLDER: |
| 352 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 353 | session['username'] |
| 354 | filepath = filepath = app.config[ |
| 355 | 'UPLOAD\_FOLDER'] + '/files/raw.txt' |
| 356 | filename = 'raw' |
| 357 | # Writes file with raw text in |
| 358 | with open(filepath, 'w') as f: |
| 359 | f.write(raw\_text) |
| 360 |  |
| 361 | # Makes actual analyser object |
| 362 | current\_file = main.Analyser(filepath, filename) |
| 363 | analysed\_texts = current\_file.analysed\_texts |
| 364 | text\_facts = current\_file.stats |
| 365 | with Database() as db: |
| 366 | categories = db.loadCategories() |
| 367 | keywords = '' |
| 368 | for word in text\_facts['Key Words']: |
| 369 | keywords += word[0] + ", " |
| 370 | keywords = keywords[:-2] |
| 371 | return render\_template('textdisplay.html', |
| 372 | title=current\_file.title, |
| 373 | texts=analysed\_texts, |
| 374 | text=analysed\_texts['Regular'], |
| 375 | facts=text\_facts, |
| 376 | keywords=keywords, |
| 377 | categories=categories, |
| 378 | ext=current\_file.text.ext, |
| 379 | upload=True) |
| 380 | except Exception as e: |
| 381 | flash(e) |
| 382 | return redirect(url\_for('index')) |
| 383 |  |
| 384 |  |
| 385 | @app.route('/changepassword/', methods=["POST", "GET"]) |
| 386 | @login\_required |
| 387 | def changepassword(): |
| 388 | """ |
| 389 | Allows the user to change their password, changes password in database |
| 390 | """ |
| 391 | try: |
| 392 | if request.method == 'POST': |
| 393 | # Makes sure the passwords match and that it meets complexity |
| 394 | validate = check\_pass( |
| 395 | request.form['newpass'], request.form['connewpass']) |
| 396 | if validate == "Passed": |
| 397 | data = [request.form['newpass'], session[ |
| 398 | 'username'], request.form['oldpass']] |
| 399 | with Database() as database: |
| 400 | database.updateUserPassword(data) |
| 401 | return redirect(url\_for('profile', username=session['username'])) |
| 402 | else: |
| 403 | flash(validate) |
| 404 | return render\_template('changepass.html') |
| 405 |  |
| 406 | else: |
| 407 | return render\_template('changepass.html') |
| 408 |  |
| 409 | except Exception as e: |
| 410 | flash("Oops, something went wrong... Try again.") |
| 411 | return render\_template('changepass.html') |
| 412 |  |
| 413 |  |
| 414 | @app.route('/profile/<username>') |
| 415 | @login\_required |
| 416 | def profile(username): |
| 417 | """ |
| 418 | Gets the users texts from the database and shows them on their profile page |
| 419 | """ |
| 420 | try: |
| 421 | with Database() as database: |
| 422 | # Makes sure the user exists |
| 423 | user = database.checkForUser(username) |
| 424 | if user == session['username']: |
| 425 | if session['username'] == username: |
| 426 | session['id'] = database.getID(session['username']) |
| 427 | owned\_texts, shared\_texts = formatTexts(\*database.getUsersTexts(session['id'])) |
| 428 | categories = database.loadCategories() |
| 429 | return render\_template('profile.html', |
| 430 | owned\_texts=owned\_texts, |
| 431 | shared\_texts=shared\_texts, |
| 432 | username=username, |
| 433 | categories=categories) |
| 434 | flash("You cannot view other users profiles") |
| 435 | return redirect(url\_for('index')) |
| 436 | flash("User %s not found" % username) |
| 437 | return redirect(url\_for('index')) |
| 438 | except Exception as e: |
| 439 | flash("Something went wrong, please try again") |
| 440 | return redirect(url\_for('index')) |
| 441 |  |
| 442 |  |
| 443 | @app.route('/profile/<username>/search\_titles', methods=["POST", "GET"]) |
| 444 | def search\_titles(username): |
| 445 | """ |
| 446 | Search through texts with titles that contain a specified string |
| 447 | """ |
| 448 | try: |
| 449 | with Database() as database: |
| 450 | user = database.checkForUser(username) |
| 451 | if user == session['username']: |
| 452 | if session['username'] == username: |
| 453 | session['id'] = database.getID(session['username']) |
| 454 | search\_string = request.form['title'] |
| 455 | # Gets texts from db and formats them |
| 456 | owned\_texts, shared\_texts = formatTexts(\*database.getUsersTexts(session['id'], search\_string)) |
| 457 | categories = database.loadCategories() |
| 458 | return render\_template('profile.html', |
| 459 | owned\_texts=owned\_texts, |
| 460 | shared\_texts=shared\_texts, |
| 461 | username=username, |
| 462 | categories=categories) |
| 463 | flash("You cannot view other users profiles") |
| 464 | return redirect(url\_for('index')) |
| 465 | flash("User %s not found" % username) |
| 466 | return redirect(url\_for('index')) |
| 467 | except Exception as e: |
| 468 | flash("Something went wrong, please try again") |
| 469 | return redirect(url\_for('index')) |
| 470 |  |
| 471 |  |
| 472 | @app.route('/profile/<username>/search\_keywords', methods=["POST", "GET"]) |
| 473 | def search\_keywords(username): |
| 474 | """ |
| 475 | Looks for texts with a certain keyword and similar keywords |
| 476 | """ |
| 477 | try: |
| 478 | with Database() as database: |
| 479 | user = database.checkForUser(username) |
| 480 | if user == session['username']: |
| 481 | if session['username'] == username: |
| 482 | session['id'] = database.getID(session['username']) |
| 483 | search\_string = request.form['keyword'] |
| 484 | # Gets the graph from a file |
| 485 | with open("word\_graph.txt", "rb") as f: |
| 486 | G = pickle.load(f) |
| 487 | owned\_texts = [] |
| 488 | shared\_texts = [] |
| 489 | if search\_string in G.nodes: |
| 490 | # Gets all nodes that are connected to your current word |
| 491 | keywords = dijsktra(G, search\_string) |
| 492 | owned\_texts = [] |
| 493 | shared\_texts = [] |
| 494 | owned\_titles = [] |
| 495 | for keyword, score in keywords.items(): |
| 496 | owned, shared = database.searchKeyword(keyword, session['id']) |
| 497 | to\_format\_owned = [] |
| 498 | to\_format\_shared = [] |
| 499 | # Checks the text has not already been loaded |
| 500 | for text in owned: |
| 501 | if text[0] not in owned\_titles: |
| 502 | to\_format\_owned.append(text) |
| 503 | owned\_titles.append(text[0]) |
| 504 | for text in shared: |
| 505 | if text[0] not in shared\_titles: |
| 506 | to\_format\_shared.append(text) |
| 507 | shared\_titles.append(text[0]) |
| 508 | owned, shared = formatTexts(to\_format\_owned, to\_format\_shared) |
| 509 | owned\_texts += owned |
| 510 | shared\_texts += shared |
| 511 | # Makes sure there is only one false statement in array |
| 512 | owned\_texts = [text for text in owned\_texts if text is not False] |
| 513 | if len(owned\_texts) == 0: |
| 514 | owned\_texts.append(False) |
| 515 | shared\_texts = [text for text in shared\_texts if text is not False] |
| 516 | if len(shared\_texts) == 0: |
| 517 | shared\_texts.append(False) |
| 518 | categories = database.loadCategories() |
| 519 | return render\_template('profile.html', |
| 520 | owned\_texts=owned\_texts, |
| 521 | shared\_texts=shared\_texts, |
| 522 | username=username, |
| 523 | categories=categories) |
| 524 | else: |
| 525 | flash("Keyword not found") |
| 526 | return redirect(request.referrer) |
| 527 | flash("You cannot view other users profiles") |
| 528 | return redirect(url\_for('index')) |
| 529 | flash("User %s not found" % username) |
| 530 | return redirect(url\_for('index')) |
| 531 | except Exception as e: |
| 532 | flash("Something went wrong, please try again") |
| 533 | return redirect(url\_for('index')) |
| 534 |  |
| 535 |  |
| 536 | @app.route('/profile/<username>/search\_category', methods=["POST", "GET"]) |
| 537 | def search\_category(username): |
| 538 | """ |
| 539 | Gets texts within a certain category |
| 540 | """ |
| 541 | try: |
| 542 | with Database() as database: |
| 543 | user = database.checkForUser(username) |
| 544 | if user == session['username']: |
| 545 | if session['username'] == username: |
| 546 | session['id'] = database.getID(session['username']) |
| 547 | category = request.args.get('category') |
| 548 | owned\_texts, shared\_texts = formatTexts(\*database.searchCategories(category, session['id'])) |
| 549 | categories = database.loadCategories() |
| 550 | return render\_template('profile.html', |
| 551 | owned\_texts=owned\_texts, |
| 552 | shared\_texts=shared\_texts, |
| 553 | username=username, |
| 554 | categories=categories) |
| 555 | flash("You cannot view other users profiles") |
| 556 | return redirect(url\_for('index')) |
| 557 | flash("User %s not found" % username) |
| 558 | return redirect(url\_for('index')) |
| 559 | except Exception as e: |
| 560 | flash("Something went wrong, please try again") |
| 561 | return redirect(url\_for('index')) |
| 562 |  |
| 563 |  |
| 564 | @app.route('/profile/<username>/search', methods=["POST", "GET"]) |
| 565 | def search\_values(username): |
| 566 | """ |
| 567 | Search for texts that have a certain language feature |
| 568 | Reading age and sentiment are in a certain range |
| 569 | """ |
| 570 | try: |
| 571 | with Database() as database: |
| 572 | user = database.checkForUser(username) |
| 573 | if user == session['username']: |
| 574 | if session['username'] == username: |
| 575 | features = ["Alliteration", "Antithesis", "Juxtaposition"] |
| 576 | needs\_feature = {} |
| 577 | for feature in features: |
| 578 | if request.form.get(feature) == "on": |
| 579 | needs\_feature[feature] = 1 |
| 580 | else: |
| 581 | needs\_feature[feature] = 0 |
| 582 | extra\_query = ";" |
| 583 | for feature, has in needs\_feature.items(): |
| 584 | if has == 1: |
| 585 | extra = "AND Text." + feature + " = 1 " |
| 586 | extra\_query = extra + extra\_query |
| 587 | sentiment\_above = float(request.form["SSA"]) |
| 588 | sentiment\_below = float(request.form["SSB"]) |
| 589 | reading\_age\_above = float(request.form["RAS"]) |
| 590 | if sentiment\_above >= sentiment\_below: |
| 591 | flash("Invalid Search...") |
| 592 | return redirect(request.referrer) |
| 593 | data = [session['id'], reading\_age\_above, |
| 594 | sentiment\_above, sentiment\_below] |
| 595 | owned\_texts, shared\_texts = formatTexts(\*database.searchTexts(data, extra\_query)) |
| 596 | categories = database.loadCategories() |
| 597 | return render\_template('profile.html', |
| 598 | owned\_texts=owned\_texts, |
| 599 | shared\_texts=shared\_texts, |
| 600 | username=username, |
| 601 | categories=categories) |
| 602 | flash("You cannot view other users profiles") |
| 603 | return redirect(url\_for('index')) |
| 604 | flash("User %s not found" % username) |
| 605 | return redirect(url\_for('index')) |
| 606 | except Exception as e: |
| 607 | flash("Something went wrong, please try again") |
| 608 | return redirect(url\_for('index')) |
| 609 |  |
| 610 |  |
| 611 | @app.route('/uploadfile/<analysis>', methods=["POST", "GET"]) |
| 612 | @login\_required |
| 613 | def changeview(analysis): |
| 614 | """ |
| 615 | Loads a different analysis of a certain text |
| 616 | """ |
| 617 | try: |
| 618 | analysed\_texts = current\_file.analysed\_texts |
| 619 | text\_facts = current\_file.stats |
| 620 | with Database() as db: |
| 621 | categories = db.loadCategories() |
| 622 | keywords = '' |
| 623 | for word in text\_facts['Key Words']: |
| 624 | keywords += word[0] + ", " |
| 625 | keywords = keywords[:-2] |
| 626 | return render\_template('textdisplay.html', |
| 627 | title=current\_file.title, |
| 628 | ext=current\_file.text.ext, |
| 629 | texts=analysed\_texts, |
| 630 | keywords=keywords, |
| 631 | text=analysed\_texts[analysis], |
| 632 | categories=categories, |
| 633 | facts=text\_facts, |
| 634 | upload=True) |
| 635 | except Exception as e: |
| 636 | flash("Something went wrong, please try again") |
| 637 | return redirect(url\_for('profile', username=session['username'])) |
| 638 |  |
| 639 |  |
| 640 | @app.route('/share/<texttitle>', methods=["POST", "GET"]) |
| 641 | @login\_required |
| 642 | def share(texttitle): |
| 643 | try: |
| 644 | """ |
| 645 | Loads the users who don't have a text with a specific name |
| 646 | """ |
| 647 | with Database() as database: |
| 648 | users = database.getTextUsers(texttitle) |
| 649 | if len(users) == 0: |
| 650 | users.append(False) |
| 651 | return render\_template('share.html', |
| 652 | title=texttitle, |
| 653 | users=users) |
| 654 | except Exception as e: |
| 655 | flash("Something went wrong, please try again") |
| 656 | return redirect(request.referrer) |
| 657 |  |
| 658 |  |
| 659 | @app.route('/download/<texttitle>', methods=["POST", "GET"]) |
| 660 | @login\_required |
| 661 | def download(texttitle): |
| 662 | """ |
| 663 | Creates a pdf with the raw text in from a html template |
| 664 | """ |
| 665 | try: |
| 666 | body = current\_file.analysed\_texts['Regular'] |
| 667 | rendered = render\_template('pdf\_template.html', title=texttitle, body=body) |
| 668 | options = {'encoding': "UTF-8"} |
| 669 | pdf = pdfkit.from\_string(rendered, False, options=options) |
| 670 | response = make\_response(pdf) |
| 671 | response.headers["Content-Type"] = 'application/pdf' |
| 672 | response.headers["Content-Disposition"] = 'attachment; filename=output.pdf' |
| 673 |  |
| 674 | return response |
| 675 | except Exception as e: |
| 676 | flash("Something went wrong, please try again") |
| 677 | return redirect(request.referrer) |
| 678 |  |
| 679 |  |
| 680 | @app.route('/share\_text/<texttitle>/<username>', methods=["POST", "GET"]) |
| 681 | @login\_required |
| 682 | def share\_text(texttitle, username): |
| 683 | """ |
| 684 | Shares the text with the user, by creating link in databse |
| 685 | """ |
| 686 | message = session['username'] + \ |
| 687 | " shared the text " + texttitle + " with you." |
| 688 | with Database() as database: |
| 689 | database.share\_text(texttitle, username, session["username"]) |
| 690 | database.sendNotif(username, message) |
| 691 | flash("Text Shared") |
| 692 | return redirect(url\_for('index')) |
| 693 |  |
| 694 |  |
| 695 | @app.route('/textdisplay/<textTitle>/<analysis>') |
| 696 | @login\_required |
| 697 | def textdisplay(textTitle, analysis): |
| 698 | """ |
| 699 | Displays the text on upload and allows the analysis to be selected |
| 700 | """ |
| 701 | try: |
| 702 | global current\_file |
| 703 | with Database() as database: |
| 704 | text\_owner = database.getTextOwner(textTitle, session['username']) |
| 705 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + text\_owner |
| 706 | path = app.config['UPLOAD\_FOLDER'] + '/objects/' + textTitle + '.txt' |
| 707 | with open(path, 'rb') as f: |
| 708 | current\_file = pickle.load(f) |
| 709 | analysed\_texts = current\_file.analysed\_texts |
| 710 | text\_facts = current\_file.stats |
| 711 | keywords = '' |
| 712 | for word in text\_facts['Key Words']: |
| 713 | keywords += word[0] + ", " |
| 714 | keywords = keywords[:-2] |
| 715 | return render\_template('textdisplay.html', |
| 716 | title=current\_file.title, |
| 717 | texts=analysed\_texts, |
| 718 | text=analysed\_texts[analysis], |
| 719 | facts=text\_facts, |
| 720 | keywords=keywords, |
| 721 | owner=text\_owner, |
| 722 | user=session['username']) |
| 723 | except Exception as e: |
| 724 | flash("Something went wrong, please try again") |
| 725 | return redirect(url\_for('profile', username=session['username'])) |
| 726 |  |
| 727 |  |
| 728 | # Deletes the text from the database and from the upload folder |
| 729 | @app.route('/deletetext/<texttitle>') |
| 730 | @login\_required |
| 731 | def deletetext(texttitle): |
| 732 | """ |
| 733 | Deletes text file and wipes any records from the database |
| 734 | """ |
| 735 | try: |
| 736 | with Database() as database: |
| 737 | canDelete = database.checkDelete(texttitle, session['id']) |
| 738 | if canDelete: |
| 739 | if app.config['UPLOAD\_FOLDER'] == UPLOAD\_FOLDER: |
| 740 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 741 | session['username'] |
| 742 | TextDelete(texttitle) |
| 743 | flash("File Deleted") |
| 744 | return redirect(url\_for('profile', username=session['username'])) |
| 745 | else: |
| 746 | flash("You do not have permission to delete this file") |
| 747 | return redirect(url\_for('profile', username=session['username'])) |
| 748 | except Exception as e: |
| 749 | flash("Oops, something went wrong... Try again.") |
| 750 | return redirect(url\_for('profile', username=session['username'])) |
| 751 |  |
| 752 |  |
| 753 | @app.route('/savetext/', methods=["POST", "GET"]) |
| 754 | @login\_required |
| 755 | def save\_text(): |
| 756 | """ |
| 757 | Saves the text object in a text file and saves it in db |
| 758 | """ |
| 759 | try: |
| 760 | global current\_file |
| 761 | if request.method == "POST": |
| 762 | current\_file.title = request.form['title'].replace(' ', '') |
| 763 | with Database() as database: |
| 764 | category = database.getCategory(request.form['Category']) |
| 765 | current\_file.category = category |
| 766 | session['id'] = database.getID(session['username']) |
| 767 | owned, shared = database.getUsersTexts(session['id']) |
| 768 | result = [x[0] for x in owned] + [x[0] for x in shared] |
| 769 | # Checks that the user does not already have |
| 770 | # access to a text with the same name |
| 771 | if current\_file.title not in result and current\_file.title != "": |
| 772 | object\_file\_path = app.config[ |
| 773 | 'UPLOAD\_FOLDER'] + '/objects/' + current\_file.title + '.txt' |
| 774 | # Puts the object in the file |
| 775 | pickle.dump(current\_file, open(object\_file\_path, 'wb')) |
| 776 | fhc = current\_file.text.content[:97] + '...' |
| 777 | data = [session['id'], current\_file.title, fhc, |
| 778 | current\_file.category] + current\_file.has\_features |
| 779 | keywords = current\_file.stats['Key Words'] |
| 780 | # Saves to database |
| 781 | database.addText(data, keywords) |
| 782 | # Adds keywords to graph |
| 783 | with open("word\_graph.txt", "rb") as f: |
| 784 | G = pickle.load(f) |
| 785 | for keyword in keywords: |
| 786 | G.add\_node(keyword[0]) |
| 787 | for k in keywords: |
| 788 | if k[0] != keyword[0]: |
| 789 | G.add\_edge(keyword[0], k[0]) |
| 790 | # Saves graph in file again |
| 791 | with open("word\_graph.txt", "wb") as f: |
| 792 | pickle.dump(G, f) |
| 793 | current\_file = None |
| 794 | return redirect(url\_for('profile', username=session['username'])) |
| 795 |  |
| 796 | else: |
| 797 | print("LONEOWDBOFHNEROSFOEBFWEBFWOD") |
| 798 | flash("A file with this name already exists.") |
| 799 | categories = database.loadCategories() |
| 800 | analysed\_texts = current\_file.analysed\_texts |
| 801 | text\_facts = current\_file.stats |
| 802 | keywords = '' |
| 803 | for word in text\_facts['Key Words']: |
| 804 | keywords += word[0] + ", " |
| 805 | keywords = keywords[:-2] |
| 806 | return render\_template('textdisplay.html', |
| 807 | title=current\_file.title, |
| 808 | texts=analysed\_texts, |
| 809 | text=analysed\_texts['Regular'], |
| 810 | facts=text\_facts, |
| 811 | keywords=keywords, |
| 812 | categories=categories, |
| 813 | ext=current\_file.text.ext, |
| 814 | upload=True) |
| 815 | else: |
| 816 | flash("Page does not exist") |
| 817 | return redirect(url\_for('index')) |
| 818 |  |
| 819 | except Exception as e: |
| 820 | flash("Something went wrong, please try again") |
| 821 | return redirect(url\_for('index')) |
| 822 |  |
| 823 |  |
| 824 | @app.route("/webtext/<analysis>", methods=["POST", "GET"]) |
| 825 | @login\_required |
| 826 | def webtext(analysis): |
| 827 | """ |
| 828 | Gets a text from the internet using an API |
| 829 | """ |
| 830 | global current\_file |
| 831 | try: |
| 832 | if request.form["url"] == "": |
| 833 | flash("No URL given") |
| 834 | return redirect(url\_for('index')) |
| 835 | url = request.form['url'] |
| 836 | current\_file = main.Analyser(url) |
| 837 | analysed\_texts = current\_file.analysed\_texts |
| 838 | text\_facts = current\_file.stats |
| 839 | with Database() as database: |
| 840 | categories = database.loadCategories() |
| 841 | keywords = '' |
| 842 | for word in text\_facts['Key Words']: |
| 843 | keywords += word[0] + ", " |
| 844 | keywords = keywords[:-2] |
| 845 | return render\_template('textdisplay.html', |
| 846 | title=current\_file.title, |
| 847 | texts=analysed\_texts, |
| 848 | text=analysed\_texts[analysis], |
| 849 | ext=current\_file.text.ext, |
| 850 | keywords=keywords, |
| 851 | categories=categories, |
| 852 | facts=text\_facts, |
| 853 | upload=True) |
| 854 |  |
| 855 | except: |
| 856 | flash("Web address not found!") |
| 857 | return redirect(url\_for('index')) |
| 858 |  |
| 859 |  |
| 860 | @app.route('/deleteaccount/') |
| 861 | @login\_required |
| 862 | def deleteaccount(): |
| 863 | """ |
| 864 | Deletes the users account, their files and texts from the database |
| 865 | """ |
| 866 | try: |
| 867 | if app.config['UPLOAD\_FOLDER'] == UPLOAD\_FOLDER: |
| 868 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 869 | session['username'] |
| 870 | with Database() as db: |
| 871 | texts = db.getOwnedTexts(session['id']) |
| 872 | for text in texts: |
| 873 | TextDelete(text[0]) |
| 874 | db.deleteUser(session['id']) |
| 875 | shutil.rmtree(app.config['UPLOAD\_FOLDER']) |
| 876 | session.clear() |
| 877 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER |
| 878 | flash("Account has been deleted") |
| 879 | return redirect(url\_for('index')) |
| 880 | except Exception as e: |
| 881 | flash("Something went wrong, please try again") |
| 882 | return redirect(url\_for('index')) |
| 883 |  |
| 884 |  |
| 885 | @app.route('/logout/') |
| 886 | @login\_required |
| 887 | def logout(): |
| 888 | """ |
| 889 | Logs the user out |
| 890 | """ |
| 891 | try: |
| 892 | session.clear() |
| 893 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER |
| 894 | flash("You have been logged out") |
| 895 | return redirect(url\_for('index')) |
| 896 | except Exception as e: |
| 897 | flash("Oops, something went wrong... Try again.") |
| 898 | return render\_template('index.html') |
| 899 |  |
| 900 |  |
| 901 | @app.route('/login/', methods=["GET", "POST"]) |
| 902 | def login\_page(): |
| 903 | """ |
| 904 | Allows the user to login |
| 905 | """ |
| 906 | try: |
| 907 | if request.method == "POST": |
| 908 | with Database() as database: |
| 909 | db\_password = database.checkPass(request.form['username']) |
| 910 | if len(db\_password) > 0: |
| 911 | db\_password = db\_password[0][0] |
| 912 | if pbkdf2\_sha256.verify(request.form['password'], db\_password): |
| 913 | session['logged\_in'] = True |
| 914 | session['id'] = database.getID(request.form['username']) |
| 915 | session['username'] = request.form['username'] |
| 916 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 917 | session['username'] |
| 918 | return redirect(url\_for('index')) |
| 919 | else: |
| 920 | flash("Invalid credentials, try again!") |
| 921 | return render\_template("login.html") |
| 922 | else: |
| 923 | flash("Invalid credentials, try again!") |
| 924 | return render\_template("login.html") |
| 925 | return render\_template("login.html") |
| 926 |  |
| 927 | except Exception as e: |
| 928 | flash("Something went wrong, please try again") |
| 929 | return render\_template("login.html") |
| 930 |  |
| 931 |  |
| 932 | @app.route('/register/', methods=["GET", "POST"]) |
| 933 | def register\_page(): |
| 934 | """ |
| 935 | Allows the user to register |
| 936 | """ |
| 937 | try: |
| 938 | if request.method == "POST": |
| 939 | form = reg\_form(request.form) |
| 940 | validation = form.validate() |
| 941 | if validation == "Passed": |
| 942 | # Hashes the passwords using sha256 |
| 943 | password\_hash = pbkdf2\_sha256.encrypt(form.password.data, rounds=200000, salt\_size=16) |
| 944 | data = [form.username.data, |
| 945 | form.email.data, password\_hash] |
| 946 | with Database() as database: |
| 947 | database.addUser(data) |
| 948 | session['logged\_in'] = True |
| 949 | session['username'] = form.username.data |
| 950 | session['id'] = database.getID(session['username']) |
| 951 | app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER + \ |
| 952 | session['username'] |
| 953 |  |
| 954 | # Makes the users directory on the server |
| 955 | os.mkdir(app.config['UPLOAD\_FOLDER']) |
| 956 | filedir = app.config['UPLOAD\_FOLDER'] + '/files' |
| 957 | objdir = app.config['UPLOAD\_FOLDER'] + '/objects' |
| 958 | os.mkdir(filedir) |
| 959 | os.mkdir(objdir) |
| 960 |  |
| 961 | flash("You have been logged in") |
| 962 | return redirect(url\_for('index')) |
| 963 | else: |
| 964 | flash(validation) |
| 965 | return render\_template('register.html') |
| 966 |  |
| 967 | return render\_template('register.html') |
| 968 |  |
| 969 | except Exception as e: |
| 970 | flash("Something went wrong, please try again") |
| 971 | return render\_template('register.html') |
| 972 |  |
| 973 |  |
| 974 | if \_\_name\_\_ == '\_\_main\_\_': |
| 975 | # Super Secret shhhh... |
| 976 | app.secret\_key = 'howmuchwoodwouldawoodchuckchuckifawoodchuckcouldchuckwood?' |
| 977 | app.config['SESSION\_TYPE'] = 'filesystem' |
| 978 | app.run() |

/static/db/dbclass.py

|  |  |
| --- | --- |
| 1 | import sqlite3 |
| 2 | FILE = "static/db/user\_text.sqlite" |
| 3 |  |
| 4 |  |
| 5 | class Database: |
| 6 |  |
| 7 | def \_\_init\_\_(self): |
| 8 | self.connection = sqlite3.connect(FILE) |
| 9 | self.cursor = self.connection.cursor() |
| 10 |  |
| 11 | def \_\_enter\_\_(self): |
| 12 | return self |
| 13 |  |
| 14 | def \_\_exit\_\_(self, exc\_type, exc\_value, traceback): |
| 15 | self.connection.close() |
| 16 |  |
| 17 | def addUser(self, data): |
| 18 | insert\_query = """INSERT INTO User (UserName, Email, Password) |
| 19 | VALUES (?,?,?);""" |
| 20 | self.cursor.execute(insert\_query, data) |
| 21 | self.connection.commit() |
| 22 |  |
| 23 | def getID(self, username): |
| 24 | id\_query = """SELECT UID |
| 25 | FROM User |
| 26 | WHERE UserName = ?;""" |
| 27 | user\_id = self.cursor.execute(id\_query, [username]).fetchall()[0][0] |
| 28 | return user\_id |
| 29 |  |
| 30 | def getTextID(self, title, user\_id): |
| 31 | id\_query = """SELECT TID |
| 32 | FROM Text |
| 33 | WHERE TextTitle = ? |
| 34 | AND UID = ?;""" |
| 35 | return self.cursor.execute(id\_query, [title, user\_id]).fetchall()[0][0] |
| 36 |  |
| 37 | def getAllKeywords(self): |
| 38 | query = """SELECT Word |
| 39 | FROM Keywords;""" |
| 40 | return [x[0] for x in self.cursor.execute(query)] |
| 41 |  |
| 42 | def addKeyword(self, word): |
| 43 | query = """INSERT INTO Keywords(Word) |
| 44 | VALUES(?)""" |
| 45 | self.cursor.execute(query, [word]) |
| 46 | self.connection.commit() |
| 47 |  |
| 48 | def getKeywordID(self, word): |
| 49 | query = """SELECT WID |
| 50 | FROM Keywords |
| 51 | WHERE Word = ?;""" |
| 52 | return self.cursor.execute(query, [word]).fetchall()[0][0] |
| 53 |  |
| 54 | def addToTextWords(self, tid, wid, score): |
| 55 | query = """INSERT INTO text\_word(TID, WID, Score) |
| 56 | VALUES(?, ?, ?);""" |
| 57 | self.cursor.execute(query, [tid, wid, score]) |
| 58 | self.connection.commit() |
| 59 |  |
| 60 | def addKeyWords(self, tid, keywords): |
| 61 | current\_keywords = self.getAllKeywords() |
| 62 | for keyword in keywords: |
| 63 | word = keyword[0] |
| 64 | if word not in current\_keywords: |
| 65 | self.addKeyword(word) |
| 66 | wid = self.getKeywordID(word) |
| 67 | self.addToTextWords(tid, wid, keyword[1]) |
| 68 |  |
| 69 | def addText(self, data, keywords): |
| 70 | query = """INSERT INTO Text(UID, TextTitle, FHC, CID, ReadingAge, |
| 71 | Sentiment, Alliteration, Antithesis, Juxtaposition) |
| 72 | VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)""" |
| 73 | self.cursor.execute(query, data) |
| 74 | self.connection.commit() |
| 75 | tid = self.getTextID(data[1], data[0]) |
| 76 | self.addKeyWords(tid, keywords) |
| 77 |  |
| 78 | def clearShares(self, text\_id): |
| 79 | query = """DELETE FROM User\_Text |
| 80 | WHERE TID = ?;""" |
| 81 | self.cursor.execute(query, [text\_id]) |
| 82 | self.connection.commit() |
| 83 |  |
| 84 | def clearKeywords(self, text\_id): |
| 85 | query = """DELETE FROM text\_word |
| 86 | WHERE TID = ?;""" |
| 87 | self.cursor.execute(query, [text\_id]) |
| 88 | self.connection.commit() |
| 89 |  |
| 90 | def deleteText(self, title, user\_id): |
| 91 | text\_id = self.getTextID(title, user\_id) |
| 92 | self.clearShares(text\_id) |
| 93 | self.clearKeywords(text\_id) |
| 94 | query = """DELETE FROM Text |
| 95 | WHERE TID = ?;""" |
| 96 | self.cursor.execute(query, [text\_id]) |
| 97 | self.connection.commit() |
| 98 |  |
| 99 | def getOwner(self, title, user\_id): |
| 100 | query = """SELECT User.UserName |
| 101 | FROM User, Text, User\_Text |
| 102 | WHERE Text.UID = User.UID |
| 103 | AND TextTitle = ? |
| 104 | AND User\_Text.TID = Text.TID |
| 105 | AND User\_Text.UID = ?;""" |
| 106 | return self.cursor.execute(query, [title, user\_id]).fetchall()[0][0] |
| 107 |  |
| 108 | def getTextOwner(self, title, username): |
| 109 | user\_id = self.getID(username) |
| 110 | owned\_texts = [x[0] for x in self.getOwnedTexts(user\_id)] |
| 111 | if title in owned\_texts: |
| 112 | return username |
| 113 | owner = self.getOwner(title, user\_id) |
| 114 | return owner |
| 115 |  |
| 116 | def getOwnedTexts(self, user\_id, search\_string=""): |
| 117 | search\_string = "%" + search\_string + "%" |
| 118 | query = """SELECT TextTitle, FHC |
| 119 | FROM Text |
| 120 | WHERE UID = ? |
| 121 | AND TextTitle like ?;""" |
| 122 | return self.cursor.execute(query, [user\_id, search\_string]).fetchall() |
| 123 |  |
| 124 | def getSharedTexts(self, user\_id, search\_string=""): |
| 125 | search\_string = "%" + search\_string + "%" |
| 126 | query = """SELECT Text.TextTitle, Text.FHC |
| 127 | FROM Text, User\_Text |
| 128 | WHERE User\_Text.UID = ? |
| 129 | AND Text.TID = User\_Text.TID |
| 130 | AND Text.TextTitle like ?;""" |
| 131 | return self.cursor.execute(query, [user\_id, search\_string]).fetchall() |
| 132 |  |
| 133 | def getUsersTexts(self, user\_id, search\_string=""): |
| 134 | return self.getOwnedTexts(user\_id, search\_string), self.getSharedTexts(user\_id, search\_string) |
| 135 |  |
| 136 | def checkDelete(self, title, user\_id): |
| 137 | owned\_texts = [text[0] for text in self.getOwnedTexts(user\_id)] |
| 138 | if title in owned\_texts: |
| 139 | return True |
| 140 | return False |
| 141 |  |
| 142 | def updateUserPassword(self, data): |
| 143 | query = """UPDATE User |
| 144 | SET Password=? |
| 145 | WHERE UserName=? |
| 146 | AND Password=?;""" |
| 147 |  |
| 148 | self.cursor.execute(query, data) |
| 149 | self.connection.commit() |
| 150 |  |
| 151 | def checkForUser(self, username): |
| 152 | query = """SELECT UserName |
| 153 | FROM User |
| 154 | WHERE UserName = ?;""" |
| 155 | matched\_user = self.cursor.execute(query, [username]).fetchall()[0][0] |
| 156 | return matched\_user |
| 157 |  |
| 158 | def checkPass(self, username): |
| 159 | query = """SELECT Password |
| 160 | FROM User |
| 161 | WHERE UserName = ?;""" |
| 162 | return self.cursor.execute(query, [username]).fetchall() |
| 163 |  |
| 164 | def searchOwnedTexts(self, data, extra\_query): |
| 165 | query = """SELECT Text.TextTitle, Text.FHC |
| 166 | FROM Text |
| 167 | WHERE Text.UID = ? |
| 168 | AND Text.ReadingAge > ? |
| 169 | AND Text.Sentiment > ? |
| 170 | AND Text.Sentiment < ?""" |
| 171 | query += extra\_query |
| 172 | return self.cursor.execute(query, data).fetchall() |
| 173 |  |
| 174 | def searchSharedTexts(self, data, extra\_query): |
| 175 | query = """SELECT Text.TextTitle, Text.FHC |
| 176 | FROM Text, User\_Text |
| 177 | WHERE User\_Text.UID = ? |
| 178 | AND Text.TID = User\_Text.TID |
| 179 | AND Text.ReadingAge > ? |
| 180 | AND Text.Sentiment > ? |
| 181 | AND Text.Sentiment < ?""" |
| 182 | query += extra\_query |
| 183 | return self.cursor.execute(query, data).fetchall() |
| 184 |  |
| 185 | def searchTexts(self, data, extra\_query): |
| 186 | return self.searchOwnedTexts(data, extra\_query), self.searchSharedTexts(data, extra\_query) |
| 187 |  |
| 188 | def getUsers(self): |
| 189 | query = """SELECT UID, UserName |
| 190 | FROM User;""" |
| 191 | users = self.cursor.execute(query).fetchall() |
| 192 | return users |
| 193 |  |
| 194 | def share\_text(self, title, username, current\_user): |
| 195 | query = """INSERT INTO User\_Text(UID, TID) |
| 196 | VALUES (?, ?);""" |
| 197 | uid = self.getID(username) |
| 198 | tid = self.getTextID(title, self.getID(current\_user)) |
| 199 | self.cursor.execute(query, [uid, tid]) |
| 200 | self.connection.commit() |
| 201 |  |
| 202 | def getUsername(self, UID): |
| 203 | query = """SELECT UserName |
| 204 | FROM User |
| 205 | WHERE UID = ?;""" |
| 206 | username = self.cursor.execute(query, [UID]).fetchall()[0][0] |
| 207 | return username |
| 208 |  |
| 209 | def getTextUsers(self, title): |
| 210 | valid\_users = [] |
| 211 | user\_ids = [x[0] for x in self.getUsers()] |
| 212 | for uid in user\_ids: |
| 213 | owned\_texts = [x[0] for x in self.getOwnedTexts(uid)] |
| 214 | shared\_texts = [x[0] for x in self.getSharedTexts(uid)] |
| 215 | texts = owned\_texts + shared\_texts |
| 216 | if title not in texts: |
| 217 | valid\_users.append({'username': self.getUsername(uid)}) |
| 218 | return valid\_users |
| 219 |  |
| 220 | def sendNotif(self, username, text): |
| 221 | query = """INSERT INTO Notifications(UID, MESSAGE) |
| 222 | VALUES (?, ?);""" |
| 223 | uid = self.getID(username) |
| 224 | self.cursor.execute(query, [uid, text]) |
| 225 | self.connection.commit() |
| 226 |  |
| 227 | def getNotifs(self, username): |
| 228 | query = """SELECT MESSAGE |
| 229 | FROM Notifications |
| 230 | WHERE UID = ?;""" |
| 231 | uid = self.getID(username) |
| 232 | notifications = [x[0] for x in self.cursor.execute(query, [uid]).fetchall()] |
| 233 | return notifications |
| 234 |  |
| 235 | def loadCategories(self): |
| 236 | query = """SELECT Cat |
| 237 | FROM Categories""" |
| 238 | thing = [x[0] for x in self.cursor.execute(query)] |
| 239 | return thing |
| 240 |  |
| 241 | def getCatID(self, cat): |
| 242 | query = """SELECT CID |
| 243 | FROM Categories |
| 244 | WHERE Cat = ?;""" |
| 245 | return self.cursor.execute(query, [cat]).fetchall()[0][0] |
| 246 |  |
| 247 | def addNewCategory(self, cat): |
| 248 | query = """INSERT INTO Categories(Cat) |
| 249 | VALUES (?);""" |
| 250 | self.cursor.execute(query, [cat]) |
| 251 | self.connection.commit() |
| 252 |  |
| 253 | def getCategory(self, category): |
| 254 | cats = self.loadCategories() |
| 255 | if category not in cats: |
| 256 | self.addNewCategory(category) |
| 257 | return self.getCatID(category) |
| 258 |  |
| 259 | def searchOwnedCategories(self, category, user\_id): |
| 260 | query = """SELECT Text.TextTitle, Text.FHC |
| 261 | FROM Text, Categories |
| 262 | WHERE Categories.Cat = ? |
| 263 | AND Categories.CID = Text.CID |
| 264 | AND Text.UID = ?""" |
| 265 | return self.cursor.execute(query, [category, user\_id]).fetchall() |
| 266 |  |
| 267 | def searchSharedCategories(self, category, user\_id): |
| 268 | query = """SELECT Text.TextTitle, Text.FHC |
| 269 | FROM Text, Categories, User\_Text |
| 270 | WHERE User\_Text.UID = ? |
| 271 | AND User\_Text.TID = Text.TID |
| 272 | AND Text.CID = Categories.CID |
| 273 | AND Categories.Cat = ?;""" |
| 274 | return self.cursor.execute(query, [user\_id, category]).fetchall() |
| 275 |  |
| 276 | def searchCategories(self, category, user\_id): |
| 277 | return self.searchOwnedCategories(category, user\_id), self.searchSharedCategories(category, user\_id) |
| 278 |  |
| 279 | def searchOwnedKeywords(self, keyword, user\_id): |
| 280 | query = """SELECT Text.TextTitle, Text.FHC, text\_word.Score |
| 281 | FROM Text, Keywords, text\_word |
| 282 | WHERE Keywords.Word = ? |
| 283 | AND Keywords.WID = text\_word.WID |
| 284 | AND text\_word.TID = Text.TID |
| 285 | AND Text.UID = ?;""" |
| 286 | return self.cursor.execute(query, [keyword, user\_id]).fetchall() |
| 287 |  |
| 288 | def searchSharedKeywords(self, keyword, user\_id): |
| 289 | query = """SELECT Text.TextTitle, Text.FHC, text\_word.Score |
| 290 | FROM Text, Keywords, text\_word, User\_Text |
| 291 | WHERE Keywords.Word = ? |
| 292 | AND Keywords.WID = text\_word.WID |
| 293 | AND text\_word.TID = User\_Text.TID |
| 294 | AND User\_Text.TID = Text.TID |
| 295 | AND User\_Text.UID = ?;""" |
| 296 | return self.cursor.execute(query, [keyword, user\_id]).fetchall() |
| 297 |  |
| 298 | def searchKeyword(self, keyword, user\_id): |
| 299 | return self.searchOwnedKeywords(keyword, user\_id), self.searchSharedKeywords(keyword, user\_id) |
| 300 |  |
| 301 | def deleteNotifs(self, user\_id): |
| 302 | query = """DELETE FROM Notifications |
| 303 | WHERE UID = ?""" |
| 304 | self.cursor.execute(query, [user\_id]) |
| 305 | self.connection.commit() |
| 306 |  |
| 307 | def deleteShares(self, user\_id): |
| 308 | query = """DELETE FROM User\_Text |
| 309 | WHERE UID = ?""" |
| 310 | self.cursor.execute(query, [user\_id]) |
| 311 | self.connection.commit() |
| 312 |  |
| 313 | def deleteUser(self, user\_id): |
| 314 | self.deleteNotifs(user\_id) |
| 315 | self.deleteShares(user\_id) |
| 316 | query = """DELETE FROM User |
| 317 | WHERE UID = ?""" |
| 318 | self.cursor.execute(query, [user\_id]) |
| 319 | self.connection.commit() |

/static/db/dbstart.py

|  |  |
| --- | --- |
| 1 | import sqlite3 |
| 2 | FILE = "user\_text.sqlite" |
| 3 |  |
| 4 |  |
| 5 | class Create\_Database: |
| 6 |  |
| 7 | def \_\_init\_\_(self): |
| 8 | self.connection = sqlite3.connect(FILE) |
| 9 | self.cursor = self.connection.cursor() |
| 10 |  |
| 11 | def create(self): |
| 12 | self.users() |
| 13 | self.texts() |
| 14 | self.user\_text() |
| 15 | self.notifcations() |
| 16 | self.categories() |
| 17 | self.keywords() |
| 18 | self.word\_text() |
| 19 |  |
| 20 | def users(self): |
| 21 | self.cursor.execute("""CREATE TABLE User (UID INTEGER PRIMARY KEY AUTOINCREMENT, |
| 22 | UserName VARCHAR(30) NOT NULL, |
| 23 | Email VARCHAR(50), |
| 24 | Password VARCHAR(100) NOT NULL);""") |
| 25 |  |
| 26 | def texts(self): |
| 27 | self.cursor.execute("""CREATE TABLE Text (TID INTEGER PRIMARY KEY AUTOINCREMENT, |
| 28 | UID INTEGER, |
| 29 | TextTitle VARCHAR(100) NOT NULL, |
| 30 | FHC VARCHAR(100), |
| 31 | CID INTEGER, |
| 32 | ReadingAge REAL, |
| 33 | Sentiment REAL, |
| 34 | Alliteration INTEGER(1), |
| 35 | Antithesis INTEGER(1), |
| 36 | Juxtaposition INTEGER(1), |
| 37 | FOREIGN KEY(UID) REFERENCES User(UID) |
| 38 | FOREIGN KEY(CID) REFERENCES Categories(CID));""") |
| 39 |  |
| 40 | def user\_text(self): |
| 41 | self.cursor.execute("""CREATE TABLE User\_Text( |
| 42 | UID INTEGER, |
| 43 | TID INTEGER, |
| 44 | FOREIGN KEY(UID) REFERENCES User(UID), |
| 45 | FOREIGN KEY(TID) REFERENCES Text(TID), |
| 46 | PRIMARY KEY (UID, TID)); |
| 47 | """) |
| 48 |  |
| 49 | def categories(self): |
| 50 | self.cursor.execute("""CREATE TABLE Categories( |
| 51 | CID INTEGER PRIMARY KEY AUTOINCREMENT, |
| 52 | Cat VARCHAR(50));""") |
| 53 |  |
| 54 | def notifcations(self): |
| 55 | self.cursor.execute("""CREATE TABLE Notifications( |
| 56 | UID INTEGER, |
| 57 | MESSAGE VARCHAR(140), |
| 58 | FOREIGN KEY(UID) REFERENCES User(UID)); |
| 59 | """) |
| 60 |  |
| 61 | def keywords(self): |
| 62 | self.cursor.execute("""CREATE TABLE Keywords( |
| 63 | WID INTEGER PRIMARY KEY AUTOINCREMENT, |
| 64 | Word VARCHAR(50));""") |
| 65 |  |
| 66 | def word\_text(self): |
| 67 | self.cursor.execute("""CREATE TABLE text\_word( |
| 68 | TID INTEGER, |
| 69 | WID INTEGER, |
| 70 | Score INTEGER, |
| 71 | FOREIGN KEY(TID) REFERENCES Text(TID), |
| 72 | FOREIGN KEY(WID) REFERENCES Keywords(WID), |
| 73 | PRIMARY KEY (WID, TID));""") |
| 74 |  |
| 75 | db = Create\_Database() |
| 76 | db.create() |

/static/scripts/adt.py

|  |  |
| --- | --- |
| 1 | class DataPos: |
| 2 | """ |
| 3 | Class to act as a node in the stack |
| 4 |  |
| 5 | Properties: |
| 6 | data - holds the information about that node |
| 7 | position - holds the posititon of the data in the external structure |
| 8 | extra\_info - any extra data that needs to persist |
| 9 | """ |
| 10 | def \_\_init\_\_(self, data, position, extra\_info={}): |
| 11 | self.data = data |
| 12 | self.position = position |
| 13 | self.extra\_info = extra\_info |
| 14 |  |
| 15 |  |
| 16 | class word\_stack: |
| 17 | """ |
| 18 | Stack used when doing operation with the texts |
| 19 |  |
| 20 | Properties: |
| 21 | \_\_stack - list used to hold the nodes |
| 22 | \_\_top\_of\_stack - pointer to the most recent list element |
| 23 | \_\_size - maximum size of the stack |
| 24 |  |
| 25 | Methods: |
| 26 | is\_empty - returns True if stack is empty, False if not |
| 27 | \_\_is\_full - returns True if stack is full, False if not |
| 28 | get\_height - returns the current height of the stack |
| 29 | peek - returns the top item of the stack |
| 30 | pop - returns and removes the top item of the stack |
| 31 | add - inserts an item at the top of the stack |
| 32 | """ |
| 33 | def \_\_init\_\_(self, size=5): |
| 34 | self.\_\_stack = [None] \* size |
| 35 | self.\_\_top\_of\_stack = -1 |
| 36 | self.\_\_size = size |
| 37 |  |
| 38 | def is\_empty(self): |
| 39 | return self.\_\_top\_of\_stack == -1 |
| 40 |  |
| 41 | def \_\_is\_full(self): |
| 42 | return self.\_\_top\_of\_stack == self.\_\_size - 1 |
| 43 |  |
| 44 | def get\_height(self): |
| 45 | return self.\_\_top\_of\_stack + 1 |
| 46 |  |
| 47 | def peek(self): |
| 48 | if not self.is\_empty(): |
| 49 | return self.\_\_stack[self.\_\_top\_of\_stack] |
| 50 | return False |
| 51 |  |
| 52 | def pop(self): |
| 53 | if not self.is\_empty(): |
| 54 | top\_item = self.\_\_stack[self.\_\_top\_of\_stack] |
| 55 | self.\_\_stack[self.\_\_top\_of\_stack] = None |
| 56 | self.\_\_top\_of\_stack -= 1 |
| 57 | return top\_item |
| 58 | return False |
| 59 |  |
| 60 | def add(self, data): |
| 61 | if not self.\_\_is\_full(): |
| 62 | self.\_\_top\_of\_stack += 1 |
| 63 | self.\_\_stack[self.\_\_top\_of\_stack] = data |
| 64 |  |
| 65 |  |
| 66 | class circular\_queue: |
| 67 | """ |
| 68 | Circular queue to compare words |
| 69 | Size can be increased to compare words that are a further distance apart |
| 70 |  |
| 71 | Properties: |
| 72 | max\_size - maximum size of the queue |
| 73 | current\_size - current number of items in the queue |
| 74 | queue - list used to hold the data nodes |
| 75 | front\_pointer - start of the queue |
| 76 | end\_pointer - end of the queue |
| 77 |  |
| 78 | Methods: |
| 79 | is\_empty - returns True if queue is empty, False if not |
| 80 | \_\_is\_full - returns True if queue is full, False if not |
| 81 | peek\_at\_end - returns the last item of the queue |
| 82 | dequeue - removes and return the item at the start of the queue |
| 83 | enqueue - inserts an item at the end of the queue |
| 84 | """ |
| 85 | def \_\_init\_\_(self, size=2): |
| 86 | self.max\_size = size |
| 87 | self.current\_size = 0 |
| 88 | self.queue = [None] \* self.max\_size |
| 89 | self.front\_pointer = 0 |
| 90 | self.rear\_pointer = -1 |
| 91 |  |
| 92 | def is\_empty(self): |
| 93 | return self.current\_size == 0 |
| 94 |  |
| 95 | def \_\_is\_full(self): |
| 96 | return self.current\_size == self.max\_size |
| 97 |  |
| 98 | def peek\_at\_end(self): |
| 99 | if self.is\_empty(): |
| 100 | return False |
| 101 | item = self.queue[self.rear\_pointer] |
| 102 | return item |
| 103 |  |
| 104 | def enqueue(self, newItem): |
| 105 | if self.\_\_is\_full(): |
| 106 | return False |
| 107 | self.rear\_pointer = (self.rear\_pointer + 1) % self.max\_size |
| 108 | self.queue[self.rear\_pointer] = newItem |
| 109 | self.current\_size += 1 |
| 110 |  |
| 111 | def dequeue(self): |
| 112 | if self.is\_empty(): |
| 113 | return False |
| 114 | item = self.queue[self.front\_pointer] |
| 115 | self.front\_pointer = (self.front\_pointer + 1) % self.max\_size |
| 116 | self.current\_size -= 1 |
| 117 | return item |
| 118 |  |
| 119 |  |
| 120 | class sentiment\_queue(circular\_queue): |
| 121 | """ |
| 122 | Inherits from the circular queue and adds a method that compares the |
| 123 | sentiment of the first and last items in the queue |
| 124 |  |
| 125 | Methods: |
| 126 | check\_sentiments - Compares the sentiment of the last and first items in the queue |
| 127 | if the difference in sentiments is greater than 3, returns the items |
| 128 | if not then return False |
| 129 | """ |
| 130 |  |
| 131 | def \_\_init\_\_(self): |
| 132 | super().\_\_init\_\_() |
| 133 |  |
| 134 | def check\_sentiments(self): |
| 135 | if self.current\_size == 2: |
| 136 | end = self.peek\_at\_end() |
| 137 | start = self.dequeue() |
| 138 | end\_sent = end.extra\_info['sentiment'] |
| 139 | start\_sent = start.extra\_info['sentiment'] |
| 140 | if abs(end\_sent - start\_sent) > 3: |
| 141 | return start, end |
| 142 | return False, False |
| 143 | return False, False |

/static/scripts/form.py

|  |  |
| --- | --- |
| 1 | import sqlite3 |
| 2 | import re |
| 3 | DATABASE = '/home/dylan/Documents/TextAnalyser/static/db/user\_text.sqlite' |
| 4 |  |
| 5 |  |
| 6 | class Field: |
| 7 | """ |
| 8 | Class to store data about the fields in a form |
| 9 |  |
| 10 | Properties: |
| 11 | data - value of data stored |
| 12 | length - length of the data |
| 13 | validate - whether the field has been validated |
| 14 | """ |
| 15 | def \_\_init\_\_(self, value): |
| 16 | self.data = value |
| 17 | self.length = len(value) |
| 18 | self.validate = "Not" |
| 19 |  |
| 20 |  |
| 21 | class reg\_form: |
| 22 | """ |
| 23 | Handles data from the registration form |
| 24 |  |
| 25 | Properties: |
| 26 | dbconn - connection to the database |
| 27 | dbcur - database cursor |
| 28 | username - username entered |
| 29 | password - password entered |
| 30 | password\_confirm - re-entry of the password |
| 31 | email - email entered |
| 32 |  |
| 33 | Method: |
| 34 | validate - checks all the fields have been validated |
| 35 | email\_validate - checks that the email is of the required form and is not already used |
| 36 | username\_validate - checks that the username is not already used and is of a required length |
| 37 | pass\_validate - checks that the password matches the confirm password and that it meets target complexity |
| 38 | """ |
| 39 | def \_\_init\_\_(self, form): |
| 40 | self.dbconn = sqlite3.connect(DATABASE) |
| 41 | self.dbconn.row\_factory = lambda cursor, row: row[0] |
| 42 | self.dbcur = self.dbconn.cursor() |
| 43 | self.username = Field(form['username']) |
| 44 | self.email = Field(form['email']) |
| 45 | self.password = Field(form['password']) |
| 46 | self.password\_confirm = Field(form['confirm']) |
| 47 |  |
| 48 | def validate(self): |
| 49 | us\_val = self.username\_validate() |
| 50 | em\_val = self.email\_validate() |
| 51 | pass\_val = self.pass\_validate() |
| 52 | if self.username.validate == "Passed": |
| 53 | if self.email.validate == "Passed": |
| 54 | if self.password.validate == "Passed": |
| 55 | return "Passed" |
| 56 | else: |
| 57 | return pass\_val |
| 58 | else: |
| 59 | return em\_val |
| 60 | else: |
| 61 | return us\_val |
| 62 |  |
| 63 | def email\_validate(self): |
| 64 | query = """SELECT Email FROM User""" |
| 65 | all\_emails = self.dbcur.execute(query).fetchall() |
| 66 | email\_regex = r"(^[a-zA-Z0-9\_.]+@[a-zA-Z0-9]+**\.**[a-zA-z0-9]+$)" |
| 67 | regex = re.compile(email\_regex) |
| 68 | if self.email.data in all\_emails: |
| 69 | return "This email has already been used." |
| 70 | elif regex.match(self.email.data): |
| 71 | self.email.validate = "Passed" |
| 72 | else: |
| 73 | return "Not a valid email address." |
| 74 |  |
| 75 | def username\_validate(self): |
| 76 | query = "SELECT UserName FROM User" |
| 77 | all\_names = self.dbcur.execute(query).fetchall() |
| 78 | if self.username.data in all\_names: |
| 79 | return "Username is already taken." |
| 80 | elif self.username.length in range(4, 21): |
| 81 | self.username.validate = "Passed" |
| 82 | else: |
| 83 | return "Username is too long/short" |
| 84 |  |
| 85 | def pass\_validate(self): |
| 86 | password\_regex = r"^(?=.\*?\d)(?=.\*?[a-z])(?=.\*?[A-Z])[a-zA-Z\d]{6,}$" |
| 87 | regex = re.compile(password\_regex) |
| 88 | if self.password.data != self.password\_confirm.data: |
| 89 | return "Passwords do not match." |
| 90 | elif regex.match(self.password.data): |
| 91 | self.password.validate = "Passed" |
| 92 | else: |
| 93 | return "Password needs to have an upper and lowercase letter and a number." |

/static/script/main.py

|  |  |
| --- | --- |
| 1 | import nltk |
| 2 | import pickle |
| 3 | import requests |
| 4 | from static.scripts.sorts import quick\_sort, radix\_sort |
| 5 | import static.scripts.adt as adt |
| 6 | from nltk import sent\_tokenize, word\_tokenize |
| 7 | from nltk.corpus import stopwords |
| 8 | from math import log |
| 9 | from flask import Markup |
| 10 | from docx import Document |
| 11 | import string |
| 12 | import pygal |
| 13 | import json |
| 14 | import PyPDF2 |
| 15 |  |
| 16 | A = 0.4 |
| 17 | # Special cases of |
| 18 | special\_case = ['s', 'c', 't', 'p'] |
| 19 | phonics = [["f", "ph"], |
| 20 | ["n", "kn", "gn"], |
| 21 | ["s", "ce", "ci", "cy"], |
| 22 | ["r", "wr"], |
| 23 | ["u", "eu"]] |
| 24 |  |
| 25 | # REQUEST: |
| 26 | # http://api.diffbot.com/v3/article?url=http://www.independent.co.uk/news/science/scientists-albert-einstein-light-universe-wrong-joao-magueijo-imperial-college-london-a7443136.html&token=bc6b3926127b4341c672dd62bcfbdc62 |
| 27 |  |
| 28 | stops = set(stopwords.words('english')) |
| 29 | DIPHTHONG = ['ou', 'ie', 'ay', 'oi', 'oo', 'ea', 'ee', 'ai', 'ion'] |
| 30 | TRIPHTHONG = ['iou', 'eau'] |
| 31 | VOWELS = ['a', 'e', 'i', 'o', 'u', 'y'] |
| 32 | punc = ["!", "?", "``", '\*', '(', ')', '-', '{', '}', '[', ']', ':', |
| 33 | ';', "'", ',', '.', '\\', '/', "''", '"', "”", "“", "–", |
| 34 | "--", "—"] |
| 35 | contracted\_words = ["'s", "'m", "n't", "'re", "it’s", "i’ve", "they’re", 'we’re'] |
| 36 |  |
| 37 |  |
| 38 | def check\_allit(current\_word, top\_item): |
| 39 | """ |
| 40 | Checks if two words are alliterative |
| 41 | """ |
| 42 | current\_word\_first = current\_word.data[0].lower() |
| 43 | current\_word\_first\_two = current\_word.data[:2].lower() |
| 44 | top\_item\_first = top\_item.data[0].lower() |
| 45 | top\_item\_first\_two = top\_item.data[:2].lower() |
| 46 | # Checks for first letter the same |
| 47 | if current\_word\_first == top\_item\_first: |
| 48 | if current\_word\_first in special\_case: |
| 49 | # Check for h's |
| 50 | if current\_word\_first\_two[1] == "h" and top\_item\_first\_two[1] == "h": |
| 51 | return True |
| 52 | elif current\_word\_first\_two[1] != "h" and top\_item\_first\_two[1] != "h": |
| 53 | return True |
| 54 | else: |
| 55 | return True |
| 56 | else: |
| 57 | # Checks for similar phonics |
| 58 | for phonic in phonics: |
| 59 | if top\_item\_first in phonic or top\_item\_first\_two in phonic: |
| 60 | if current\_word\_first in phonic or current\_word\_first\_two in phonic: |
| 61 | return True |
| 62 | return False |
| 63 |  |
| 64 |  |
| 65 | def loadAfinn(): |
| 66 | """ |
| 67 | Loads the words from the library of afinn words |
| 68 | Adds each word to a dictionary with key as word, value as sentiment |
| 69 | """ |
| 70 | afinnwords = {} |
| 71 | with open('static/scripts/AFINN-111.txt', encoding="utf-8") as f: |
| 72 | for line in f: |
| 73 | split\_line = line.split() |
| 74 | joined = '' |
| 75 | for w in range(0, len(split\_line) - 1): |
| 76 | joined += split\_line[w] |
| 77 | afinnwords[joined] = int(split\_line[len(split\_line) - 1]) |
| 78 | return afinnwords |
| 79 |  |
| 80 |  |
| 81 | def check\_source(directory): |
| 82 | """ |
| 83 | Returns the file type |
| 84 | """ |
| 85 | eow = directory[-3:] |
| 86 | sow = directory[:3] |
| 87 | if eow == 'txt': |
| 88 | return eow |
| 89 | elif eow == 'pdf': |
| 90 | return eow |
| 91 | elif eow == "ocx": |
| 92 | return "docx" |
| 93 | elif sow == 'htt' or sow == 'www': |
| 94 | return 'url' |
| 95 | else: |
| 96 | return None |
| 97 |  |
| 98 |  |
| 99 | def check\_syl(word): |
| 100 | """ |
| 101 | Approximates the number of sylables in a word, based on the number of vowels |
| 102 | """ |
| 103 | count = 0 |
| 104 | # Count number of vowels |
| 105 | for letter in word: |
| 106 | if letter in VOWELS: |
| 107 | count += 1 |
| 108 | # Subtracts 1 for double vowel sounds |
| 109 | for di in DIPHTHONG: |
| 110 | if di in word: |
| 111 | count -= 1 |
| 112 | # Subtracts 2 for triple vowel sounds |
| 113 | for tri in TRIPHTHONG: |
| 114 | if tri in word: |
| 115 | count -= 2 |
| 116 | # Subtracts 1 if a silent e is at the end of the word |
| 117 | if count != 0 and word[len(word) - 1] == 'e': |
| 118 | count -= 1 |
| 119 | return count |
| 120 |  |
| 121 |  |
| 122 | class TF\_IDF(): |
| 123 | """ |
| 124 | Algorithm to calculate the significance of a word in a text |
| 125 |  |
| 126 | Properties: |
| 127 | document\_frequency - has a record for each text trained and holds the |
| 128 | number of occurences of each word in that text |
| 129 |  |
| 130 | Methods: |
| 131 | count\_words - given a text counts the instances of each word |
| 132 | train\_frequencies - adds a new text to the document\_frequency |
| 133 | normalise\_frequency - defines the frequency of a word in relation to the |
| 134 | frequencies of other words |
| 135 | find\_idf - finds the inverse document frequnecy of a word based on the |
| 136 | number of texts that word appears in |
| 137 | check\_new\_text - Calculates the tf-idf of a new text and sorts the list |
| 138 | to find the top 5 most significant words in that text |
| 139 | """ |
| 140 |  |
| 141 | def \_\_init\_\_(self): |
| 142 | self.number\_of\_documents = 0 |
| 143 | self.document\_frequency = {} |
| 144 |  |
| 145 | def count\_words(self, text): |
| 146 | words = word\_tokenize(text) |
| 147 | text\_words = {} |
| 148 | for word in words: |
| 149 | word = word.lower() |
| 150 | if word not in stops and word not in punc and word not in contracted\_words: |
| 151 | current\_words = list(text\_words.keys()) |
| 152 | if word in current\_words: |
| 153 | text\_words[word] += 1 |
| 154 | else: |
| 155 | text\_words[word] = 1 |
| 156 | return text\_words |
| 157 |  |
| 158 | def train\_frequencies(self, text): |
| 159 | text\_words = word\_tokenize(text) |
| 160 | unique\_words = set(text\_words) |
| 161 | for word in unique\_words: |
| 162 | try: |
| 163 | self.document\_frequency[word] += 1 |
| 164 | except: |
| 165 | self.document\_frequency[word] = 1 |
| 166 | self.number\_of\_documents += 1 |
| 167 |  |
| 168 | def normalise\_frequency(self, word\_count, highest\_raw): |
| 169 | # Formula to normalise the frequency |
| 170 | normalise\_frequency = A + ((1 - A) \* (word\_count / highest\_raw)) |
| 171 | return normalise\_frequency |
| 172 |  |
| 173 | def find\_idf(self, word): |
| 174 | try: |
| 175 | documents\_containing = self.document\_frequency[word] + 1 |
| 176 | except: |
| 177 | documents\_containing = 1 |
| 178 | idf = log((self.number\_of\_documents / (documents\_containing)), 10) |
| 179 | return idf |
| 180 |  |
| 181 | # returns the top five word in the new text |
| 182 | def check\_new\_text(self, text): |
| 183 | words = set(word\_tokenize(text)) |
| 184 | word\_count = self.count\_words(text) |
| 185 | highest\_raw = word\_count[max(word\_count, key=lambda i: word\_count[i])] |
| 186 | tf\_idfs = {} |
| 187 | for word in words: |
| 188 | word = word.lower() |
| 189 | if word not in stops and word not in punc and word not in contracted\_words: |
| 190 | ntf = self.normalise\_frequency(word\_count[word], highest\_raw) |
| 191 | idf = self.find\_idf(word) |
| 192 | tf\_idf = str(ntf \* idf) |
| 193 | # converts in to format that is sortable by radix Algorithm |
| 194 | if float(tf\_idf) >= 0: |
| 195 | tf\_idfs[word] = tf\_idf[0] + tf\_idf[2:5] |
| 196 | lst = [] |
| 197 | if len(tf\_idfs) == 0: |
| 198 | return [] |
| 199 | for word, ti in tf\_idfs.items(): |
| 200 | lst.append(tuple([word, ti.zfill(4)])) |
| 201 | print(lst) |
| 202 | print(radix\_sort(lst)) |
| 203 | # Gets 5 words with the highest tf-idf |
| 204 | self.train\_frequencies(text) |
| 205 | return radix\_sort(lst)[-5:] |
| 206 |  |
| 207 |  |
| 208 | class Analyser: |
| 209 | """ |
| 210 | Class that performs and stores the analyses as well as the statistics |
| 211 | about the text |
| 212 |  |
| 213 | Properties: |
| 214 | path - path to the text |
| 215 | title - text title, name of file or user defined title |
| 216 | category - category of the text |
| 217 | text - reader object that stores the text |
| 218 | tf - holds the trained tf-idf object from the file |
| 219 | has\_features - holds boolean values as to whether a text contains a |
| 220 | certain type of analysis |
| 221 | stats - dictionary holding the statistics about the text |
| 222 | analysed\_texts - dictionary holding the analysed version of the text |
| 223 |  |
| 224 | Methods: |
| 225 | open\_tf - opens and returns the trained tf-idf object |
| 226 | get\_facts - calls all the functions that return text facts and stores them |
| 227 | get\_keywords - gets the keywords using the check\_new\_text in tf-idf class |
| 228 | number\_of\_words - counts the number of words in the text |
| 229 | \_\_get\_words - counts the number of occurences of each word in the text |
| 230 | sorted\_words - sorts the words from \_\_get\_words and takes the top ten |
| 231 | and creates bar graph |
| 232 | \_\_get\_punctuation - counts the number of occurences of each type of |
| 233 | punctuation in the text |
| 234 | sorted\_words - sorts the results from \_\_get\_punctuation and creates bar graph |
| 235 | total\_syl - calculates the total number of sylables |
| 236 | find\_reading\_age - calculates the reading age using the flesch-kincaid formula |
| 237 | get\_sentiment - finds the average sentiment of all words in the text |
| 238 | analyse\_all - calls all the functions that return new analyses and stores them |
| 239 | find\_allit - finds all the instances of alliteration and marks them |
| 240 | antithesis - finds all the instances of antithesis and marks them |
| 241 | juxtaposition - finds all the instances of juxtaposition and marks them |
| 242 | """ |
| 243 | def \_\_init\_\_(self, directory, title=''): |
| 244 | self.path = directory |
| 245 | self.title = title |
| 246 | self.category = '' |
| 247 | file\_type = check\_source(directory) |
| 248 | if file\_type == 'txt': |
| 249 | self.text = TextFile(directory) |
| 250 | elif file\_type == 'url': |
| 251 | self.text = WebPage(directory) |
| 252 | self.title = self.text.title |
| 253 | elif file\_type == 'pdf': |
| 254 | self.text = PDFFile(directory) |
| 255 | elif file\_type == "docx": |
| 256 | self.text = DOCXFile(directory) |
| 257 | self.tf = self.open\_tf() |
| 258 | self.has\_features = [] |
| 259 | self.stats = self.get\_facts() |
| 260 | self.analysed\_texts = self.analyse\_all() |
| 261 |  |
| 262 | def open\_tf(self): |
| 263 | with open("static/scripts/tf.txt", "rb") as f: |
| 264 | tf = pickle.load(f) |
| 265 | return tf |
| 266 |  |
| 267 | def get\_facts(self): |
| 268 | facts = {} |
| 269 | facts['Number of Words'] = self.number\_of\_words() |
| 270 | facts['Reading Age'] = round(self.find\_reading\_age(), 2) |
| 271 | facts['Punctuation'] = self.sorted\_punctuation() |
| 272 | facts['Words'] = self.sorted\_words() |
| 273 | facts['Sentiment'] = round(self.get\_sentiment(), 2) |
| 274 | facts['Key Words'] = self.get\_keywords() |
| 275 | self.has\_features.append(facts['Reading Age']) |
| 276 | self.has\_features.append(facts['Sentiment']) |
| 277 | return facts |
| 278 |  |
| 279 | def get\_keywords(self): |
| 280 | text = self.text.content |
| 281 | keywords = self.tf.check\_new\_text(text) |
| 282 | return keywords |
| 283 |  |
| 284 | def number\_of\_words(self): |
| 285 | total = 0 |
| 286 | for word in self.text.word\_tok(): |
| 287 | if word not in punc or word not in contracted\_words: |
| 288 | total += 1 |
| 289 | return total |
| 290 |  |
| 291 | def analyse\_all(self): |
| 292 | analysed\_texts = {} |
| 293 | analysed\_texts['Regular'] = self.text.content |
| 294 | analysed\_texts['Alliteration'], has\_allit = self.find\_allit() |
| 295 | analysed\_texts['Antithesis'], has\_anti = self.antithesis() |
| 296 | analysed\_texts['Juxtaposition'], has\_juxta = self.juxtaposition() |
| 297 | self.has\_features.append(has\_allit) |
| 298 | self.has\_features.append(has\_anti) |
| 299 | self.has\_features.append(has\_juxta) |
| 300 | return analysed\_texts |
| 301 |  |
| 302 | def sorted\_punctuation(self): |
| 303 | tally = self.\_\_get\_punctuation() |
| 304 | if len(tally) > 0: |
| 305 | sorted\_tally = quick\_sort(tally) |
| 306 | for x in range(len(sorted\_tally)): |
| 307 | sorted\_tally[x] = (sorted\_tally[x][0], int(sorted\_tally[x][1])) |
| 308 | punc\_chart = pygal.HorizontalBar() |
| 309 | punc\_chart.title = 'Most Used Punctuation in Text' |
| 310 | for punc in reversed(sorted\_tally): |
| 311 | punc\_chart.add(punc[0], punc[1]) |
| 312 | else: |
| 313 | punc\_chart = pygal.HorizontalBar() |
| 314 | punc\_chart.title = 'Most Used Punctuation in Text' |
| 315 | return punc\_chart.render\_data\_uri() |
| 316 |  |
| 317 | def \_\_get\_punctuation(self): |
| 318 | words = self.text.word\_tok() |
| 319 | text\_punc = [w for w in words if w in punc] |
| 320 | punc\_tally = list(tuple((p, str(text\_punc.count(p)).zfill(5)) |
| 321 | for p in punc if text\_punc.count(p) > 0)) |
| 322 | return punc\_tally |
| 323 |  |
| 324 | def sorted\_words(self): |
| 325 | tally = self.\_\_get\_words() |
| 326 | if len(tally) != 0: |
| 327 | sorted\_tally = radix\_sort(tally) |
| 328 | for x in range(len(sorted\_tally)): |
| 329 | sorted\_tally[x] = (sorted\_tally[x][0], int(sorted\_tally[x][1])) |
| 330 | top\_ten = sorted\_tally[-10:] |
| 331 | word\_chart = pygal.HorizontalBar() |
| 332 | word\_chart.title = 'Most Used Words in Text' |
| 333 | for word in reversed(top\_ten): |
| 334 | word\_chart.add(word[0], word[1]) |
| 335 | else: |
| 336 | word\_chart = pygal.HorizontalBar() |
| 337 | word\_chart.title = 'Most Used Words in Text' |
| 338 | return word\_chart.render\_data\_uri() |
| 339 |  |
| 340 | def \_\_get\_words(self): |
| 341 | words = self.text.no\_stops\_words() |
| 342 | words\_used = [] |
| 343 | for w in words: |
| 344 | if w not in words\_used and w not in punc and w not in contracted\_words: |
| 345 | words\_used.append(w) |
| 346 | word\_tally = list(tuple((w, str(words.count(w)).zfill(5)) |
| 347 | for w in words\_used if words.count(w) > 0)) |
| 348 | return word\_tally |
| 349 |  |
| 350 | def find\_reading\_age(self): |
| 351 | words = self.text.word\_tok() |
| 352 | words = [w for w in words if ( |
| 353 | w not in punc and w not in contracted\_words)] |
| 354 | words = len(words) |
| 355 | sents = len(self.text.sent\_tok()) |
| 356 | syls = self.total\_syl() |
| 357 | grade\_flesch = (0.39 \* (words / (sents + 1))) + (11.8 \* (syls / (words + 1))) - 10 |
| 358 | flesch = 206.835 - (1.015 \* (words / (sents + 1))) - (84.6 \* (syls / (words + 1))) |
| 359 | if grade\_flesch < 0: |
| 360 | return 0 |
| 361 | return grade\_flesch |
| 362 |  |
| 363 | def total\_syl(self): |
| 364 | counts = [] |
| 365 | words = self.text.word\_tok() |
| 366 | for word in words: |
| 367 | counts.append(check\_syl(word)) |
| 368 | return sum(counts) |
| 369 |  |
| 370 | def find\_allit(self): |
| 371 | has = 0 |
| 372 | text = self.text.word\_tok() |
| 373 | non\_words = punc + contracted\_words |
| 374 | text\_stack = adt.word\_stack() |
| 375 | for index, word in enumerate(text): |
| 376 | if word not in non\_words and word not in stops: |
| 377 | current\_word = adt.DataPos(word, index) |
| 378 | top\_item = text\_stack.peek() |
| 379 | if top\_item is False: |
| 380 | text\_stack.add(current\_word) |
| 381 | elif check\_allit(current\_word, top\_item): |
| 382 | text\_stack.add(current\_word) |
| 383 | else: |
| 384 | if text\_stack.get\_height() > 1: |
| 385 | last\_item = text\_stack.pop() |
| 386 | while not text\_stack.is\_empty(): |
| 387 | first\_item = text\_stack.pop() |
| 388 | first\_item\_index = first\_item.position |
| 389 | last\_item\_index = last\_item.position |
| 390 | text[first\_item\_index] = "<mark>" + \ |
| 391 | text[first\_item\_index] |
| 392 | text[last\_item\_index] = text[ |
| 393 | last\_item\_index] + "</mark>" |
| 394 | has = 1 |
| 395 | while not text\_stack.is\_empty(): |
| 396 | text\_stack.pop() |
| 397 | text\_stack.add(current\_word) |
| 398 | if index == len(text) - 1: |
| 399 | if text\_stack.get\_height() > 1: |
| 400 | last\_item = text\_stack.pop() |
| 401 | while not text\_stack.is\_empty(): |
| 402 | first\_item = text\_stack.pop() |
| 403 | first\_item\_index = first\_item.position |
| 404 | last\_item\_index = last\_item.position |
| 405 | text[first\_item\_index] = "<mark>" + \ |
| 406 | text[first\_item\_index] |
| 407 | text[last\_item\_index] = text[ |
| 408 | last\_item\_index] + "</mark>" |
| 409 | has = 1 |
| 410 | while not text\_stack.is\_empty(): |
| 411 | text\_stack.pop() |
| 412 | text\_stack.add(current\_word) |
| 413 |  |
| 414 | result = '' |
| 415 | for w in text: |
| 416 | if w in punc or w in contracted\_words: |
| 417 | result += w |
| 418 | else: |
| 419 | result += " " + w |
| 420 |  |
| 421 | return Markup(result), has |
| 422 |  |
| 423 | def get\_sentiment(self): |
| 424 | afinnwords = loadAfinn() |
| 425 | sentiment = 0 |
| 426 | no = 0 |
| 427 | words = self.text.word\_tok() |
| 428 | for w in words: |
| 429 | w = w.lower() |
| 430 | try: |
| 431 | sentiment += afinnwords[w] |
| 432 | no += 1 |
| 433 | except: |
| 434 | pass |
| 435 | if no > 0: |
| 436 | sentiment = sentiment / no |
| 437 | return sentiment |
| 438 |  |
| 439 | def antithesis(self): |
| 440 | has = 0 |
| 441 | afinnwords = loadAfinn() |
| 442 | non\_words = punc + contracted\_words |
| 443 | words = self.text.word\_tok() |
| 444 | comparison\_stack = adt.sentiment\_queue() |
| 445 | for index, word in enumerate(words): |
| 446 | word = word.lower() |
| 447 | if word not in non\_words and word not in stops: |
| 448 | try: |
| 449 | word\_info = {"sentiment": afinnwords[word]} |
| 450 | except: |
| 451 | word\_info = {"sentiment": 0} |
| 452 | new\_word = adt.DataPos(word, index, word\_info) |
| 453 | comparison\_stack.enqueue(new\_word) |
| 454 | first, second = comparison\_stack.check\_sentiments() |
| 455 | if first is not False: |
| 456 | words[first.position] = "<mark>" + words[first.position] |
| 457 | words[second.position] = words[second.position] + "</mark>" |
| 458 | has = 1 |
| 459 | result = '' |
| 460 | for w in words: |
| 461 | if w in punc or w in contracted\_words: |
| 462 | result += w |
| 463 | else: |
| 464 | result += " " + w |
| 465 |  |
| 466 | return Markup(result), has |
| 467 |  |
| 468 | def juxtaposition(self): |
| 469 | has = 0 |
| 470 | afinnwords = loadAfinn() |
| 471 | sentences = self.text.sent\_tok() |
| 472 | sentiments = [] |
| 473 | output\_text = '' |
| 474 | for sentence in sentences: |
| 475 | sentiment\_total = 0 |
| 476 | words\_checked = 0 |
| 477 | sentence\_words = word\_tokenize(sentence) |
| 478 | for word in sentence\_words: |
| 479 | try: |
| 480 | sentiment\_total += afinnwords[word.lower()] |
| 481 | words\_checked += 1 |
| 482 | except: |
| 483 | pass |
| 484 | if words\_checked != 0: |
| 485 | sentence\_sentiment = sentiment\_total / words\_checked |
| 486 | sentiments.append(sentence\_sentiment) |
| 487 | else: |
| 488 | sentiments.append(0) |
| 489 | if len(sentences) == 2: |
| 490 | if abs(sentiments[0] - sentiments[1]) > 3: |
| 491 | sentences[0] = "<mark>" + sentences[0] |
| 492 | sentences[1] = sentences[1] + "</mark>" |
| 493 | has = 1 |
| 494 | else: |
| 495 | for sent in range(0, len(sentences) - 2): |
| 496 | if abs(sentiments[sent] - sentiments[sent + 1]) > 3: |
| 497 | sentences[sent] = "<mark>" + sentences[sent] |
| 498 | sentences[sent + 1] = sentences[sent + 1] + "</mark>" |
| 499 | has = 1 |
| 500 | output\_text = ' '.join(s for s in sentences) |
| 501 | return Markup(output\_text), has |
| 502 |  |
| 503 |  |
| 504 | class Reader: |
| 505 | """ |
| 506 | Class that defines all the ways that the text can be tokenised |
| 507 |  |
| 508 | Properties: |
| 509 | content - raw text extracted from the file |
| 510 |  |
| 511 | Methods: |
| 512 | word\_tok - returns an array containing all the words in the text |
| 513 | sent\_tok - returns an array containing all the sentences in the text |
| 514 | no\_stops - returns the raw text without any stop words |
| 515 | no\_stops\_words - returns an array containing all the words in the text |
| 516 | without any stop words |
| 517 | no\_stops\_sents - returns an array containing all the sentences in the text |
| 518 | without any stop words |
| 519 | """ |
| 520 | def \_\_init\_\_(self): |
| 521 | self.content = '' |
| 522 |  |
| 523 | def word\_tok(self): |
| 524 | words = word\_tokenize(self.content) |
| 525 | return words |
| 526 |  |
| 527 | def sent\_tok(self): |
| 528 | sents = sent\_tokenize(self.content) |
| 529 | return sents |
| 530 |  |
| 531 | def no\_stops(self): |
| 532 | words = self.word\_tok() |
| 533 | no\_stop = ''.join(w + ' ' for w in words if w not in stops) |
| 534 | return no\_stop |
| 535 |  |
| 536 | def no\_stops\_words(self): |
| 537 | text = self.no\_stops() |
| 538 | words = word\_tokenize(text) |
| 539 | return words |
| 540 |  |
| 541 | def no\_stops\_sents(self): |
| 542 | text = self.no\_stops() |
| 543 | sents = sent\_tokenize(text) |
| 544 | return sents |
| 545 |  |
| 546 | """ |
| 547 | Basic class overview for the following 4 classes |
| 548 | All get text from the specified file type |
| 549 |  |
| 550 | Properties: |
| 551 | ext - extension of the file (if applicable) |
| 552 | title - title, depends on file name or web page title |
| 553 | content - raw text from the source |
| 554 |  |
| 555 | Methods: |
| 556 | get\_text - method used to extract the text from the source |
| 557 |  |
| 558 | """ |
| 559 |  |
| 560 |  |
| 561 | class WebPage(Reader): |
| 562 | """ |
| 563 | Loads the text from a web page using the diffbot article api |
| 564 | """ |
| 565 | def \_\_init\_\_(self, directory): |
| 566 | super().\_\_init\_\_() |
| 567 | self.ext = '' |
| 568 | self.title = '' |
| 569 | self.content = self.get\_text(directory) |
| 570 |  |
| 571 | def get\_text(self, source): |
| 572 | r = requests.get("http://api.diffbot.com/v3/article?url=" + |
| 573 | source + "&token=a38083295817ceee22404a79c7400b96") |
| 574 | ro = json.loads(r.text) |
| 575 | self.title = ro['objects'][0]['title'] |
| 576 | return ro['objects'][0]['text'] |
| 577 |  |
| 578 |  |
| 579 | class TextFile(Reader): |
| 580 | """ |
| 581 | Gets text from a text file\_text |
| 582 | """ |
| 583 | def \_\_init\_\_(self, directory): |
| 584 | super().\_\_init\_\_() |
| 585 | self.ext = ".txt" |
| 586 | self.content = self.get\_text(directory) |
| 587 |  |
| 588 | def get\_text(self, source): |
| 589 | file\_text = '' |
| 590 | with open(source, encoding='utf-8') as f: |
| 591 | for line in f: |
| 592 | line = line.replace('\n', ' ') |
| 593 | if line != ' ' or line != '': |
| 594 | file\_text += line |
| 595 | return Markup(file\_text) |
| 596 |  |
| 597 |  |
| 598 | class DOCXFile(Reader): |
| 599 | """ |
| 600 | Gets text from a docx file |
| 601 | """ |
| 602 | def \_\_init\_\_(self, directory): |
| 603 | super().\_\_init\_\_() |
| 604 | self.ext = ".docx" |
| 605 | self.content = self.get\_text(directory) |
| 606 |  |
| 607 | def get\_text(self, source): |
| 608 | text = '' |
| 609 | document = Document(source) |
| 610 | for p in document.paragraphs: |
| 611 | text += p.text |
| 612 | return text |
| 613 |  |
| 614 |  |
| 615 | class PDFFile(Reader): |
| 616 | """ |
| 617 | Gets text from a pdf file |
| 618 | """ |
| 619 | def \_\_init\_\_(self, directory): |
| 620 | super().\_\_init\_\_() |
| 621 | self.ext = ".pdf" |
| 622 | self.content = self.get\_text(directory) |
| 623 |  |
| 624 | def get\_text(self, source): |
| 625 | text = '' |
| 626 | pdfObject = open(source, 'rb') |
| 627 | pdfReader = PyPDF2.PdfFileReader(pdfObject) |
| 628 | for page in range(0, pdfReader.numPages): |
| 629 | pageObj = pdfReader.getPage(page) |
| 630 | text += pageObj.extractText() |
| 631 | return text |

/static/scripts/sorts.py

|  |  |
| --- | --- |
| 1 | def quick\_sort(numbers\_to\_sort): |
| 2 | """ |
| 3 | Implementation of the quick sort algorithm |
| 4 | """ |
| 5 | less = [] |
| 6 | greater = [] |
| 7 | equal = [] |
| 8 |  |
| 9 | if len(numbers\_to\_sort) > 1: |
| 10 | pivot = int(numbers\_to\_sort[0][1]) |
| 11 | equal.append(numbers\_to\_sort[0]) |
| 12 | for x in range(1, len(numbers\_to\_sort)): |
| 13 | if int(numbers\_to\_sort[x][1]) > pivot: |
| 14 | greater.append(numbers\_to\_sort[x]) |
| 15 | elif int(numbers\_to\_sort[x][1]) < pivot: |
| 16 | less.append(numbers\_to\_sort[x]) |
| 17 | else: |
| 18 | equal.append(numbers\_to\_sort[x]) |
| 19 | return quick\_sort(less) + equal + quick\_sort(greater) |
| 20 | else: |
| 21 | return numbers\_to\_sort |
| 22 |  |
| 23 |  |
| 24 | def radix\_sort(numbers\_to\_sort): |
| 25 | """ |
| 26 | Implementation of the radix sort algorithm |
| 27 | """ |
| 28 | list\_length = len(numbers\_to\_sort) |
| 29 | length = len(numbers\_to\_sort[0][1]) |
| 30 | while length > 0: |
| 31 | for x in range(0, length): |
| 32 | buckets = [0 for i in range(10)] |
| 33 | for tup in numbers\_to\_sort: |
| 34 | num\_str = tup[1] |
| 35 | buckets[int(num\_str[x])] += 1 |
| 36 | for x in range(1, 10): |
| 37 | buckets[x] += buckets[x - 1] |
| 38 | sort\_list = [0 for i in range(list\_length)] |
| 39 | for tup in reversed(numbers\_to\_sort): |
| 40 | num\_str = tup[1] |
| 41 | buckets[int(num\_str[length - 1])] -= 1 |
| 42 | sort\_list[buckets[int(num\_str[length - 1])]] = tup |
| 43 | length -= 1 |
| 44 | numbers\_to\_sort = sort\_list |
| 45 | return numbers\_to\_sort |

/static/scripts/tf\_idf.py

|  |  |
| --- | --- |
| 1 | from nltk import sent\_tokenize, word\_tokenize |
| 2 | from nltk.corpus import stopwords |
| 3 | from math import log |
| 4 | import pickle |
| 5 | from nltk.corpus import gutenberg, brown, reuters, inaugural |
| 6 | A = 0.4 |
| 7 |  |
| 8 | stops = set(stopwords.words('english')) |
| 9 | punc = ["!", "?", "``", '\*', '(', ')', '-', '{', '}', '[', ']', ':', |
| 10 | ';', "'", ',', '.', '\\', '/', "''", '"', "”", "“", "–", |
| 11 | "--", "—"] |
| 12 | contracted\_words = ["'s", "'m", "n't", "'re", "it’s", "i’ve", "they’re", 'we’re'] |
| 13 |  |
| 14 |  |
| 15 | def radix\_sort(numbers\_to\_sort): |
| 16 | """ |
| 17 | Implementation of the radix sort algorithm |
| 18 | """ |
| 19 | list\_length = len(numbers\_to\_sort) |
| 20 | length = len(numbers\_to\_sort[0][1]) |
| 21 | while length > 0: |
| 22 | for x in range(0, length): |
| 23 | buckets = [0 for i in range(10)] |
| 24 | for tup in numbers\_to\_sort: |
| 25 | num\_str = tup[1] |
| 26 | buckets[int(num\_str[x])] += 1 |
| 27 | for x in range(1, 10): |
| 28 | buckets[x] += buckets[x - 1] |
| 29 | sort\_list = [0 for i in range(list\_length)] |
| 30 | for tup in reversed(numbers\_to\_sort): |
| 31 | num\_str = tup[1] |
| 32 | buckets[int(num\_str[length - 1])] -= 1 |
| 33 | sort\_list[buckets[int(num\_str[length - 1])]] = tup |
| 34 | length -= 1 |
| 35 | numbers\_to\_sort = sort\_list |
| 36 | return numbers\_to\_sort |
| 37 |  |
| 38 |  |
| 39 | class TF\_IDF(): |
| 40 | """ |
| 41 | Algorithm to calculate the significance of a word in a text |
| 42 |  |
| 43 | Properties: |
| 44 | document\_frequency - has a record for each text trained and holds the |
| 45 | number of occurences of each word in that text |
| 46 |  |
| 47 | Methods: |
| 48 | count\_words - given a text counts the instances of each word |
| 49 | train\_frequencies - adds a new text to the document\_frequency |
| 50 | normalise\_frequency - defines the frequency of a word in relation to the |
| 51 | frequencies of other words |
| 52 | find\_idf - finds the inverse document frequnecy of a word based on the |
| 53 | number of texts that word appears in |
| 54 | check\_new\_text - Calculates the tf-idf of a new text and sorts the list |
| 55 | to find the top 5 most significant words in that text |
| 56 | """ |
| 57 |  |
| 58 | def \_\_init\_\_(self): |
| 59 | self.number\_of\_documents = 0 |
| 60 | self.document\_frequency = {} |
| 61 |  |
| 62 | def count\_words(self, text): |
| 63 | words = word\_tokenize(text) |
| 64 | text\_words = {} |
| 65 | for word in words: |
| 66 | word = word.lower() |
| 67 | if word not in stops and word not in punc and word not in contracted\_words: |
| 68 | current\_words = list(text\_words.keys()) |
| 69 | if word in current\_words: |
| 70 | text\_words[word] += 1 |
| 71 | else: |
| 72 | text\_words[word] = 1 |
| 73 | return text\_words |
| 74 |  |
| 75 | def train\_frequencies(self, text): |
| 76 | text\_words = word\_tokenize(text) |
| 77 | unique\_words = set(text\_words) |
| 78 | for word in unique\_words: |
| 79 | try: |
| 80 | self.document\_frequency[word] += 1 |
| 81 | except: |
| 82 | self.document\_frequency[word] = 1 |
| 83 | self.number\_of\_documents += 1 |
| 84 |  |
| 85 | def normalise\_frequency(self, word\_count, highest\_raw): |
| 86 | # Formula to normalise the frequency |
| 87 | normalise\_frequency = A + ((1 - A) \* (word\_count / highest\_raw)) |
| 88 | return normalise\_frequency |
| 89 |  |
| 90 | def find\_idf(self, word): |
| 91 | try: |
| 92 | documents\_containing = self.document\_frequency[word] + 1 |
| 93 | except: |
| 94 | documents\_containing = 1 |
| 95 | idf = log((self.number\_of\_documents / (documents\_containing)), 10) |
| 96 | return idf |
| 97 |  |
| 98 | # returns the top five word in the new text |
| 99 | def check\_new\_text(self, text): |
| 100 | words = set(word\_tokenize(text)) |
| 101 | word\_count = self.count\_words(text) |
| 102 | highest\_raw = word\_count[max(word\_count, key=lambda i: word\_count[i])] |
| 103 | tf\_idfs = {} |
| 104 | for word in words: |
| 105 | word = word.lower() |
| 106 | if word not in stops and word not in punc and word not in contracted\_words: |
| 107 | ntf = self.normalise\_frequency(word\_count[word], highest\_raw) |
| 108 | idf = self.find\_idf(word) |
| 109 | tf\_idf = str(ntf \* idf) |
| 110 | # converts in to format that is sortable by radix Algorithm |
| 111 | if float(tf\_idf) >= 0: |
| 112 | if tf\_idf[0] == "1": |
| 113 | tf\_idfs[word] = "1" + tf\_idf[2:5] |
| 114 | else: |
| 115 | tf\_idfs[word] = "0" + tf\_idf[2:5] |
| 116 | lst = [] |
| 117 | if len(tf\_idfs) == 0: |
| 118 | return [] |
| 119 | for word, ti in tf\_idfs.items(): |
| 120 | lst.append(tuple([word, ti.zfill(4)])) |
| 121 | print(lst) |
| 122 | # Gets 5 words with the highest tf-idf |
| 123 | self.train\_frequencies(text) |
| 124 | return radix\_sort(lst)[-5:] |

/static/js/index.js

|  |  |
| --- | --- |
| 1 | document.getElementById('fileInput').onchange = function () { |
| 2 | var filename; |
| 3 | filename = this.value.split('\\'); |
| 4 | filename = filename[filename.length - 1]; |
| 5 | document.getElementById("filename-dis").value = filename; |
| 6 | }; |

/Templates/header.html

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html> |
| 2 | <html lang="en"> |
| 3 |  |
| 4 | <head> |
| 5 | <meta charset="UTF-8"> |
| 6 | <title>Text Analyser</title> |
| 7 | <meta name="viewport" content="width=device-width, initial-scale=1"> |
| 8 | <link href="{{ url\_for ('static', filename='css/bootstrap.min.css') }}" rel="stylesheet"> |
| 9 | <link href="{{ url\_for ('static', filename='css/mystyle.css') }}" rel="stylesheet"> |
| 10 | </head> |
| 11 |  |
| 12 | <header> |
| 13 | <nav class="navbar navbar-default"> |
| 14 | <div class="container-fluid"> |
| 15 | <div class="navbar-header"> |
| 16 | <a style="color: #2b669a;" class="navbar-brand" href="{{ url\_for('index') }}">Text Analyser</a> |
| 17 | </div> |
| 18 |  |
| 19 | <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1"> |
| 20 |  |
| 21 | <ul class="nav navbar-nav"> |
| 22 | <li><a data-toggle="modal" data-target="#AboutModal">About</a></li> |
| 23 |  |
| 24 | <div class="modal fade" tabindex="-1" role="dialog" id="AboutModal"> |
| 25 | <div class="modal-dialog" role="document"> |
| 26 | <div class="modal-content"> |
| 27 | <div class="modal-header"> |
| 28 | <h4 class="modal-title">About</h4> |
| 29 | </div> |
| 30 | <div class="modal-body"> |
| 31 | <div class="container"> |
| 32 | <h3> |
| 33 | A text Analyser made using Python 3 and <br> |
| 34 | Flask. Simple to use, just insert the url <br> |
| 35 | or directory of the file that you want to <br> |
| 36 | be analysed and press enter. Or if you'd <br> |
| 37 | prefer you can copy the text in to the box <br> |
| 38 | box for the same effect. The analyser return <br> |
| 39 | a wide array of results including uses of <br> |
| 40 | punctuation, words and sentences, and other <br> |
| 41 | features such as reading age and alliteration.<br> |
| 42 | </h3> |
| 43 | </div> |
| 44 | </div> |
| 45 | <div class="modal-footer"> |
| 46 | <button type="button" class="btn btn-default" data-dismiss="modal">Close</button> |
| 47 | </div> |
| 48 | </div> |
| 49 | <!-- /.modal-content --> |
| 50 | </div> |
| 51 | <!-- /.modal-dialog --> |
| 52 | </div> |
| 53 | <!-- /.modal --> |
| 54 |  |
| 55 | </ul> |
| 56 |  |
| 57 | <ul class="nav navbar-nav navbar-right"> |
| 58 | {% if session.logged\_in %} |
| 59 | <li><a data-toggle="modal" data-target="#notifs"> |
| 60 | <span class="glyphicon glyphicon-bell"></span> |
| 61 | </a></li> |
| 62 | <div class="modal fade" id="notifs"> |
| 63 | <div class="modal-dialog modal-sm"> |
| 64 | <div class="modal-content"> |
| 65 | <div class="modal-header"> |
| 66 | <h4 class="modal-title">Notifications</h4> |
| 67 | </div> |
| 68 | <div class="modal-body"> |
| 69 | <div class="container"> |
| 70 | {% for notif in session.notifs %} |
| 71 | <p> {{ notif }} <p> |
| 72 | {% endfor %} |
| 73 | </div> |
| 74 | </div> |
| 75 | <div class="modal-footer"> |
| 76 | <button type="button" class="btn btn-default" data-dismiss="modal">Close</button> |
| 77 | </div> |
| 78 | </div> |
| 79 | </div> |
| 80 | </div> |
| 81 |  |
| 82 | <li> |
| 83 | <a href="{{ url\_for('profile', username=session.username) }}">{{ session.username }}</a> |
| 84 | </li> |
| 85 |  |
| 86 | <li> |
| 87 | <a href="{{ url\_for('logout') }}">Logout</a> |
| 88 | </li> |
| 89 | {% else %} |
| 90 | <li> |
| 91 | <a href="{{ url\_for('login\_page') }}">Login</a> |
| 92 | </li> |
| 93 |  |
| 94 | <li> |
| 95 | <a href="{{ url\_for('register\_page') }}">Sign Up</a> |
| 96 | </li> |
| 97 | {% endif %} |
| 98 |  |
| 99 | </ul> |
| 100 |  |
| 101 | </div> |
| 102 | </div> |
| 103 | </nav> |
| 104 | </header> |
| 105 |  |
| 106 | <body> |
| 107 | <div class="border-row" style="width:100%; margin-bottom: 15px;"></div> |
| 108 | <div class="container-fluid" align="center" style="min-height:100%"> |
| 109 | {% block body %}{% endblock %} |
| 110 | </div> |
| 111 | <script src="//code.jquery.com/jquery-1.11.1.min.js"></script> |
| 112 | <script type="text/javascript" src="{{ url\_for('static', filename='js/bootstrap.min.js') }}"></script> |
| 113 | <script type="text/javascript" src="{{ url\_for('static', filename='js/myjs.js') }}"></script> |
| 114 | </body> |
| 115 |  |
| 116 | </html> |

/template/index.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} |
| 2 | {% block body %} |
| 3 | <body> |
| 4 | <div class="page-header"> |
| 5 | <img src="{{ url\_for('static', filename='images/textanalyser\_header3.png') }} " style="width:100%;"></img> |
| 6 | </div> |
| 7 |  |
| 8 | {% with messages = get\_flashed\_messages() %} {% if messages %} |
| 9 | <div class="alert alert-warning"> |
| 10 | {% for message in messages %} {{ message }} {% endfor %} |
| 11 | </div> |
| 12 | {% endif %} {% endwith %} |
| 13 |  |
| 14 | <div class="container-fluid"> |
| 15 | <div class="row"> |
| 16 | <div class="col-md-4"> |
| 17 | <form method="POST" action="{{ url\_for('upload\_file') }}" enctype=multipart/form-data> |
| 18 | <div class="input-group"> |
| 19 | <label class="input-group-btn"> |
| 20 | <span class="btn btn-primary"> |
| 21 | Browse&hellip; <input type="file" id="fileInput" style="display: none;" name="file" multiple> |
| 22 | </span> |
| 23 | </label> |
| 24 | <input type="text" id="filename-dis" class="file-name-dis input-group-addon" readonly> |
| 25 | <input class="btn btn-success" type=submit value=Upload> |
| 26 | </div> |
| 27 | </form> |
| 28 | </div> |
| 29 |  |
| 30 | <div class="col-md-4 ta-but"> |
| 31 | <form method="POST" action="{{ url\_for('raw\_text\_upload') }}" enctype=multipart/form-data> |
| 32 | <textarea cols="57%" rows="10%" name="raw\_text" placeholder="Type or paste your text here..." style="resize:vertical;"></textarea> |
| 33 | <input class="btn btn-success but" type=submit value=Upload> |
| 34 | </form> |
| 35 | </div> |
| 36 |  |
| 37 | <div class="col-md-4"> |
| 38 | <div class="input-group"> |
| 39 | <label class="input-group-btn"> |
| 40 | <button class="btn btn-primary" onclick="paste\_url()"> |
| 41 | Paste&hellip; |
| 42 | </button> |
| 43 | </label> |
| 44 | <form method="POST" action="{{ url\_for('webtext', analysis="Regular") }}" enctype=multipart/form-data> |
| 45 | <input id="url-dis" type="text" name="url" class="file-name-dis input-group-addon" placeholder="Paste URL here..."> |
| 46 | <input class="btn btn-success" type=submit value=Go...> |
| 47 | </form> |
| 48 | </div> |
| 49 | </div> |
| 50 | </div> |
| 51 | </div> |
| 52 | <script src="//code.jquery.com/jquery-1.11.1.min.js"></script> |
| 53 | <script type="text/javascript" src="{{ url\_for('static', filename='js/bootstrap.min.js') }}"></script> |
| 54 | <script type="text/javascript" src="{{ url\_for('static', filename='js/index.js') }}"></script> |
| 55 | </body> |
| 56 |  |
| 57 | </html> |
| 58 | {% endblock %} |

/templates/login.html

|  |  |
| --- | --- |
| 1 | {% extends "header.html" %} {% block body %} |
| 2 |  |
| 3 | <body> |
| 4 | <div class="container"> |
| 5 | {% with messages = get\_flashed\_messages() %} {% if messages %} |
| 6 | <div class="alert alert-warning"> |
| 7 | {% for message in messages %} {{ message }} {% endfor %} |
| 8 | </div> |
| 9 | {% endif %} {% endwith %} |
| 10 | <br> |
| 11 | <h4>Please Login:</h4> |
| 12 | <br> |
| 13 | <form action="" class="form-inline" method="post"> |
| 14 | <input type="text" class="form-control" placeholder="Username" name="username" value="{{request.form.username}}"> |
| 15 | <input type="password" class="form-control" placeholder="Password" name="password" value="{{request.form.password}}"> |
| 16 | <input class="btn btn-default" type="submit" value="Login"> |
| 17 | </form> |
| 18 |  |
| 19 | </div> |
| 20 | </body> |
| 21 |  |
| 22 | {% endblock %} |

/templates/register.html

|  |  |
| --- | --- |
| 1 | {% extends "header.html" %} |
| 2 |  |
| 3 | {% block body %} |
| 4 |  |
| 5 | <div class="container"> |
| 6 |  |
| 7 | {% with messages = get\_flashed\_messages() %} |
| 8 | {% if messages %} |
| 9 | <div class="alert alert-warning"> |
| 10 | {% for message in messages %} |
| 11 | {{ message }} |
| 12 | {% endfor %} |
| 13 | </div> |
| 14 | {% endif %} |
| 15 | {% endwith %} |
| 16 |  |
| 17 | <form method="POST" action="/register/"> |
| 18 | <dl> |
| 19 | <dt><label for="username">Username</label></dt> |
| 20 | <dd><input id="username" name="username" type="text" value=""></dd> |
| 21 | </dl> |
| 22 | <dl> |
| 23 | <dt><label for="email">Email Address</label></dt> |
| 24 | <dd><input id="email" name="email" type="text" value=""></dd> |
| 25 | </dl> |
| 26 | <dl> |
| 27 | <dt><label for="password">Password</label></dt> |
| 28 | <dd><input id="password" name="password" type="password" value=""></dd> |
| 29 | </dl> |
| 30 | <dl> |
| 31 | <dt><label for="confirm">Confirm Password</label></dt> |
| 32 | <dd><input id="confirm" name="confirm" type="password" value=""></dd> |
| 33 | </dl> |
| 34 | <input class="btn btn-default" type="submit" value="Login"> |
| 35 | </form> |
| 36 | </div> |
| 37 |  |
| 38 | {% endblock %} |

/templates/profile.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} {% block body %} |
| 2 |  |
| 3 | {% with messages = get\_flashed\_messages() %} |
| 4 | {% if messages %} |
| 5 | <ul class="alert alert-warning"> |
| 6 | {% for message in messages %} |
| 7 | {{ message }} |
| 8 | {% endfor %} |
| 9 | </ul> |
| 10 | {% endif %} |
| 11 | {% endwith %} |
| 12 | <div class="row"> |
| 13 | <div class="page-header"> |
| 14 | <h1 style="font-size: 50px; color: #2b669a;"> {{ session.username }} </h1> |
| 15 | <div class="btn-group"> |
| 16 | <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"> |
| 17 | Account Settings |
| 18 | <span class="caret"></span> |
| 19 | <span class="sr-only">Toggle Dropdown</span> |
| 20 | </button> |
| 21 | <ul class="dropdown-menu"> |
| 22 | <li><a href="{{ url\_for('changepassword') }}">Change Password</a></li> |
| 23 | <li role="separator" class="divider"></li> |
| 24 | <li><a href="{{ url\_for('deleteaccount') }}">Delete Account</a></li> |
| 25 | </ul> |
| 26 | </div> |
| 27 | </div> |
| 28 | </div> |
| 29 | <div class="row"> |
| 30 | <div class="col-md-8"> |
| 31 | <h2 style="text-align:left; width:90%;"> Owned Text Search Results: </h2> |
| 32 | <br> |
| 33 | {% for text in owned\_texts %} |
| 34 | {% if text == False %} |
| 35 | <h3> No Texts Found </h3> |
| 36 | {% else %} |
| 37 | <div class="panel panel-default" style="width:90%; align:right;"> |
| 38 | <div class="panel-heading"> |
| 39 | <a href={{ url\_for( 'textdisplay', textTitle=text.title, analysis="Regular" ) }} class="panel-title" style="text-decoration: none;">{{ text.title }} |
| 40 | </div> |
| 41 |  |
| 42 | <div class="panel-body"> |
| 43 | {{ text.body }} |
| 44 | </div> |
| 45 | </a> |
| 46 | </div> |
| 47 | {% endif %} |
| 48 | {% endfor %} |
| 49 |  |
| 50 | <h2 style="text-align:left; width:90%;"> Shared Text Search Results: </h2> |
| 51 | <br> |
| 52 | {% for text in shared\_texts %} |
| 53 | {% if text == False %} |
| 54 | <h3> No Texts Found </h3> |
| 55 | {% else %} |
| 56 | <div class="panel panel-default" style="width:90%; align:right;"> |
| 57 | <div class="panel-heading"> |
| 58 | <a href={{ url\_for( 'textdisplay', textTitle=text.title, analysis="Regular" ) }} class="panel-title" style="text-decoration: none;">{{ text.title }} |
| 59 | </div> |
| 60 |  |
| 61 | <div class="panel-body"> |
| 62 | {{ text.body }} |
| 63 | </div> |
| 64 | </a> |
| 65 | </div> |
| 66 | {% endif %} |
| 67 | {% endfor %} |
| 68 | <br> |
| 69 | <br> |
| 70 | </div> |
| 71 |  |
| 72 | <div class="col-md-2" style="padding-left:0px;padding-right:50px;"> |
| 73 | <form name="search-title" method="POST" action="{{ url\_for('search\_titles', username=username) }}"> |
| 74 |  |
| 75 | <h3 style="text-align:left;">Search by Title: </h3> |
| 76 | <input type="text" name="title"> |
| 77 | <br> |
| 78 | <hr> |
| 79 | <button class="btn btn-success" type="submit"> Search </button> |
| 80 |  |
| 81 | <br> |
| 82 | <hr> |
| 83 | </form> |
| 84 |  |
| 85 | <form name="search-keywords" method="POST" action="{{ url\_for('search\_keywords', username=username) }}"> |
| 86 |  |
| 87 | <h3 style="text-align:left;">Search Keyword: </h3> |
| 88 |  |
| 89 | <input type="text" name="keyword"> |
| 90 | <br> |
| 91 | <hr> |
| 92 | <button class="btn btn-success" type="submit"> Search </button> |
| 93 |  |
| 94 | <br> |
| 95 | <hr> |
| 96 | </form> |
| 97 |  |
| 98 | <form name="search-category" methods="POST" action="{{ url\_for('search\_category', username=username) }}"> |
| 99 | <h3 style="text-align:left;">Search Category: </h3> |
| 100 | <select name="category"> |
| 101 | {% for category in categories %} |
| 102 | <option value="{{ category }}"> {{ category }} </option> |
| 103 | {% endfor %} |
| 104 | </select> |
| 105 | <br> |
| 106 | <hr> |
| 107 | <button class="btn btn-success" type="submit"> Search </button> |
| 108 | <br> |
| 109 | <hr> |
| 110 | </form> |
| 111 |  |
| 112 | </div> |
| 113 |  |
| 114 | <div class="col-md-2" style="padding-left:0px;padding-right:50px;"> |
| 115 | <form name="search\_features" method="POST" action="{{ url\_for('search\_values', username=username) }}"> |
| 116 | <h3 style="text-align:left;"> Search by Feature: </h3> |
| 117 | <label> Reading Age Above: </label> |
| 118 | <input type="range" min="0" max="25" value="0" id="RAS" |
| 119 | step="0.5" oninput="outputUpdate(value, id)" name="RAS"> |
| 120 | <output for="fader" id="RASO">0</output> |
| 121 |  |
| 122 | <label> Sentiment Above: </label> |
| 123 | <input type="range" min="-5" max="5" value="-5" id="SSA" |
| 124 | step="0.1" oninput="outputUpdate(value, id)" name="SSA"> |
| 125 | <output for="fader" id="SSAO">-5</output> |
| 126 |  |
| 127 | <label> Sentiment Below: </label> |
| 128 | <input type="range" min="-5" max="5" value="5" id="SSB" |
| 129 | step="0.1" oninput="outputUpdate(value, id)" name="SSB"> |
| 130 | <output for="fader" id="SSBO">5</output> |
| 131 |  |
| 132 | <label> Text Contains: </label> |
| 133 | <br> |
| 134 | <input type="checkbox" name="Alliteration">Alliteration</input> |
| 135 | <br> |
| 136 | <input type="checkbox" name="Antithesis">Antithesis</input> |
| 137 | <br> |
| 138 | <input type="checkbox" name="Juxtaposition">Juxtaposition</input> |
| 139 |  |
| 140 | <script> |
| 141 | function outputUpdate(val, output\_id) { |
| 142 | output\_id = output\_id.concat("O") |
| 143 | output\_id = "#".concat(output\_id) |
| 144 | document.querySelector(output\_id).value = val; |
| 145 | } |
| 146 | </script> |
| 147 | <hr> |
| 148 | <button class="btn btn-success" type="submit"> Search </button> |
| 149 | <br> |
| 150 | <hr> |
| 151 | </form> |
| 152 | </div> |
| 153 | </div> |
| 154 | {% endblock %} |

/templates/textdisplay.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} {% block body %} {% with messages = get\_flashed\_messages() %} {% if messages %} |
| 2 | <div class="alert alert-warning"> |
| 3 | {% for message in messages %} {{ message }} {% endfor %} |
| 4 | </div> |
| 5 | {% endif %} {% endwith %} {% if upload %} |
| 6 | <form method="POST" action="{{ url\_for('save\_text')}}" name="titleform"> |
| 7 | <div class="page-header"> |
| 8 | <div class="container"> |
| 9 | <div class="row"> |
| 10 | <textarea name='title' rows=1 cols=15 class="titleinput col-sm-7 col-sm-offset-1" type=text>{{ title }}</textarea> {% if ext %} |
| 11 | <textarea rows=1 cols=15 class="titleinput form-control col-sm-1" type=text readonly>{{ ext }}</textarea> {% endif %} |
| 12 | </div> |
| 13 | </div> |
| 14 | </div> |
| 15 | {% else %} |
| 16 | <div class="page-header"> |
| 17 | <h1 style="font-size: 75px; color: #2b669a;"> {{ title }} </h1> |
| 18 | <span class="label label-primary">Owner: {{ owner }}</span> |
| 19 | <a class="btn btn-default" href="{{ url\_for('download', texttitle=title)}}" target="\_blank"> Download </a> |
| 20 | </div> |
| 21 | {% endif %} |
| 22 |  |
| 23 | <div class="row"> |
| 24 | <div class="col-md-10"> |
| 25 | <p style="width:80%;font-size:15px;">{{ text }}</p> |
| 26 | {% if upload %} |
| 27 | <label>Category: </label> |
| 28 | <input type="text" name="Category"></input> |
| 29 | <input type=submit value=Save> |
| 30 | {% else %} |
| 31 | {% if user == owner %} |
| 32 | <a class="btn btn-default inner" href="{{ url\_for('share', texttitle=title)}}" >Share Text</a> |
| 33 | <form action="{{ url\_for('deletetext', texttitle = title) }}"> |
| 34 | <button class="btn btn-default inner" type=submit value="Delete Text">Delete Text</button> |
| 35 | </form> |
| 36 | {% endif %} |
| 37 | <button class="btn btn-default inner" type="button" onclick="goBack()">Go Back</button> |
| 38 | <script> |
| 39 | function goBack() { |
| 40 | window.history.back(); |
| 41 | } |
| 42 | </script> |
| 43 | {% endif %} |
| 44 |  |
| 45 | </form> |
| 46 | </div> |
| 47 |  |
| 48 | <div class="col-md-2"> |
| 49 | {% for key in texts %} |
| 50 | {% if upload %} |
| 51 | <form action="{{ url\_for('changeview', analysis=key) }}" method="POST"> |
| 52 | <button class="btn btn-success" type=submit style="margin:2px;">{{ key }}</button> |
| 53 | {% else %} |
| 54 | <a href="{{ url\_for('textdisplay', textTitle = title, analysis=key) }}" method="POST" class="btn btn-success" style="margin:2px;">{{ key }}</a><br> |
| 55 | {% endif %} |
| 56 | </form> |
| 57 | {% endfor %} |
| 58 | </div> |
| 59 | </div> |
| 60 | <hr> |
| 61 | <h2> Text Statistics </h2> |
| 62 | <div class="row"> |
| 63 | {% for key, fact in facts.items() %} |
| 64 | {% if key not in ['Words','Punctuation', 'Key Words'] %} |
| 65 | <h4>{{ key }} <span class="label label-primary">{{ fact }}</span></h4> |
| 66 | {% endif %} |
| 67 | {% endfor %} |
| 68 |  |
| 69 | <h4> Keywords <span class="label label-primary">{{ keywords }}</span></h4> |
| 70 |  |
| 71 | </div> |
| 72 | <hr> |
| 73 | <h2> Text Graphs </h2> |
| 74 | <div class="row"> |
| 75 | {% for key, fact in facts.items() %} |
| 76 | {% if key in ['Words','Punctuation'] %} |
| 77 | <embed type="image/sgv+xml" src={{fact|safe}} style="width:45%"> |
| 78 | </embed> |
| 79 | {% endif %} |
| 80 | {% endfor %} |
| 81 | </div> |
| 82 | {% endblock %} |

/templates/share.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} {% block body %} {% with messages = get\_flashed\_messages() %} {% if messages %} |
| 2 | <div class="alert alert-warning"> |
| 3 | {% for message in messages %} {{ message }} {% endfor %} |
| 4 | </div> |
| 5 | {% endif %} {% endwith %} |
| 6 | <div class="container"> |
| 7 | <div class="page-header"> |
| 8 | <h1 style="font-size: 50px; color: #2b669a;" >Share '{{ title }}'</h1> |
| 9 | </div> |
| 10 | {% if users[0] != False %} |
| 11 | <h2> Select user to share with: </h2> |
| 12 | <br> |
| 13 | {% for user in users %} |
| 14 | <div class="well"> |
| 15 | <a href="{{ url\_for('share\_text', texttitle = title, username=user.username) }}" method="POST" >{{ user.username }}</a> |
| 16 | </div> |
| 17 | {% endfor %} |
| 18 | {% else %} |
| 19 | <h2> No Valid Users </h2> |
| 20 | {% endif %} |
| 21 | </div> |
| 22 | {% endblock %} |

/templates/changepass.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} {% block body %} |
| 2 |  |
| 3 | <div class="container"> |
| 4 |  |
| 5 | {% with messages = get\_flashed\_messages() %} {% if messages %} |
| 6 | <div class="alert alert-warning"> |
| 7 | {% for message in messages %} {{ message }} {% endfor %} |
| 8 | </div> |
| 9 | {% endif %} {% endwith %} |
| 10 |  |
| 11 | <form method="POST" action="{{ url\_for('changepassword') }}"> |
| 12 | <dl> |
| 13 | <dt><label for="oldpass">Current Password</label></dt> |
| 14 | <dd><input id="oldpass" name="oldpass" type="text" value=""></dd> |
| 15 | </dl> |
| 16 | <dl> |
| 17 | <dt><label for="newpass">New Password</label></dt> |
| 18 | <dd><input id="newpass" name="newpass" type="text" value=""></dd> |
| 19 | </dl> |
| 20 | <dl> |
| 21 | <dt><label for="connewpass">Confirm New Password</label></dt> |
| 22 | <dd><input id="connewpass" name="connewpass" type="text" value=""></dd> |
| 23 | </dl> |
| 24 | <input class="btn btn-default" type="submit" value="Login"> |
| 25 | </form> |
| 26 | </div> |
| 27 |  |
| 28 | {% endblock %} |

/templates/404.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} {% block body %} |
| 2 |  |
| 3 | <h1>404 Error</h1> |
| 4 |  |
| 5 | <p>This page does not exist, stop looking for it...</p> |
| 6 | <a href="{{ url\_for('index') }}">Go home, you're drunk.</a> {% endblock%} |

/templates/405.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} |
| 2 |  |
| 3 | {% block body %} |
| 4 |  |
| 5 | <h1>405 Error</h1> |
| 6 |  |
| 7 | <p>Stop trying to break my website, it's frail enough as it is...</p> |
| 8 | <a href="{{ url\_for('index') }}">Go home, you're drunk.</a> |
| 9 |  |
| 10 | {% endblock%} |

/templates/500.html

|  |  |
| --- | --- |
| 1 | {% extends 'header.html' %} |
| 2 |  |
| 3 | {% block body %} |
| 4 |  |
| 5 | <h1>500 Error</h1> |
| 6 |  |
| 7 | <p>My bad this time I guess...</p> |
| 8 | <a href="{{ url\_for('index') }}">Go home, you're drunk.</a> |
| 9 |  |
| 10 | {% endblock%} |