University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Email sensitivity classifier: A web application for predicting sensitive emails

Lewis D M Munro

March 24th 2023

Abstract

The manual labour needed to carry out document sensitivity reviews is rising due to the increase in data collection worldwide. This project develops an email sensitivity classifier and topic model to display results on a web application to the user. A random forest machine learning classifier was used to classify the Enron email collection, these results were then stored in an SQLite database and displayed through a Django based web application. Unit testing was implemented to ensure the reliability of the system and a user study was carried out to test the features implemented in the system. The user study revealed that the system was efficient in determining the sensitivity of emails and that the application was well-designed and easy to use. Overall the system was able to classify an email collection into sensitive and non-sensitive, clarifying that the collection is safe for public release.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Lewis Douglas McDonald Munro Date: 24 March 2023

Contents

1 Introduction

With the rise in data collection and movement for private company information to be made public, a new stress has been placed on those required to scan the documents before public release. Document reviewing is a manual labour intensive process that requires an individual to review a document that may contain sensitive information. Relying heavily on an persons judgement and their ability to process information, we are in a desperate need to transition this process to a digital one.

1.1 Motivation

With the transition from paper based storage to electronic databases we are now able to apply new methods to storing, manipulating and reviewing documents, and people are eager to eliminate all of the tedious processes that came with handling physical documents. The task of reviewing documents before being released to the public is seen as one of the most tedious of them all, but with this transition we are now able to explore and implement new methods to ease this process. With these new methods of document classification and a ready to use collection, we are able to research and produce a solution to this problem.

While any document that is in line for the public eye needs to be reviewed for sensitivity, emails seem to pose a more difficult challenge. Company documents are very likely to be well structured, use professional language, not discuss personal matters and overall be very similar, while emails can be drastically different from one another. An email can contain slang language, bad formatting, personal conversations and many other factors that may contribute towards the emails overall sensitivity. Due to these issues, the individual responsible for looking through a collection of emails may struggle a lot more than one classifying company documents.

The reason behind the movement for these documents to be released, more specifically emails is the fact that they provide accountability for businesses, governments and third sector organisations decisions. It is too common that companies have made poor decisions that drastically effect people lives and due to their data being protected, the public cannot prove them guilty for their actions. With a system that can find these sensitive subjects in the businesses documents, they will be more likely to be held accountable. As well as whole company decisions, a single employee could be using a business email as a personal one, possibly breaking their employee contract by misusing company resources. A system that can be filter sensitive emails by address could aid in the detection of the employees by large organisations.

1.2 Aim

This project will develop a way that machine learning and text classification can be used to aid the manual labour that is needed to sort emails. We will discuss past research and projects that relate to this problem, go through the process of designing and developing the software needed to classify these emails and talk about the results when comparing different approaches and from user evaluations of the final product.

We will mainly be focusing on classifying these emails sensitive and non-sensitive. For an email to classified as sensitive it must contain anything that falls under the personal information category. Another way we will consider organising these emails are by topics. A topic is a group of items that are related in some way, for example there may be a topic for emails that are about federal business and another that talk about customer handling etc.

This specific system will aid users in the identification of sensitive emails in a collection, allow for the identification of addresses that are sending sensitive information regularly and provide a way to organise the email collection by similarity.

1.3 Outline

In this project we will build a web application to provide users with a way to upload and view the classification of the Enron email collection.

The background will discuss the previous research done on the possibilities of new ways to carry out the task of reviewing these documents, different classifiers and how they help in different scenarios.

The requirements will talk about what the web application needs to fulfil the problem, all the possible functionalities it should have and any other features that could be implemented after.

We will talk about the Design the of web application, all of the processes that need to take place for the web application to function and how each aspect of the software will interact with each other. We will also look at the user interface design with wire frames, designs of each page.

We will discuss all the steps taken to implement the web application, this will include the framework of the application, how the data will be stored, all of the changes being made to the data and how each email with be classified. Each major feature feature of the user interface will also be discussed and explained.

An evaluation of the testing and user study will be carried out to further see the impact each feature and choice has had on the application.

2 Background

In this section we will talk about past research papers and projects that relate closely to our problem and we will also go deeper into the main topics that are relevant such as the Freedom of Information act and a background on Machine Learning classifiers.

2.1 Freedom of information

As stated by the Information Commissioner's Office the Freedom of Information Act 2000 (FOI) provides public access to information held by public authorities. A public authority is any organisation that receives money from the public such as the government, schools and the NHS.

With public authorities storing more data about the public than ever before it has become necessary that any information that is held against someone must be readily given to anyone with the right to access it. This is extremely important as these authorities make decisions that could drastically affect the country but by giving the public access to the information they have it is then easier to make them accountable for their actions.

Due to this law and the mass amounts of information that need to be published, the manual processing power to classify these documents has quickly become a hindrance. Public authorities have taken a step towards digitising this process, Using our in-house digital assistance applications, our experienced reviewers can review digital files quickly and accurately. FCDO. The Foreign, Commonwealth and Development Office services (FCDO) provide a combination of digital and manual review process to speed up the manual labour, their still exists a main flaw in the system, the manual reviewer. Compared to computers, humans are slow and are prone to misjudgement, while a well trained systems can process hundreds of documents by the time a manual reviewer has done one.

2.2 Machine learning classification

Machine learning (ML) is a subset of artificial intelligence in which computers use data and algorithms to improve upon themselves. Brewster. With the increase of processing power in our computers we are now able to implement ways for an algorithm to teach itself, in our case we are talking about classification of documents. An algorithm is able to take pre-classified documents and teach itself the patterns that make a specific document part of a category, it can then take what it has learned and look for patterns in new documents and classify them into specified categories.

ML is quickly becoming the foundation of any new software that is being developed, with large organisations applying it to their existing software such as Chat GPT, Google, Facebook and Amazon. Brewster. The applications of ML seem limitless and with everyday computers now able to take advantage of it, we are able to develop new systems that can help in tedious and labour-intensive tasks.

2.3 Research

We will now talk about existing research papers that will give some background on the main topics of this project and some methods we can take to classify sensitive data. Some existing classification systems will be discussed to see if there are any gaps we can fill or features they have that we can take inspiration from.

2.3.1 Related papers

There are multiple research papers on what makes data sensitive and how we can classify it. We will elaborate on why sensitive data can be hard to define and then discuss some options that can be taken to solve our problem.

The Difficulty of Defining Sensitive Data

We will briefly cover the paper The Difficulty of Defining Sensitive Data by Quinn and Malgieri (2022) that discusses the evolution of sensitive data and how it's definition has changed throughout the years.Quinn and Malgieri (2022) also talks about a theory that sensitive data is a "means to an end".

Quinn and Malgieri (2022) claim that in recent times, the nature and use of sensitive data has been changing at a rapid pace. As more of everyday life is becoming digitised, the more sensitive data we are allowing companies to store, and like currency, the more we have of it the less valuable it becomes. Following this increase of sensitive data, the laws that correspond to its protection of it have had to change, the concept has been expanded to cover new categories (Quinn and Malgieri 2022). As the concept of sensitive data covers these new categories the laws around them need to adapt to retain the protection of individuals, but with this changing definition there must come and end to where the concept is so broad that it starts to cover over data protection and freedom of information laws.

The definition of sensitive data is crucial towards how we classify it, if the definition is too vague then the classification of documents becomes opinionated, and the law that protects this data from the public becomes weak. Also with the growing concept of sensitive data, there may come a point where it can be described as useless and the need to classify and hold it from the public becomes irrelevant.

The current definition of sensitive, personal, data as described by the General Data Protection Regulation (GDPR) is.

> 'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person; Intersoft-consulting

We will be using this as our definition for sensitive data throughout the rest of the project.

Machine learning methods for classification of sensitive data

With our definition of sensitive data, we can now go on to discuss approaches for classifying it. We will be discussing a paper called Machine learning methods for classification of sensitive data by (Rudusans and Vitols 2021). In this paper Rudusans and Vitols discuss machine learning methods for classifying defined by general data protection regulation. We will briefly discuss these approaches and their conclusion to a Naive Bayes classifying example.

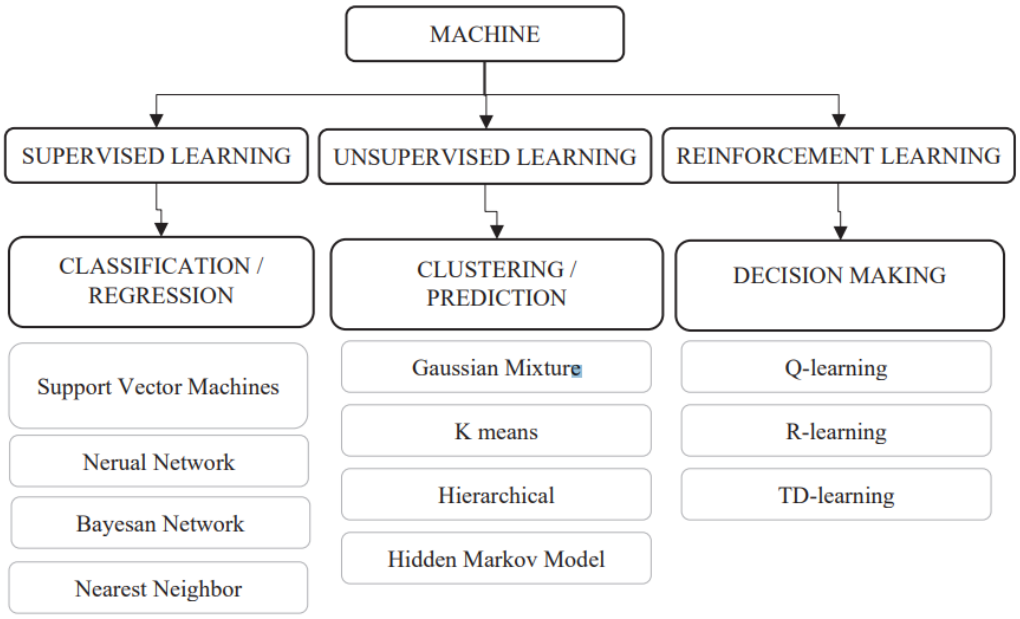


Figure 2.1: *Machine learning techniques. (Rudusans and Vitols 2021)*

As shown in figure 2.1, we can see that they identify 3 machine learning techniques. Supervised learning uses pre-defined labels on a training set of data to teach itself the patterns that belong in each specified category. The classifier is supervised by this training set as it maps data to its known output, identifying the patterns between documents with the same output. This trained classifier can then predict the documents in a new dataset using what it has learned previously.

Unsupervised learning is the opposite of supervised learning, it takes in completely new datasets that have not been classified already and clusters similar documents together. The main difference in the output between supervised and unsupervised is that supervised will label each document from a pre-defined list of labels while unsupervised will output which documents belong to each cluster, these clusters can then be given a name by the user, depending on the patterns that they possess.

Reinforcement learning takes a different approach as it learns on trial and error. These classifiers will take in a dataset and a target end goal to try and reach. It does this by constantly adjusting the parameters it uses to classify a document, if it predicts a document correctly it will be rewarded and learn that the parameters used were a step in the right direction but if it gets a prediction wrong it will be punished and know that it has taken a step in the wrong direction. This is similar to supervised as it knows what the output is meant to be but instead of looking for patterns between what has known it instead learns these patterns from trial and error.

Rudusans ad Vitols carry out an example of using the Naive Bayes classifying method. Naive Bayes is a supervised machine learning method that uses probabilities to classify documents, this method will be discussed later on in this report. They report their conclusions of this example as points on how to achieve an effective classifying system, the two main points are. When we talk about sensitive data and data regulations, we might need to look at some other algorithms or a combination of algorithms (Rudusans and Vitols 2021). They claim that while Naive Bayes is effective and simple, the decision on which classifier to use relies heavily on the scenario in which it is implemented. There is no easy way to know this before implementing a classifier so extensive testing on different methods is required to produce the best output possible. Getting

a good training data set for sensitive data is also a challenging task (Rudusans and Vitols 2021). The output of a supervised machine learning method relies heavily on the training data that it is given and different classifiers will prove more effective on one training set than another. This again comes down to testing which classifier is giving the best outputs and deciding based on these results.

2.3.2 Related systems

We will now look at existing systems that relate to this project. This research is helpful as it allows us to find a gap that needs filled or take inspiration from features that we had not previously thought of. While a specific system that handles sensitivity classification in emails do not seem to exist, there are systems that companies have built into their products that can be discussed.

Gmail spam filter:

One of the main email classifiers that millions of people use everyday without knowing it is Google's Gmail spam filter. While not a lot is known about this filter and how it works, we can still take ideas from what we do know. Gmail also uses an in-house machine learning framework called TensorFlow – alongside some smart AI – to train new spam filters moving forward (Campbell 2022). Tensorflow is a platform that allows for the construction of machine learning systems, it is similar to a web application framework but for machine learning. Although nothing is stated about the "smart AI" that Google uses, it is safe to assume that it is some sort of classification method employed to detect spam emails. While you can apply your own filters in Gmail there is no external application to allow whole collections to be filtered through Google's system, leaving a gap for our system to fill.

Microsoft Purview:

Microsoft Purview is a data management tool that provides a large number of services, including an automatic document sensitivity and labelling system. While I could not find details on how it classifies the data, we can view the user interface and the features that it provides.

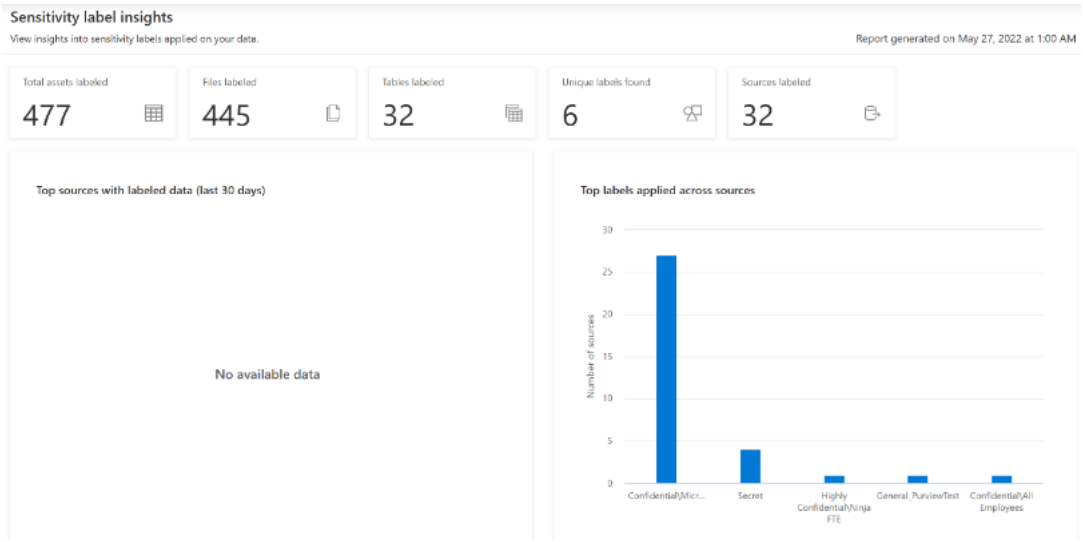


Figure 2.2: *Microsoft Purview user interface. (Microsoft 2023a)*

We can see that in figure 2.2, Microsoft splits the data up into specified labels. While our system will split the emails into sensitive and non-sensitive, we can implement topic modelling. Topic

modelling is an unsupervised machine learning method that clusters documents together that relate closely to each other, we can then look at these clusters and give them appropriate labels. This would allow the user to view how the collection could be split up in a different manner and show that a certain group of documents that belong to the same topic are highly likely to be sensitive.

We can also see specific details about the classification in the user interface, the total number of documents classified, how many labels were identified etc. We can take this and implement graphs to show how many emails were classified as personal or not, we could also show the topics that were identified.

Everlaw:

Everlaw is what is called an ediscovery software, ediscovery is a term used for the process of identifying and delivering electronic information that can be used as evidence in legal cases, (Microsoft 2023b). This software enables law firms, companies and governments to discover and highlight information in documents to then be available for use in investigations.

Everlaw provides a large variety of ways to view documents and aid in speeding up reviewing such as translations into different languages, redacting confidential information and many more helpful features. The main system it seems to be missing though is a sensitivity classifier, it does however contain a clustering algorithm to automatically cluster similar emails together.

Due to our problem being the classification of sensitive and non-sensitive emails, this system would aid massively in manually reviewing emails but is still missing this crucial element our system needs. Our project can develop the idea of combining topic modelling and sensitivity classification together to fill the gap that Everlaw has left. We can also build on the idea of redacting information from the emails, instead of removing the sensitive features we can instead highlight them to show the user what makes the email sensitive.

PleaseReview: PleaseReview is a document reviewing software created by Ideagen, where companies can allocate different documents to reviewers and provide many features to help the speed of document reviews. Real-time document review, co-authoring and redaction, (Ideagen). These are their main two components, allowing reviewers to have real-time discussions when reviewing a document and allowing easy redaction of information.

While this system will drastically speed up the document reviewing process, it does not satisfy the requirements of our system. There is no kind of automatic document classifying, again leaving this gap to be filled by our system.

3 Analysis/Requirements

We will now talk about the requirements that our system will have to meet, these requirements will be taken from our user stories and the research in the previous section. The requirements have been decided using the MoSCoW prioritisation method which splits our requirements up into must have, should have, could have and won't have.

3.1 User Stories

User stories are short scenarios that are used to help highlight certain features that can be implemented into a system, they help give context as to why a feature is or is not important and help solidify the target audience.

> "As the principal of a state school in Scotland I need to view and remove any emails containing personal information before the collection is released to the public about the developments we plan to make to the school."

From this scenario we can see that this person needs some kind of way to view which emails are personal or not. We can consider a few options here, the first being a table of every email with a column that shows if an email has been classified as personal or not, or we can have a feature that highlights every email on the table that has been classified as personal. Both of these options can be combined with a filter that can be added to the table to only show personal or non-personal emails.

> "As a chief of police is would like to be able to view all emails sent and received from a certain officer as I suspect they have been sharing personal information that they have no right to share."

Here we can see the user wants some kind of way to view the details of a specific email address. Here we could implement a user page, this page could show details of an individual email address such as how many emails they have sent or received, how many of these emails are personal and non-personal, who they have been sending emails to etc. Another way to solve this issue is to provide a filter on the email table so we can filter by emails someone has sent or received.

> "As a minister for housing I would like to see charts about the percentages of how many emails contain personal information in our collection and per email address so that we can differentiate between a personal email account and a business one. "

This user has requested a full overview of the email collection itself. We could consider an overview page that shows details of the whole email collection that is being classified, this page could show the total number of personal and non-personal emails in the collection and the email addresses who have sent the most personal emails. This feature also helps solve the issue from the chief of police scenario as a table that shows who has sent the most personal emails can help show if the officer in question is sending a large number of personal emails.

3.2 Functional Requirements

Functional requirements are features that are implemented to help the user accomplish their task, in other words, these are the physical features of the system that the user directly interacts with.

3.2.1 Must have

These are features that our web application must have to provide a full solution to our problem.

Automatic classifying of emails – FM1: This is the core feature of our web application and system with the necessary information to then be provided to the user in different forms

Email table – FM2: A table that shows all of the emails in one place, this provides the user with the ability to easily see all personal and non-personal emails. the table will have the columns To, From, Body, Classified as and Topic

Filter – FM3: A filter on the table to allow the user to filter by each column shown above. The user can then filter by who has sent or received emails, all personal or non-personal emails and by emails that belong to a certain topic.

Email Address page – FM4: A page to show the details of a specific email address. this page will show how many personal and non-personal they have sent or received, who they have sent personal emails to and who they have received personal emails from.

Collection overview – FM5: A page that shows details of the whole email collection. This page will contain graphs that show the amount of personal and non-personal emails in the whole collection, and how many personal emails each email address has sent.

3.2.2 Should have

These are features that our application should have to provide extra functionality to the user, add to our solution in a smaller way, or to provide ease of life aspects.

Email detail page – FS1: A page to allow the user to click on an email on the table to access a page that shows the email in its entirety. It will say if the email has been classified as personal or not, all of the contents of the email i.e. To, From, Body etc.

Manual classify – FS2: An option on the email detail page that allows the user to manually classify the email as personal or not. This allows the user to overwrite the classifiers choice and give a reason for their decision.

Highlight features – FS3: A button to highlight key words on the body of the email. These key words are the features that the classifier used to determine if the email is personal or not.

Topic Modelling – FS4: Topic modelling classification to provide the user with another way to view the collection, this satisfies the discussion about topic modelling from the previous section.

3.2.3 Could have

These are the features that do not benefit to our problem but would instead be a nice additional feature for the user that may help towards a positive experience.

Top words graph – FC1: A graph to show the most important words that the classifier sees as personal. This would fit well on the collection overview page, Allows the user to see the classifiers general reasoning and process on how it has classified the collection

Top words for topics – FC2: A graph to show the most important words for each topic. Similar to the top words graph it would fit well in the collection overview page and allow the user to see why each topic has been chosen.

3.2.4 Won't have

These are the features that have been considered but will not be implemented into the application. This may be due to time restrictions or after consideration they do not seem to benefit the system in any way

Login – FW1: A login feature would allow for the user to store their email collection online and then be able to access their account from different devices. For this project this is not necessary and would take up a considerable amount of time.

3.3 Non-functional Requirements

Non-functional requirements are aspects of the web application that are not so much physical features but instead help to ease the use for users and provide a more positive experience. The requirements are mainly aspects that make a well designed web application and are more general in their specification.

3.3.1 Must have

Consistent User interface – NM1: The main way to make a web application easy to navigate and be pleasant to use is having a consistent user interface. This relates to a consistent design throughout the whole application, similar layouts on every page and a consistent language used for title and text.

Navigation – NM2: To give users a pleasant experience they need an easy way to navigate the application. A consistent navigation bar on every page is needed for this. The navigation bar should never change position from page to page and should be accessible at all times.

Language – NM3: Due to this application implementing very technical features, it is necessary for proper language to be used to allow the user to understand what is going on at all times. There should be no technical terms used unless explained and every figure and piece of data that is shown should be explained in detail.

Performance – NM4: The web application should be able to run smoothly for the user. Any web application that loads pages slowly or takes a long time to load search results will be frustrating for the user.

3.3.2 Won't have

Accessibility options – NW1: Web applications should be flexible and accessible for as many users as possible but due to time restrictions and priorities in other features, there will not be any customisable options on this web application. These options would normally include colour corrections or text to speech but for this project we will be assuming this is available through the browser the user is using.

4 Design

In this section we will talk about possible design approaches we could take towards our application. We will first discuss how the systems of the web application will work, showing the processes of how the user and each aspect of the application interact with each other. We will then look at the user interface, and what the user will see. Each design choice that was made was to fulfil the requirements stated in the previous section, the codes assigned to each requirement will be used to reference it being fulfilled.

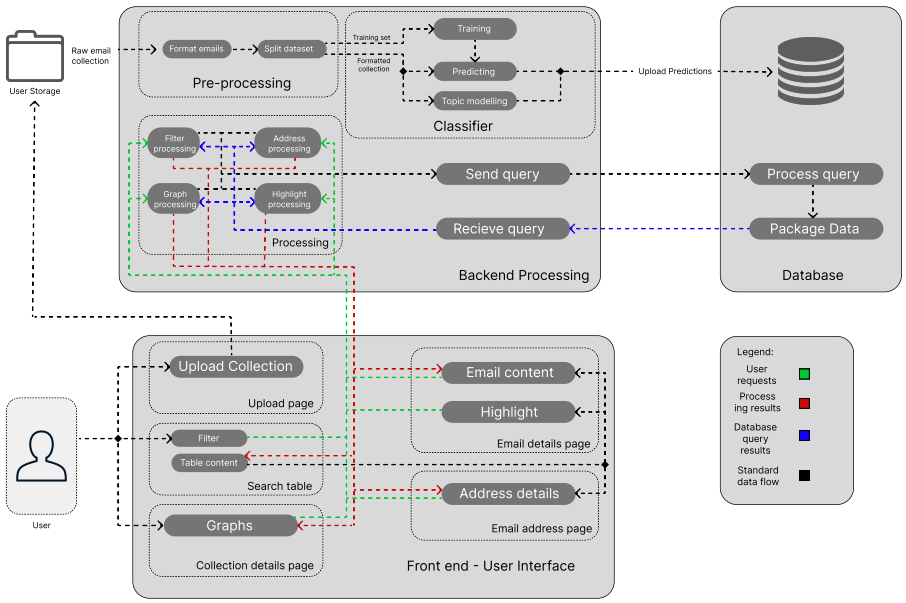


Figure 4.1: *Architecture Diagram*

Figure 4.1 shows an architecture diagram, a diagram showing the main components of the systems and how each aspect interacts with each other. From figure 4.1 we can see there are three main components to our system, the front end which is our user interface, the back end which handles most of the processing of data and the database which will store all of our data. The system is very basic in its structure as requests are very linear in how they are handled.

4.1 Front end

The front end handles all of the direct inputs from the user interface. This is everything that the user can see and interact with, in our system we have a very basic flow of inputs from the user, with accessing all but the upload page triggering a request to the database.

The upload page has a linear data path, with the user accessing the page and selecting to upload the collection. This request is then passed into the pre-processing section of the back end. This feature builds on satisfying requirements FM1 and FS4.

The rest of the components of the front end all follow the same processes, therefore we will describe the overall process that they all take and then specify which requirements each component helps fulfil. Each component sends a request to the corresponding processing section in the back end when the user accesses the corresponding page, each component then takes in the processed data from its processing section to display to the user.

The search table section shows a table consisting of each email in the collection, we can see from figure 4.1 that the user can access the email details page and the email address page from this section. This feature builds on satisfying requirements FM2 and FM3. The collection details page, also referred to as the graphs page, shows graphs relating to the details of the classification of the entire collection, this feature builds on satisfying requirement FM5, The email data page shows the full details of the specified email, the user also has the option to highlight the key features of the email. This feature builds on satisfying requirements FS1 and FS3. Finally, the email address page shows details of the specified email address, this feature satisfies requirement FM4.

4.2 Back end

The back end will handle all of the processing of the data and requests from the user. There are three main components to the back end, the pre-processing, the classifier and the processing. We will talk about each section and how they communicate with the other sections in the back end and of the system, then we will state the requirements each section helps satisfy.

The first section is the pre-processing, when the user makes a request to upload a collection for classification the user should select from their storage where the collection is based. The collection will then get passed into formatting which will break the emails down into simpler structures for the classifier to handle, the collection is then split into a training set and the whole collection to be passed into the classifier section.

The classifier section handles the sensitivity classification and the topic modelling, the training section takes in the training dataset from pre-processing to train the classifier. The predicting section then takes in this trained model and the entire pre-processed collection to carry out the sensitivity prediction. The topic modelling does not require a training model therefore it only takes in the pre-processed collection to assign the emails into topics. The predictions from both classifiers are then uploaded to the database. The pre-processing and classifier sections of the back end both satisfy the requirements FM1 and FS4.

The processing section of the back end will handle all of the requests from the user that needs to access the database. All of the components in this section follow the same processing, therefore we will describe the overall process that all of the components take. The user requests will be processed and a query will be sent to the database, the data from the database is then received back into the processing component for further processing before being sent back to the corresponding front end component.

4.3 Database

The database will store all of the predictions and other data about the email collection. The database will receive queries from the back end processing and return the corresponding data to the back end processing.

4.4 User Interface

The User Interface is what the user will see, the structure of the web applications front end. We will be discussing some of the design choices that were made and how each feature of the user interface will match the the architecture diagram seen in figure 4.1.

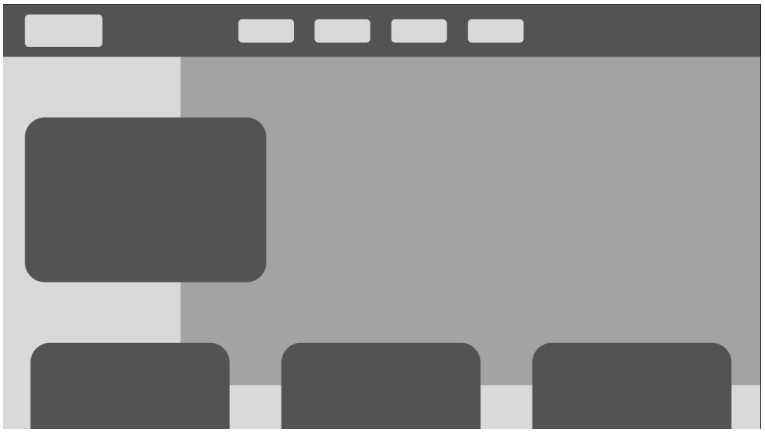


Figure 4.2: *Home page wireframe*

Figure 4.2 shows a wireframe, a simple design of what a product might look like, of the home page. The web applications target audience are adults with minor experience in computer science or usage, this is due to the fact that anyone that is uploading a collection to the public will already have prior knowledge or experience in sensitivity reviews and data handling, as seen in the user stores in the previous section. The colour scheme should be professional and not overwhelming, our application possible deals with very sensitive data and therefore the user interface should be very subtle and in no way seem like it trying to be humorous. The home page will contain basic information on what the web application is and basic instructions for each page or feature. A large background image will help connect all of the sections of the page together and help fill in any black space. A navigation bar along the top with provide the user with an easy way to navigate through the web application, this should stay the same on every page to keep the consistency of the application intact, satisfying requirement NM2.

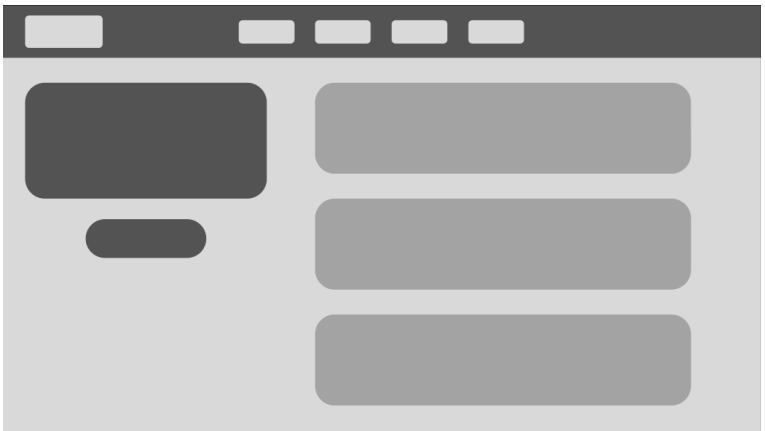


Figure 4.3: *Upload page wireframe*

The upload page, figure 4.3, provides the user to select their email collection from their storage for the classifier to run. This page will show basic status reports on the classification process

to give the user some feedback while the classifier is running, classifiers can take a significant amount of time to run therefore the user should have some way to know that it is working as intended. Again the navigation bar is the exact same as the home page, this will stay the same on every other page, satisfying requirement NM1.

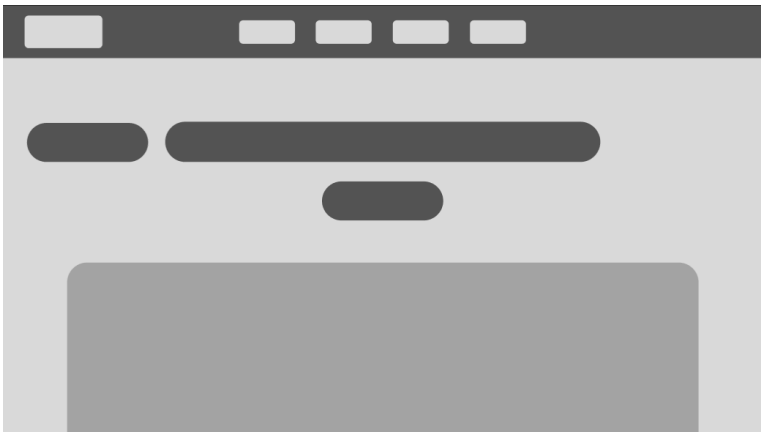


Figure 4.4: *Search page wireframe*

The search page, figure 4.4, will show the user a table containing each email in the collection. This tables columns will be 'To', 'From', 'Body', 'Prediction', and 'Topic'. , satisfying requirement FM2. There will be a filter above the table to allow the user to filter by each column, satisfying FM3. The user will be able to select an email to view its detail page, they will also be able to select who the email was sent from to view more details about that specific email address. Ideally the user could select who an email was sent to as well to view the details from there but some emails are sent to multiple people therefore trying to select a specific person from this column would be difficult.

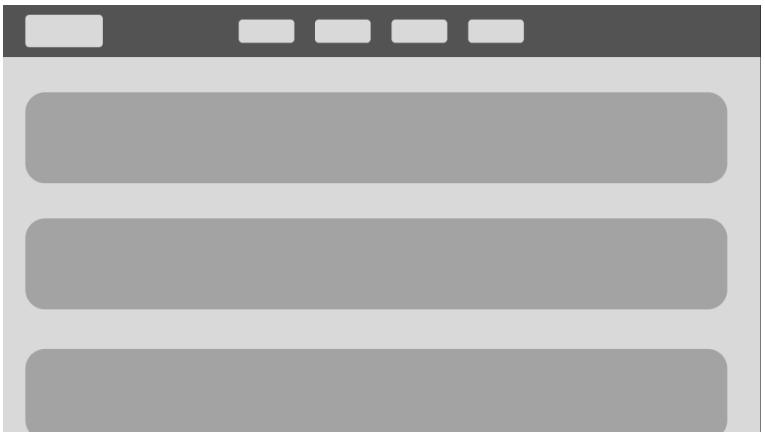


Figure 4.5: *Email detail page wireframe*

The email details, figure 4.5, page will show the contents of a specific email chosen from the search table, satisfying requirement FS1. From here a user can see the keywords, sometimes referred to as features, that the classifier found to signify sensitivity, satisfying requirement FS3. They will also be able to see the keywords that assigned the email to its designated topic. At the bottom of the page, not shown in the wireframe, the user can overwrite the classifiers prediction of the email by changing if the email is sensitive or not, there will also be an input box so the user can provide their reasoning for classifying the document manually, satisfying requirement FS2.

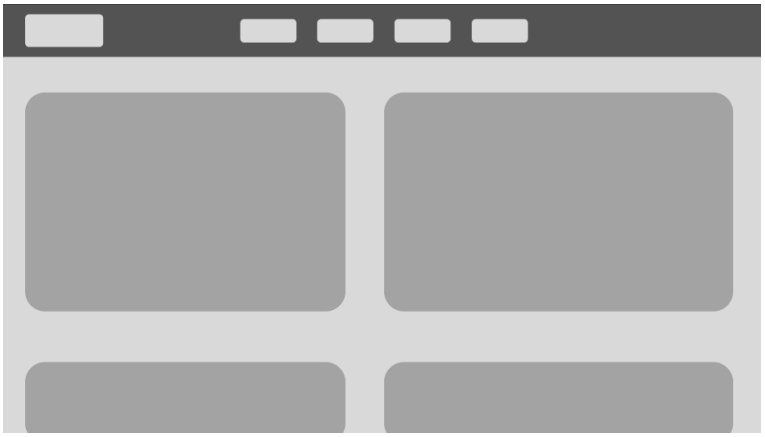


Figure 4.6: *Email address detail page wireframe*

The email address detail page, figure 4.6, will provide some statistics on the specified email address, satisfying requirement FM4. Their will be graphs showing how many sensitive and non sensitive the address has sent and received. There will also be tables showing who they have sent sensitive emails to and who they have received sensitive emails from.

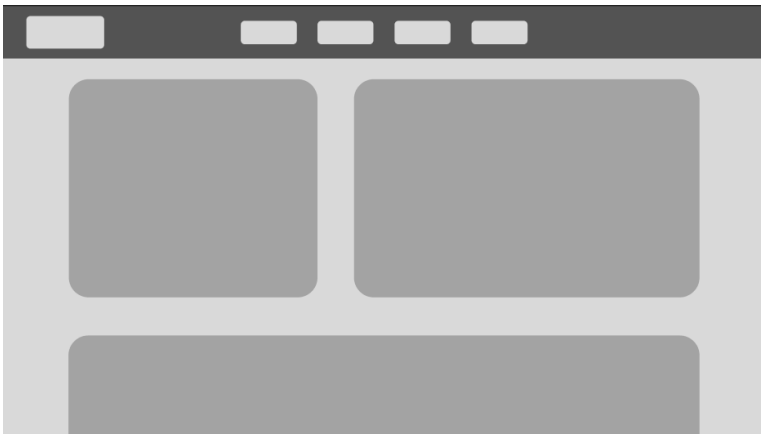


Figure 4.7: *Graph page wireframe*

The graph page, figure 4.7, will show the classification details of the whole collection, satisfying requirement FM5. This will show how many sensitive and non-sensitive emails were found throughout the collection, there will also be a table showing how many sensitive emails each email address has sent. There will also be a table of the most sensitive words that the classifier found throughout the whole collection, satisfying requirement FC1. A similar table will be implemented showing the top words for each topic that is found in the collection, the user will be able to select which topic they want to see the top words for, satisfying requirement FC2.

5 Implementation

In this section we will discuss the options considered for each main feature of the web application and then discuss why the decisions to use certain methods and tools were made. This will include all of points spoken about in the design chapter and some extra features that were crucial in the application being more efficient and user friendly. We will also discuss the software development process and the tools that were used to aid in the development of the system.

5.1 Collection

Just before the collapse of the Enron Corporation in 2001, a database of over 600,000 emails from its employees was released to the public for research. This collection of emails contained personal information from most of the employees, a lot of the highly sensitive information was filtered out before release, which means it is one of the few public collections of real emails that can be used by the public for research. Wikipedia (2022). This the email collection that was used during development of the web application.

5.2 Software development process

This section will describe the different software development practises and tools that were used to build the system. We will discuss the software development methodology that was taken and why and the version control systems we implemented to help control the development of the system.

Software development process: This project followed an incremental development approach. Incremental development is the process of breaking the system down into separate parts known as increments. Each increment is worked on individually with each increment being fully completed before moving onto the next part of the system. This was an easy and logical decision to make since, as seen in figure 4.1, we can see that the system is easily broken down into separate increments and can be fully integrated individually without relying too heavily on other parts of the system. While there were no strict time constraint set on each increment, a weekly report was made to document what had been completed, any issues faced during that week and a plan for the next weeks development. A time sheet was also continuously updated to specify the corresponding hours that each development process took.

Version control system: A version control system is a tool used to constantly update the current system and its progress, this is used to update any changes made to the system to keep a fully up to date version available at all times. Gitlab was chosen to be the version control system for this project. While Github and Gitlab both offer very similar services, Gitlab provides a continuous integrating tool built into its system. This continuous integration system allows us to constantly push the changes to our system directly onto the main system that is stored in the Gitlab repository. Github does provide compatibility of these systems but for convenience Gitlab was chosen.

5.3 Framework

The Framework of the web application was a very important choice as it would lay the foundations to the rest of the implementations to come. A web application framework is a software tool that is developed to aid in building web applications, they provide a way to structure and run your own website without needing to make your own files to run servers or store data etc. There were two options that were considered, Flask and Django, we will discuss all of the positives and negatives for each of these frameworks and then conclude on why Django was chosen.

Flask: Flask is a very popular python framework and as stated by Makai (2022) is a Python we framework built with a small core and easy-to-extend philosophy. Small core means that it is not a bulky framework, it does not require a lot of storage space and it is not difficult for a system to run, and an easy-to-extend framework does not require extensive knowledge of frameworks and web development, it has a simple folder structure and it is very simple to get a web application going from instillation.

Alongside Flask's simplistic approach to web development there are a few more benefits to it. Flask provides a built-in debugger, this is extremely useful as it makes development much smoother.If there is any error Flask will highlight it and provide an interactive trace-back in the browser, providing the user with an instant answer to their problems and removing the need to search through the code manually for the error. Flask also does not rely on any external libraries, allowing the structure to be lightweight and simple to manage. It also provides the use of templates, HTML pages that can be reused, to allow for consistent and easy structuring of the user interface.

With the benefits of Flask's simplicity there also comes some negative features. The main one being it is not suitable for larger applications, it is very capable of handling larger applications but due to its simplicity and minimalism in structure it means that the more you want to build and implement into the application, you may have to create your own base structures for certain features. Another massive downside to Flask is that there is no build in database support, it can implement databases but it requires you to manually link the application to an external database and there is no built in modules for handling queries to the database. This is a major flaw for Flask in consideration of this project as the main structure of the application relies heavily on a well structured and easy to use database.

Django: Django can be considered the best framework as its development speed is fast and easy, InterviewBit (2022), a high-level Python web framework that encourages rapid development and clean, pragmatic design, Django Software Foundation (a). Rapid development is a software development approach that suits short term projects that have frequent updates and new requirements constantly being added. The clean design refers to the easy setup and simple design which allows for easy to follow tutorials, this leads to a fast deployment of an initial application and an easy to understand structure throughout development.

Django gains its status as the most popular framework for a few reasons, mainly due to the size of the team behind it. The developers of Django have created a framework that has been used for some major companies such as Instagram, Mozilla and Pinterest, this has led to a large community that have spent time learning all that Django has to offer and in turn have given their knowledge to new developers. There are countless numbers of tutorials and forums for all sorts of questions to do with Django, which allows for developers to debug and find solutions very quickly. Django also supports many different databases, this is extremely helpful in setting up and managing databases straight from instillation. Queries are automatically handled which makes accessing and modifying data simple and quick to do, there is also an admin feature which allows the developer to access a built in user interface for the database, allowing for easy modification to models and data.

There are a few negatives that come with Django, the main one being a relatively steep learning curve. Compared to Flask's simple instillation and setup, Django does require some knowledge to properly setup your application. There are changes you have to make to the settings of the framework, creating authorised users to be able to access admin pages and due to a large variety in configurations it can be daunting to those without proper training.

Why Django was chosen: As seen in the design stages of the project, the database is a massive requirement. With access to an 'out of the box' database management tool, the project will run smoother and the time that would have gone into installing and setting up an external database can instead go into other aspects of developing the application. Although Django is considered to have a higher learning curve that Flask, I have developed applications in Django in the past and therefore it was an easier decision to make since I would not need to spend time learning how to use Flask. Django was the obvious choice for this project and provided a solid foundation to the application.

5.4 Front end

The front end houses all of the processes and parts of the system that the user directly interacts with. We will discuss the decision made in the programming languages used for the user interface and any external tools needed to provide certain features to the user.

Templates: Django supports HTML templates for implementing a user interface in a web application, these templates are HTML based pages that allow for the Django framework to send data to and from the front and back ends of the system. A template contains variables, which get replaced with values when the template is evaluated, and tags, which control the logic of the template, (Django Software Foundation b). This is extremely useful when creating web applications since we do not need to manually configure for data to be sent using certain data transport methods, Django does everything for us automatically.

Styling When it came to the design of the user interface, CSS was implemented. CSS is the language we use to style an HTML document, W3Schools. As stated, CSS is the most common language for styling HTML based web applications, it can be used to describe how any HTML element should be displayed for the user. Due to its simplicity and wide range of application, all of the designing of the web application is done using CSS.

Search table: HTML does support simple table creation but when including filters and a large database if items, there are tools made by other developers that we can implement. There is an extension called django_tables2 that helped solve this problem. Support for automatic table generation based on a Django model, django tables2, a Django model is the structure of table in the database. The table can then be edited to remove certain columns or provide extra functionality such as being able to click on cells of the table and be re-directed to another page.

Graphs: HTML and CSS do not have an explicit feature to create graphs for a web page, what did use however was chart.js. A free JavaScript library for making HTML-based charts. It is one of the simplest visualization libraries for JavaScript, cha. chart.js provides a very simple way to display graphs in a HTML web page, another benefit is that we can pass the data straight from the database into these charts. Due to these facts, chart.js was used to create all graphs shown in the web application.

5.5 Database

The database is one if the main blocks of the systems as stated in the architecture section earlier. While most database systems that Django supports function very similarly and for a small project like this, some systems that other developers claim are superior do not get to provide their benefits

to our application as much as they would a large scale development. We will briefly discuss three different database systems, SQLite, MySQL and PostgreSQL. Similarly to the framework discussion above, and what will remain the structure throughout this whole chapter, we will go through the positives and negatives of each system and then conclude on our decision.

SQLite: SQLite is the most used database engine in the world, that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine, (SQLite 2023). SQLite is Django's default database system as it is simple to use and is great for small projects.

SQLite is called a server-less database system, this means it does not rely on a server when integrated into a system, but instead uses the storage on the users device when the web application is running. This is useful for small projects that will not be released online or require multiple users to be accessing, or editing the database at once.

A downside to SQLite is also that it is a sever-less system, if a developer is trying to use it on a large scale project that handles a large amount of data, or requires users to access the same database at once then SQLite cannot handle these requirements. There are also storage capacity concerns and this relies fully on how much storage space the user has on their device, it cannot support as much as client/server RDBMSs like MySQL or PostgreSQL, (ostezer and Drake 2022).

MySQL: MySQL is a widely used relational database management system, ideal for both small and large applications, (W3schools). As stated MySQL is a very popular database system that can be fitted into a large variety of projects.

MySQL is the opposite of SQLite as it is a server based database management system, this means it relies on an external server to handle requests. This is very useful as it allows for multiple users to update the data that is stored and allow for more requests to be processed at once. A server side database also has access to a larger storage size as it is not restricted by the users storage capacity.

Although MySQL is extremely popular, it is also regarded as the most different from standard SQL, (ostezer and Drake 2022). This refers to the fact that the developers of MySQL traded user satisfaction and productivity for certain SQL features that other database systems like SQLite may have access to. This may reveal problems later on as you may need to use an SQL feature that you thought was standard but in fact MySQL does not support it.

PostgreSQL: PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance, (PostgreSQL).

Postgres, like MySQL, is a server side database system, meaning it can handle a lot of requests, and updates to the database at once. PostgreSQL is compatible with a wide array of programming languages and platforms, (ostezer and Drake 2022). This means that no matter what other systems you may already have in place, you are able to integrate a Postgres database system easily into the project.

Postgres is susceptible to memory problems, Each new process is allocated about 10MB of memory, which can add up quickly for databases with lots of connections, (ostezer and Drake 2022). This means that with a large amount of users the system could slow down and become a hindrance in the projects usability.

Why SQLite was chosen: Due to the simplicity of this project and that a user only needs access to their own database, SQLite fits into our system perfectly. This choice also saves time in setting up the database system as it is already integrated into Django when the framework was setup. Although there are storage size concerns, the Enron collection is not particularly large in terms of data collections therefore the storage capacity of the device the system is being used on is not a concern.

5.6 Back end

The back end handles all of the processes required to deliver to the correct data to the front end to be displayed to the user. The back end as explained in the previous chapter consists of three sections, pre-processing, classifier and the processing. Each section's implemented components will be explained to show the manipulation and flow of the data being processed.

5.6.1 Pre-processing

The pre-processing of the data is crucial into how well the classifier is able to identify key features in each email, and can help drastically in the efficiency of the collection processing system. pre-processing refers to how the data is prepped for the classifier, there are many changes that can be made to the data such as removing stop-words, lemmatising and many other processes. We will look into each process and why it is useful in our system.

Tokenization: Tokenization is the process of splitting up the text that the classifier will use. This process will take in the chosen data and convert it into a list containing each individual word, this is used as the foundation of building a phrase or paragraph with individual tokens being put together. The classifier handles predictions in a similar way, it looks at the building blocks of a phrase or paragraph.

Normalization: Text normalization is the process of breaking down words into their most basic form, for example 'running' becomes 'run'. This is extremely useful for a classifier as it reduces the size of the vocabulary, decreasing runtime and efficiency. We will look at two type of normalization, Stemming and Lemmatizing.

Stemming is a very basic way of normalization, it will take similar words and reduce them down to the same token. The main downside of stemming is that it can reduce words down into meaningless tokens such as 'watch', 'watching' and 'watcher' can be reduced down to 'wat', this is fine for the classifier to use but when reviewing the key tokens they may not make sense.

Lemmatizing removes this downside completely, when reducing words down it will only make them into real words. A lemmatizer will make 'watch', 'watching' and 'watcher' will all become 'watch'. This does come with a downside of runtimes, it takes more time to find real words rather than being able to make fake words. Due to our application having the feature to review the key features in an email, it is important that the words shown make sense to the user, therefore lemmatization was used for our system.

Data splitting: For a supervised classifier a training and testing dataset is needed, the training dataset allows the classifier to look at the patterns in sensitive and non-sensitive in documents while the testing set is used to classify the documents and then check the results. To create a non bias system a random split of the collection is useful as it allows the classifier to train for a large variety of documents. This random split was used with a specified random state to give the same results during testing, this random would be removed in deployment.

Vectorization: Text vectorization is the process of changing words into numbers, this is mainly done to speed up the process of classification as it is easier the computers to understand and process number than words and letters. There are multiple ways to do this with each one having benefits and drawbacks, we will take a look at one-hot encoding, bag of words and TF-IDF vectorization.

One hot encoding is a process that turns each word into a column of a vector, then each word is turned into a vector where the corresponding column in the vector is a 1 with the rest of the vector being filled with 0's. This is a very simple encoding process but also a costly one as each vector is the length of every unique word in our collection.

Bag of words is a slightly more advanced version of one hot encoding as instead of storing each word as a vector, it instead combines all of the vectors in a line of text into one vector, the columns till represent individual words but instead each word in a line of text will be represented with a 1 and each word not in the line is a 0. This is again extremely costly as the vector is still the size of the entire vocabulary but it is overall more effective and efficient that one hot encoding.

Finally we will talk about TF-IDF vectorization, term frequency - inverse document frequency. The term frequency is how often the particular word being looked at appears in the current document while the inverse document frequency calculates how common the word or phrase is in the whole collection. When you combine these factors together you get how important the word is in terms of the current document. The vector that is produced is a list of importance values for each word in the document. This is extremely efficient compared to the previous methods and overall gives a more reliable outcome. Therefore TF-IDF vectorization was chosen for this project to get the best possible results from the classifier.

N-grams: N-grams refer to the number of words a classifier will use at once to determine importance, in our case sensitivity. A unigram refers to a singular word, only one word will be used at a time when checking importance and therefore no context is taken into account when classifying a document. A bigram refers to two words being used together, these words are next to each other in a document and may provide some context for the classifier to determining them as more or less sensitive. Most classifiers give the option to consider both, this method is better as by forcing it only to look at bigrams an important token may be combined with a non-important token therefore bringing the overall importance down, while if it had the option of unigrams it could consider the individual tokens importance alone, this is the method that was used in our system.

Stopwords: Stopwords are common words in the English language that do not provide any extra meaning to a phrase. For example the most common stopwords are a, an, the, and, it and for, (Fontanella 2020). Since these words do not provide any importance in sensitivity to a document, they will only take up more time and processing power. Therefore we removed these stopwords from our documents when classifying them.

5.6.2 Classifier

The classifier is the main component of our system, therefore depending on which classifying method chosen may greatly impact the efficiency and effectiveness of the entire web application. To many incorrect predictions would result in the web application not being reliable enough to prove that machine learning can help towards the classification of documents, yet a method that is more in depth may take a very long time to execute, which would fail requirement NM4. Scikit-learn is python library that contains tools for machine learning, while it is possible to build classifiers manually, scikit-learn provides a large variety of classifying methods for developers to import and use for their own systems. We will be taking advantage of these classifying methods provided by scikit-learn, specifically looking into three different kinds, Naive-Bayes, Support Vector classifier (SVC) and Random Forest.

Each classifying method will be explained, they were all executed on the Enron collection to give their performance metrics. The classifiers will be compared based on these metrics to give reasons to why the final method was chosen.

Naive-Bayes: Naive-Bayes uses a probabilistic approach to classifying documents, this means a document is classified by the probability that each individual feature in the document is sensitive. To go into detail, this classifying method make use of Bayes' theorem. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred, (GeeksforGeeks 2023). the naive Bayes is a supervised classifying method therefore is uses a training set, from this set it calculates the probability that each token is sensitive, Bayes'

theorem calculates this over a whole document and not individual tokens. The naive part of the name come from the fact that instead of using the probability of a whole document, in Bayes' theorem, it instead gets the probability of each token and multiplies them together, producing a probability for the whole document being sensitive.

Support Vector Classifier: Support Vector classification, as suggested by the name, uses vectors to calculate the best possible split of data. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points, (Gandhi 2018). SVC is also a supervised classifying method, it uses the training set to try and find the best fitting vector that splits the already classified documents in the best way possible.

Random Forest: Random forest, as with the other classifying methods, is a supervised classifier method. Instead of looking at probabilities or fitting a vector to split the classes, it builds decision trees to make its predictions. In a random forest classification, multiple decision trees are created using different random subsets of the data and features, (Shafi 2023). Random forest works off of a most popular opinion design, the trees can be considered as people who have their own opinions, when shown a document these trees make a prediction based off of their opinion and the most popular prediction is chosen as final.

Performance metric comparisons: Classifiers can be measured on their performance by using a testing set of documents and then calculating certain metrics such as precision, recall, f1 and f2.

Precision refers to the quality of a positive prediction made by the model, (C3.ai). This is measured by taking the number of true positives, the number of correctly identified sensitive documents, and diving it by the total number of sensitive documents we predicted.

Recall is slightly different to precision, instead recall is the measurement of out of the documents that are sensitive, how many did we correctly predict. There is a trade off between precision and recall, to make a system more precise you may predict less documents and rely on a small number of correctly predicted documents, but to gain more recall you need to make more predictions to try and cover all of the sensitive documents in the collection, the more predictions made the higher the chance of getting wrong predictions, therefore losing precision.

This is where the F-score metric comes in, F-Measure or F-Score provides a way to combine both precision and recall into a single measure that captures both properties, (Brownlee 2020). Instead of trying to maximise precision or recall you can try to maximise this metric instead to improve the classifier overall, instead of in a singular category. We can however change the balance of precision and recall in the f-score metric, F1 and F2. F1 is a perfect balance between precision and recall, but F2 puts more weight on recall, this is very helpful as for this system we want to be able to predict as many of the sensitive documents as we can.

	Precision	Recall	F1	F2
Random Forest	0.8333	0.3846	0.5263	0.4310
Support Vector Classifier	0.7895	0.2884	0.4225	0.3304
Naive-Bayes	0.6774	0.4038	0.5060	0.4393

Table 5.1: *Performance metrics*

Table 5.1 shows all of the metrics stated above for each classifier we have discussed. We can see from this table that random forest and naive-Bayes perform significantly better than support vector classifier, this is most likely due to sensitivity being a broad term as discussed earlier, and a vector to separate sensitive from non-sensitive would likely have a very small margin for error.

We can see that naive-Bayes has the highest F2 metric but only by a small margin, but when comparing the precision of random forest and naive-Bayes, random forest trumps naive-Bayes by

a huge amount. Due to this factor Random forest was chosen for our system, it is a very strong classifier when compared against others as seen in table 5.1 and will provide a great foundation to build the web application around.

5.6.3 Topic Modelling

Topic modelling is another form of classification, but instead of specifying what groups to sort documents into, the classifier will instead group together similar documents into clusters which can then be labelled by the developer. Topic modelling is also an unsupervised form of classification, meaning it does not need to train on a data set. Just like the classifiers stated earlier there are multiple ways in which this process can be achieved, we will look at Latent semantic analysis (LSA) and Latent Dirichlet analysis (LDA).

Latent Semantic analysis: Latent semantic analysis attempts to find hidden topics in documents without specifying exactly what that topic is. When you read many texts, themes begin to emerge, even if they're never stated explicitly, (Ioana 2020). We can read a document and find the topic of it just from how the sentences are structured, LSA does the same thing but by using maths. In basic terms the LSA classifier breaks down a document into more basic meanings for each token, and tries to find significance in certain tokens being together.

Latent Dirichlet analysis: Latent Dirichlet analysis provides the same results as LSA but it takes a different approach. LDA generates probabilities for the words using which the topics are formed and eventually the topics are classified into documents, (Seth 2021). LDA breaks down the collection into two tables, a table of possible topics in each document, and a table of the tokens that a topic contains. Using these tables the classifier calculates the probability that a topic belongs to a document, giving a list of probabilities per document for each topic that may exist in it, with the highest probability being the predicted topic.

Why LDA was chosen: While I could not find specific reasons as to why LDA is more popular and overall a better topic modelling method than LSA, Garbhapu and Bodapati (2020) stated that from their research and experiments, LDA achieves 60 to 75% superior performance when compared to LSA using document similarity within-corpus. Based off of these findings LDA was chosen for the topic modelling classifier system in this project.

6 Final Product

In this section we will take a look at the final web application, we will show of each of the major features and relate back to the requirements each of them satisfies.

Navigation:

Figure 6.1: *Navigation bar*

Figure 6.1 shows the navigation bar that sits at the top of every page. its simple design and ease of use allows for easy navigation and consistency throughout the entire web application. This feature satisfies requirements NM1 and NM2.

Upload:

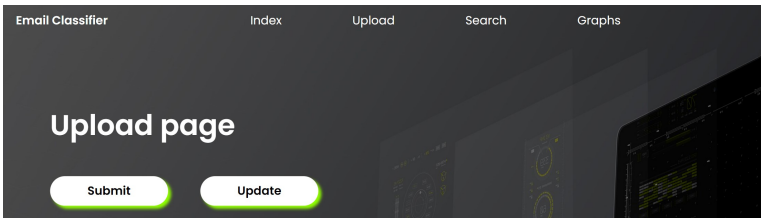


Figure 6.2: *Upload Feature*

Figure 6.2 shows the upload feature on the web application. This feature went through a few changes from the original design. In a full deployment of the system, a user would be able to upload their own collection for classification, but due to this project only using the Enron collection, we do not need the functionality to choose a collection from storage from the web application. There were also issues in getting status updates from the running classifier onto the web page therefore there are no status updates when running the system. What this upload page does instead is when 'Submit' is clicked, the classifier will locate the Enron collection from storage and classify it, storing the results in the user storage, then when 'Upload' is clicked the system takes the results from the classifier and uploads it into the system's database. In deployment, this would all happen at the same time but due to testing and debugging, they are handled separately. This feature satisfies requirements FM1 and FS3.

Search table:

Figure 6.3 shows the search page, this page consists of a table that stretches the whole length of the page, showing 25 emails, with a pagination bar at the bottom, for convenience in showing this feature the figure has been edited to shorten the page. A pagination bar shows different pages of the table due to a large number of emails, you are able to navigate through these pages from the bar. The filter above the table allows the user to filter by every column in the table, the user is able to filter by multiple columns at once and the corresponding results will be shown in

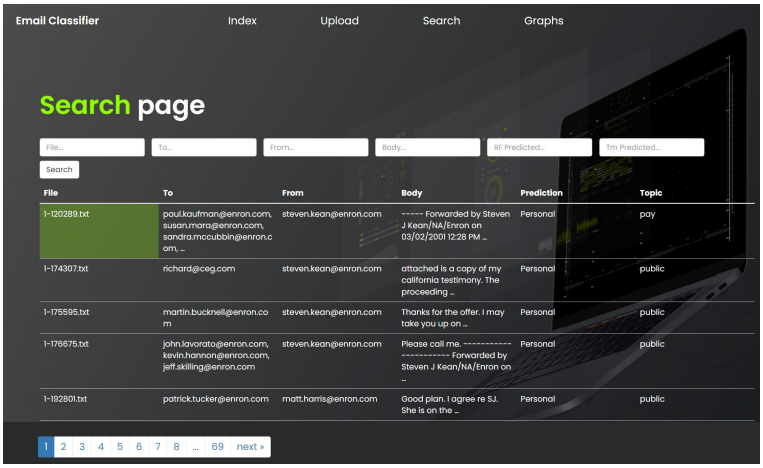


Figure 6.3: *Search table and filter*

the table below. As shown in figure 6.3 you can see one of the cells of the table is highlighted, this is where the cursor shows that you are able to click on this cell. Clicking on the filename of the email will re-direct you to the email details page, and clicking on a 'From' column cell will re-direct you to the email address details page. As shown in the table column 'Prediction' each email has been classified as 'Personal' or 'Non-personal', you can see the same in the 'Topic' column that each email has been assigned to a topic, there are 10 topics in total.

Email Address page:

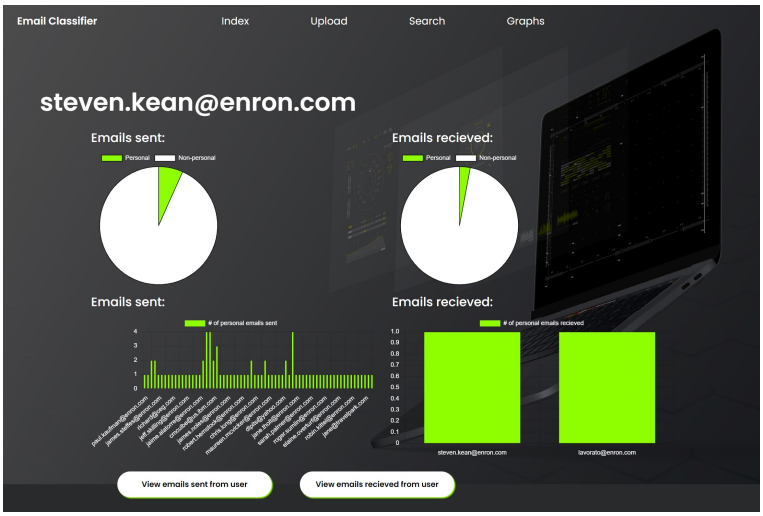


Figure 6.4: *Email Address page*

Figure 6.4 shows the email address details page, this page shows classification detail for a specified email address. The page allows the user to see how many sensitive and non-sensitive emails an address has sent or received. The page also shows who the email address has sent sensitive emails to and who the address has received sensitive emails from. There is also an extra feature at the bottom of the page that allows the user to filter the table by emails sent or received from the specified address. This page satisfies requirement FM4.

Email detail page:

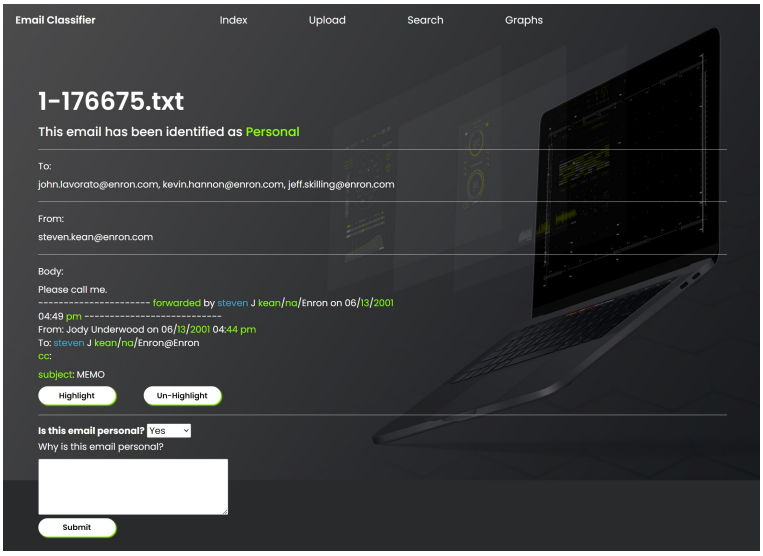


Figure 6.5: *Email details page*

Figure 6.5 shows the email details page, this page consists of every aspect of the email, 'To', 'From' and 'Body'. At the top of the page, you can see that the classifier's prediction is highlighted in green, this is to make it stand out as it is the main functionality of the entire web application. This page satisfies requirement FM4.

There are buttons below the 'Body' section, these buttons will highlight and un-highlight the key features that the classifier found to be sensitive. You can see this highlighting throughout the 'Body' section with green being features that the classifier found sensitive and blue being features that assigned the email to a topic. This feature satisfies requirement FS3.

Below the highlighting feature there is a form that the user can fill out, this form allows the user to overwrite the classifiers prediction of a document. The user can choose between sensitive and non-sensitive and then give a reason as to why they have manually classifier this document. When a manual review is given it will display the review on this email details page. This feature satisfies requirement FS2.

Collection overview: The collection overview page, referred to as the graph page in the web application, shows the overall details of classification over the Enron collection. It provides four graphs that each show a different section of the classifiers predictions.

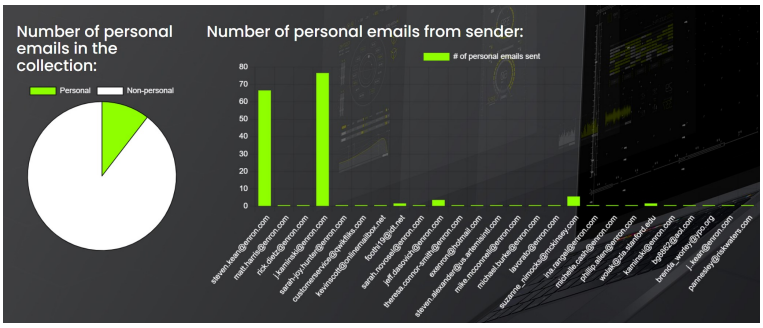


Figure 6.6: *Collection overview graphs*

Figure 6.6 shows two graphs which show the overall classification of the collection. The pie chart shows how many emails in the entire collection have been classifies as sensitive or non-sensitive, when you hover over the section the number of emails will appear. The bar chart shows how many sensitive emails an address has sent, with the addresses along the x axis and the number of sensitive emails sent along the y axis. This feature satisfies requirement FM5.

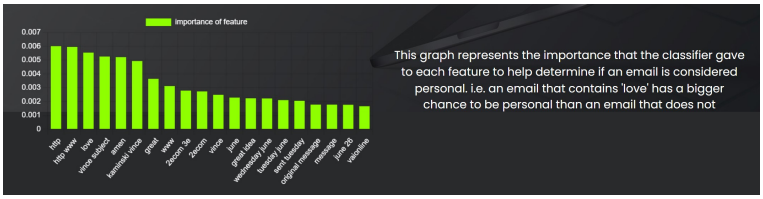


Figure 6.7: *Sensitive features graph*

Figure 6.7 shows a graph that gives the top 10 features that the classifier determined as sensitive. This provides the user with an overall understanding of how the classifier has distinguished sensitive terms from non sensitive. Most of the features given do not make sense in terms of sensitivity, such as 'http' and 'june', this may come down to not filtering out stopwords enough or there may be a pattern between links and sensitivity in the Enron collection. This graph satisfies requirement FC1.

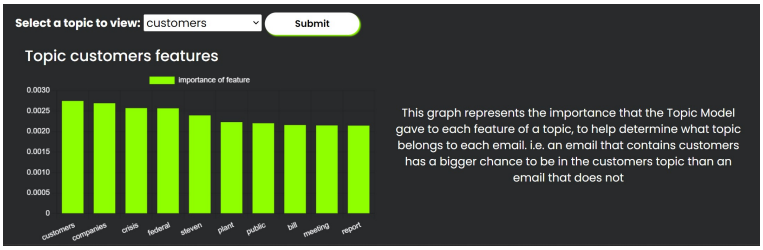


Figure 6.8: *Topic features graph*

Figure 6.8 shows a graph that gives the top 10 features that assign a document to a certain topic. There is a filter the user can apply to view different topic's top words, from there the table will show the top features that fit into the specified topic. This allows the user to get an understanding of the difference in topics and how each email has been assigned one. This satisfies requirement FC2.

7 Evaluation

This chapter will go over how well our system performed in the task we are trying to solve. We have unit tests which verify that the code behind each part of the system is working as intended. A user study was also carried out to test how new users approach the systems and to identify any issues they find or new features that were not thought of before development.

7.1 Unit Testing

Unit tests are pieces of code placed throughout the system that verify that everything is working correctly. (Coverage.py) is a python testing tool that helps to view the test coverage over an entire system, it provides statistics about which parts of the system are covered by testing, allowing for easier implementations of where unit tests should be considered.

Module ↑	statements	missing	excluded	coverage
Classifier_website__init__.py	0	0	0	100%
Classifier_website\settings.py	18	0	0	100%
Classifier_website\urls.py	4	0	0	100%
manage.py	12	2	0	83%
personal_classifier__init__.py	0	0	0	100%
personal_classifier\admin.py	6	0	0	100%
personal_classifier\apps.py	4	0	0	100%
personal_classifier\filters.py	20	4	0	80%
personal_classifier\migrations__init__.py	0	0	0	100%
personal_classifier\migrations\0001_initial.py	5	0	0	100%
personal_classifier\models.py	29	0	0	100%
personal_classifier\python_files__init__.py	0	0	0	100%
personal_classifier\python_files\enron_classifier.py	149	0	0	100%
personal_classifier\tables.py	14	0	0	100%
personal_classifier\tests.py	46	2	0	96%
personal_classifier\urls.py	4	0	0	100%
personal_classifier\views.py	136	79	0	42%
Total	**447**	**87**	**0**	**81%**

Figure 7.1: *Coverage of unit testing*

Figure 7.1 shows the outputted web page from coverage, we can see from this figure, each part of the system and how much they are covered by unit testing. The statements column represents how many parts of that file can be covered by unit testing, with missing showing how many of these parts are not covered. We then get a total coverage of each file with a total coverage percentage at the bottom right of the table. We can see that our system is 81% covered, this is great coverage considering the struggles of testing the front end and their corresponding processing components in the back end.

Most of the processing section in the back end of the system would require implementing a Selenium web driver. A Selenium driver can simulate a user on the running system, being able to click buttons and use features of the web application. Due to time constraints and the resources required to implement this driver, a decision was made to not implement these tests. We can see this reflected in the coverage of the views.py file which handles the processing to and from each page of the application.

To give a brief overview of the larger parts of the system that were tested, the database models that define the structure of the data being stored in the database were tested to ensure the correct data formats and returned values were accurate. The pre-processing was then tested to make sure the reformatting of the emails was correctly handled before being passed into the classifier for training and predicting. The whole classifying procedure was then tested to make sure no errors were thrown during execution.

7.2 User Study

A user study was carried out on the system to provide a new insight into how an untrained person may confront this system. A user study can help reveal issues missed during development and the users themselves then have the opportunity to give their own feedback and what new features they would like to see implemented into the system.

7.2.1 User Study structure

The User study is comprised of two parts, a feature test where 8 users are split into 2 groups, with each group getting a different version of the system. These versions of the system will have certain features turned on or off, the users were then asked to carry out tasks which tested how relevant each feature was to the system. The second part of the study was a questionnaire that each user was asked to fill out after completing the feature testing, these questions would allow the user to give their input on the system, the design, the usability and any addition they believe would help elevate the system.

Feature testing: With the users split into 2 groups we are able to test how effective a feature is to the system by having 4 users complete a task with the feature active and 4 users complete the same task without the feature. Each user was given a list of tasks to complete on their given version of the system, the time taken to complete each task and the path they took to navigate through the web application to get to their answer were recorded to then be compared against other users.

Version 1 had the emails sent and emails received graphs turned off, removing requirements FS1 and part of FM5. Task 3 tested the relevancy of the sensitive emails sent from an email address, this task had users try and find how many sensitive emails a certain address had sent, where users with the feature could look at the provided graphs to find their answer while users without would have to individually count the amount of emails displayed on the search table after filtering for the specific email address. Tasks 4 and 5 tested the emails received graphs, these tasks had users trying to find how many sensitive emails an address had received and how many non-sensitive email a different address had received. These tasks followed a very similar structure as task 3 where users with the graphs could read out the results from the address details page while users without, again would have to count from the search table.

Version 2 had the highlighting and the topic modelling classification turned off, removing requirements FS3 and FS4. Tasks 1 and 2 tested the highlight feature where users were asked to identify the features in an email that made the classifier allocate it as sensitive for task 1, and do the same for task 2 but instead list the features that the classifier used to cluster it into a topic. Users with the highlight feature could click the button on the email details page and state the

highlighted words in front of them, while users without the feature would have to navigate back and forth from the feature tables on the collection overview page and state the matching words from the table and the body of the email. Task 6 tested the topic modelling classification, this task had the users try and identify which topic an email belonged to out of the list of specified topics. The users with the classification could see the topic stated in the search table, while users without would have to read through the email and match it to the most similar topic.

Questionnaire: As stated earlier the questionnaire gave the users the opportunity to give their feedback on the system, giving an insight into a first time user's opinion on certain aspects of the system. The questionnaire consisted mainly of linear scales where a user could rate each aspect of the system out of 5. The users are asked 7 questions in total, they ask what the user thought of the design, navigation, usability, if they struggled during any of the tasks, any ideas for new features, any features they found irrelevant and finally if they had any additional input towards the system. These questions allowed the user to express how they felt in every aspect of the system and helping towards new possible developments towards the project.

7.2.2 Results

The results for the feature testing will reveal if each feature has an overall benefit to the system, each task will be compared to the user times with and without the feature and from these times we will identify if the feature is relevant. We will then discuss the results from the questionnaire, specifically the overall score the users gave each part of the system and then discuss the positives and negatives that each user experienced.

Feature results:

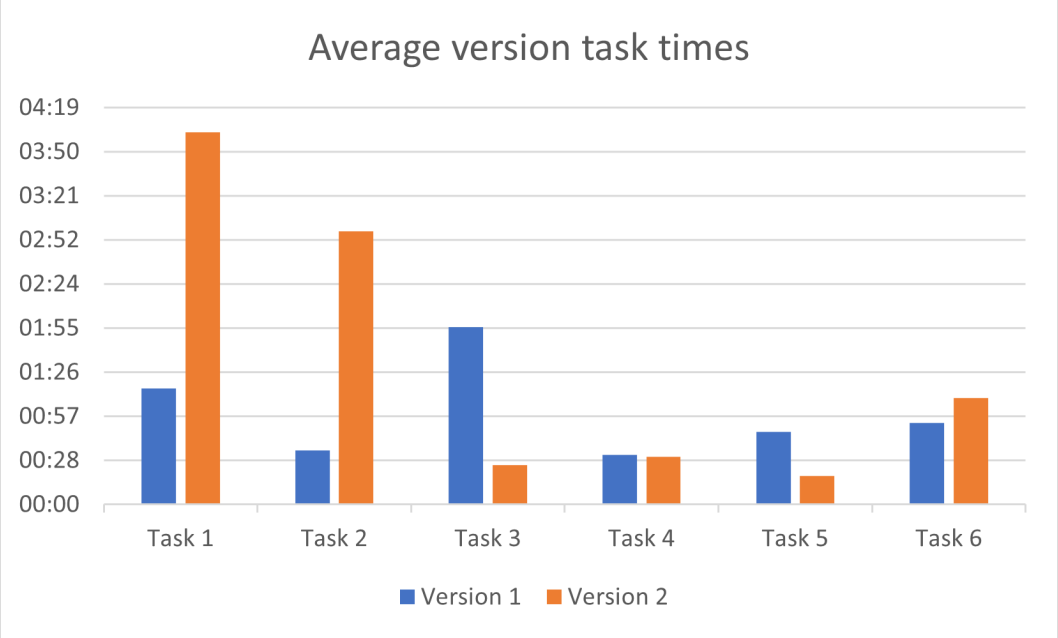
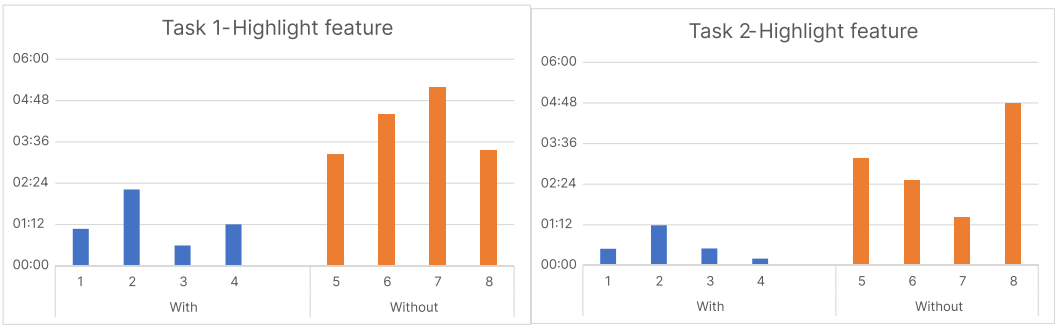


Figure 7.2: *Version task comparison*

Figure 7.2 shows the comparison results between the versions and the average time taken by the users to complete the task in their version. As stated above each task or tasks corresponds to a feature being tested, we will break the graph down into each feature being tested to compare the times with and without the features.



(a) Task 1 *(b) Task 2*

Figure 7.3: *Tasks 1 and 2 - Highlight feature*

Figure 7.3 shows each user's time to complete tasks 1 and 2, they have been split into who did and did not have the highlight feature. As we can clearly see from the charts, the users with the highlight feature were all faster than the users without, this was to be expected as the task pushes for the user to make use of this feature. This helps verify that the highlighting feature is a benefit to our system, satisfying requirement FS3.

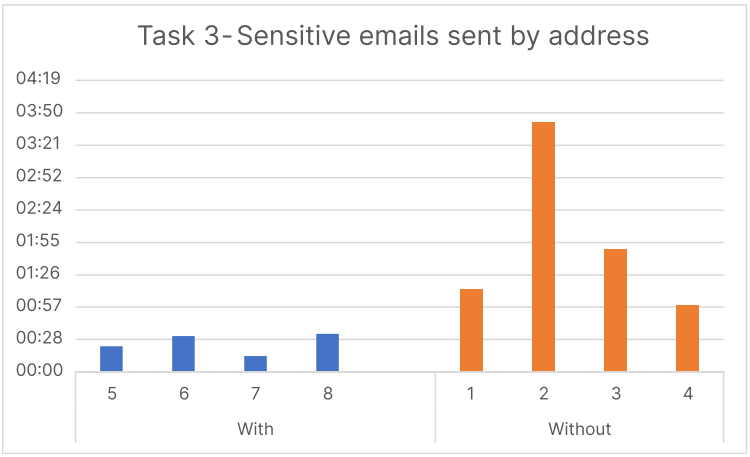
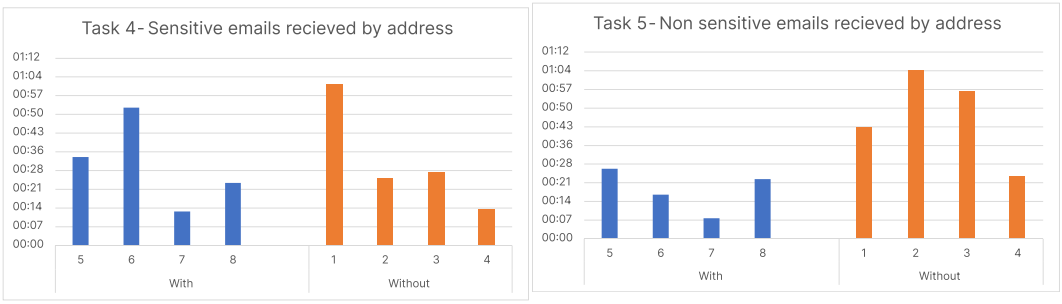


Figure 7.4: *Task 3 - Sensitive emails sent*

Figure 7.4 shows the times taken by each user to complete task 3. Again following the same pattern and as expected, the users with the feature were faster than the users without. We can see that the users without the features times vary by a large amount, this is due to the range of users that were asked to participate. It is safe to assume that the users that took longer may not be so adept in their computer skills. This chart helps verify that the emails sent charts for a specified address is beneficial to our application.

Figure 7.5 shows each user's times to complete tasks 4 and 5. We can see from the charts that these tasks were closer in time comparisons. Task 4 asked the user to identify how many sensitive emails an address had received, due to the address being chosen did not have a large amount of sensitive emails sent to them, the users without the graph did not have to count many emails. Task 5 has a slightly larger gap between completion times, this task asked for the user to identify how many non sensitive emails an address had received. Although the number of emails for the answer was larger than task 4, the user had gained experience in how to carry out this task therefore they knew the exact procedure to follow to get to their answer. While this task does



(a) Task 5 *(b) Task 5*

Figure 7.5: *Tasks 4 and 5 - Email received by address graphs*

not solidify the relevancy of these features they due suggest that the usability of the application is increased between having and not having the features.

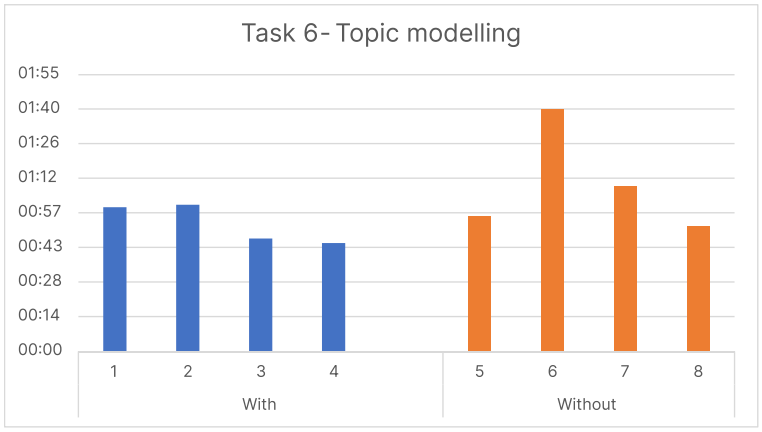


Figure 7.6: *Task 6 - Topic modelling*

Figure 7.6 shows each user's times to complete task 6. This task seemed to throw the users off slightly. The task asked the user to identify which topic a certain email belonged to, with the topic modelling feature the users could look at the search table and read to the corresponding column, while users without would have to identify the topic out of a list. The users with the feature missed the column on the table when looking for the email, they assumed that due to being asked about a specific email they had to look into the details of the email content, which led to the times being closer than expected. This helped reveal an oversight when developing the email detail page, the topic that was assigned to the email was not placed inside this page, which when considered would make perfect sense and should have been thought of during development.

Overall the feature testing proved positive in each features relevancy, while it was expected that the users with the features turned on would perform better, some of the times were close than anticipated. The variety in the users also aided in providing evidence for a features importance, with some users having more computing knowledge they still performed slower without features than inexperienced users with the feature active, as the user experience was balanced between the 2 groups.

Questionnaire results: The questionnaire, as stated earlier, provides the user with the opportunity to express how they felt about their experience during the feature testing. We will discuss the ratings that the user gave to the system and then talk about their feedback and what additions could be implemented in the future.

The overall response about the design of the application was extremely positive with the majority of users giving 5 out of 5. There were also multiple comments about the design in the additional feedback section, with users stating 'The design and layout were really good', 'very good looking website', 'very nice design'. This allows us to state that requirement NM1 has been satisfied.

The navigation was very highly rated by users with the majority giving a 4 out of 5. The navigation is a key aspect of the usability of the application, allowing the user to access any section from anywhere. Some users also stated 'easy to navigate once got to use it' and 'easy to navigate', solidifying that effectiveness of the application navigation in aiding the user to use the system and satisfying requirement NM2.

When the users were asked about what functionality they think could be added to the application, a few stated features that were turned off during their feature testing such as 'highlighting key words' and 'list of features used to classify a certain email next to email'. This was expected as users without certain features struggled more than those with, therefore they would like those features implemented. Some users had some other features they thought would be useful, we will discuss these features and why they would or would not be effective in this system.

Two users stated they would like an email counter at the bottom of the search page so that you know when filtering how many results have been found, I think this is a great additional feature to add, when looking through other website, the majority of them with a search table will state how many results have been found. This would provide an even quicker way to solve tasks 3, 4 and 5 as the user can simply filter the table and state the number of results.

One user suggested that the search table should have memory of the last filter that was executed, they stated this as during their feature testing they had to swap back and forth between the collection overview and the email to identify the key features. This feature would make sense for this scenario but due to the reason being to make this specific task easier, this extra feature seems unnecessary when the highlighting feature is active. A user would have no need to swap back and forth between pages constantly, therefore although this feature may benefit the system slightly I do not believe it is worth the time and resources necessary to execute it.

As identified as a missed feature in the feature testing, a user suggested that the topic of an email could be shown on the email details page. As stated earlier this feature makes perfect sense to be implemented and was a missed requirement for the email detail page. This feature would definitely be implemented in the future as it would build massively on the usability of the system.

One user suggested that the cells in the search table that redirect the user to the specified page should be more clear on where the link is actually taking you. This opens up a whole new discussion about the ability to interact with these cells to access certain pages. While clicking on the email to re-direct the user to the email detail page is logical, having it just be one column with another re directing you to the address details page is confusing. This could be fixed by having a separate search table for searching email addresses, the user could then search up a specific email and click on the result to re-direct them to the address details page. This would suggest creating a new page on the application to hold this table. I think this implementation would benefit the system overall, while it is more time and effort and for users experienced in using the system may be fine with the current version, new users would find this process more logical and clear.

After clarification for one of the responses of any unnecessary features, no users stated any features they interacted with were unnecessary towards completing their tasks. This helps us clarify that all of the features contribute towards the usability and functionally of the system.

8 Conclusion

As the amount of data companies are collection is rising and the movement for this data to be available to the public is growing, the hindrance of manually processing these documents is exponentially expanding. Email collections are especially concerning as, unlike company document, they have no set structure, no need for formal language and are more likely to contain sensitive information.

A new system is needed to aid in the automation of reviewing emails for companies that are required to publish their data to the public. With machine learning becoming more readily available for public development, a system was proposed to implement a machine learning classifier to review these email collections and display the results to the user through a web application.

This system is built around the Enron email collection with the architecture consisting of three main components, the front end handles all of the interactions that the user has with the system, the back end which handles all of the processing of information to and from the front end and the database, and the database which stores all of the results from the classifying process. The Random forest classifier takes in the Enron collection, applying multiple pre-processing techniques and executing a training, and prediction process to then output the results to the database. A topic model classifier takes in the pre-processed data to assign each email to a cluster of similar emails. The user can interact with these email to see why each email was classified as sensitive or not, they can also look into the details of the email addresses associated in the Enron collection, being able to view who has sent the most sensitive emails and who they are communicating these emails with.

Two types of evaluation were carried out on the fully developed system, unit testing and a user study. The unit testing checked that every part of the system was operating correctly, we used a tool called Coverage to show that the system was 82% covered by unit tests, this is sufficient enough to claim that the system would always work as intended. The user study was comprised of two parts, feature testing and a user questionnaire.

The results from the feature testing concluded that all of the features that were tested were helping build a more usable system. The users were able to carry out an analysis of the classified emails and draw results from the information they were given from the system. The questionnaire helped solidify that the overall design of the system was effective and usable. A user commented that the design was good and easy to navigate, they also stated additional features that could be implemented in the future.

Overall the system helped in automatically classifying an email collection, that when confirmed by a knowledgeable reviewer, could be uploaded for public use. This statement is backed up by the results from the unit tests showing that the system is reliable, and from the user study to show that the reviewing process is made faster and easier by machine learning.

8.1 Future work

There is massive potential for this system to be improved in every aspect. These improvements were identified during the user study and from drawbacks encountered throughout development of the system. The main source of improvement in data collecting and user testing would be to deploy the system online for anyone to use, this could be carried out as a test of how everyday users would interact with the system instead of chosen participants executing specific tasks in a controlled environment. This would provide new inputs into where the system may be lacking features and may also help solidify the useful features.

The main components behind the system, while tested and evaluated to be efficient and reliable, could be developed further to improve the overall systems results. The classifier used is not complicated and thorough when compared to ones used by large corporations. With a fully developed and trained classifier behind this system it would help build trust on the classifiers predictions, so that the user does not need to spend the time checking every email to make sure the classifier was correct.

Future work could implement other types of document reviewing, while this project specifically focused on emails, as stated in the introduction, this manual reviewing process is labour intensive for every kind of document that could be released to the public. By having a public document classifier than anyone with doubts their data contains sensitive data could use this system to gain confidence and satisfaction that the information they are putting out is safe.

With a proper development team behind this system it could be applicable in almost limitless number of companies. Any company could integrate the system to investigate employees about misusing company resources and sharing restricted data, national libraries can scan documents and cluster similar ones together to know where they should store them, it could be used by digital forensics teams to scan storage systems for sensitive information. With some changes to the handling of document types, this system could be taken in a large variety of directions.

A Ethics approval

School of Computing Science
University of Glasgow

Ethics checklist form for assessed exercises (at all levels)

This form is only applicable for assessed exercises that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. *No information about the evaluation or materials was intentionally withheld from the participants.*
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. *Neither I nor my supervisor is in a position of authority or influence over any of the participants.*
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. *All participants were informed that they could withdraw at any time.*
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. *All participants have been informed of my contact details.*
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

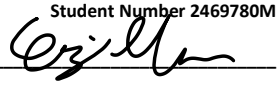
11. *The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.*
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. *All the data collected from the participants is stored in an anonymous form*.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Course and Assessment Name COMPSCI4025P Level 4 Individual Project

Student's Lewis Douglas McDonald Munro

Student Number 2469780M

Student's Signature 

Date 24/03/2023

Ethics checklist for assessed exercises

Figure A.1: Ethics approval

B User Study questionnaire

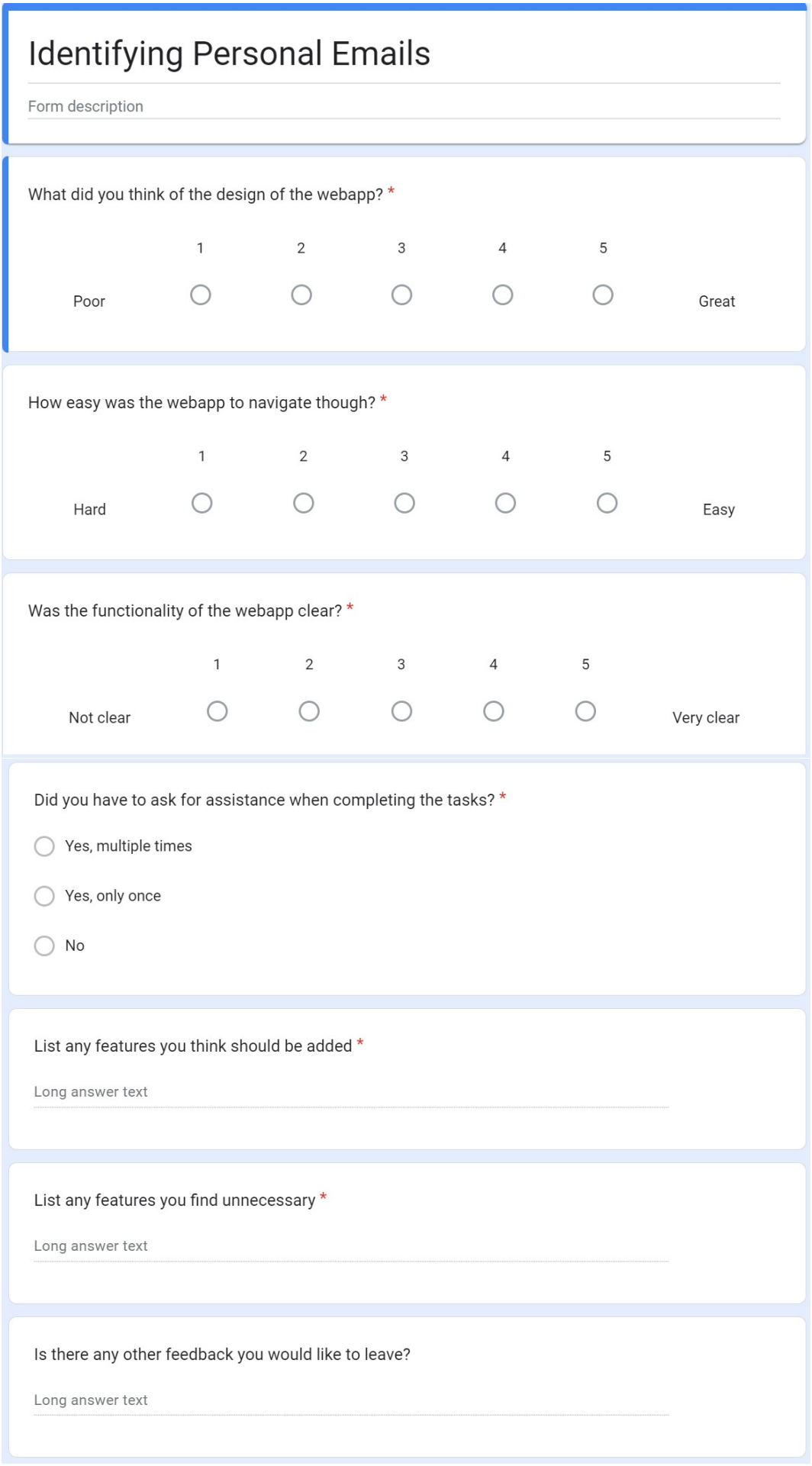


Figure B.1: *User Questionnaire*

C User Study Tasks

User Study Plan

Important features to test

- Highlight Feature
  - Name what you think the key features of this email are that consider it personal?
    - Search for the Personal email 3-122926 and tell me what you think the key features are that make this email Personal?
      - Subject, know, got, guest
  - Name what you think the key features of this email are that fit it to its assigned topic?
    - Search for the Personal email to j.kaminski@enron.com from wolak@zia.stanford.edu with the Topic of public and tell me what you think the key features are that make this email of topic public?
      - Public, need, many

- Number of personal emails from user graph – graph and user page
  - How many Personal emails has this user sent?
    - How many Personal emails has steven.kean@enron.com sent?
      - 67

- Emails received graph on user page
  - How many Personal emails has this user received?
    - How many Personal emails has j.kaminski@enron.com received?
      - 4
  - How many Non-personal emails has this user received?
    - How many Non-personal emails has steven.kean@enron.com received?
      - 66

- Topic graph on the graph page
  - Name what you think the topic of this email should be?
    - Search for the email 1-174050 and tell me what you think the topic should be?
      - federal

Figure C.1: *User study tasks*

8 Bibliography

C. Brewster. 14 companies using machine learning. URL https://www.trio.dev/blog/companies-using-machine-learning.

J. Brownlee. A gentle introduction to the fbeta-measure for machine learning, 02 2020. URL https://machinelearningmastery.com/fbeta-measure-for-machine-learning/#:~:text=F1%2DMeasure%20(beta%3D1.0,precision%2C%20more%20weight%20on%20recall.

C3.ai. Precision. URL https://c3.ai/glossary/machine-learning/precision/#:~:text=What%20is%20Precision%3F,the%20number%20of%20false%20positives).

D. Campbell. Gmail spam filter: Everything you need to know, 05 2022. URL https://www.rightinbox.com/blog/gmail-spam-filter.

Coverage.py. Coverage.py. URL https://coverage.readthedocs.io/en/7.2.2/#.

D. Django Software Foundation. Meet django, a. URL https://www.djangoproject.com/.

D. Django Software Foundation. The django template language, b. URL https://docs.djangoproject.com/en/4.1/ref/templates/language/.

django tables2. django-tables2 - an app for creating html tables. URL https://django-tables2.readthedocs.io/en/latest/index.html#.

FCDO. Digital records sensitivity review. URL https://www.fcdoservices.gov.uk/what-we-offer/digital-records-sensitivity-review/.

C. Fontanella. 75 stop words that are common in seo when you should use them, 11 2020. URL https://blog.hubspot.com/marketing/stop-words-seo#:~:text=The%20most%20common%20SEO%20stop,your%2C%20our%2C%20and%20their.

R. Gandhi. Support vector machine — introduction to machine learning algorithms, 06 2018. URL https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

V. K. Garbhapu and P. Bodapati. A comparative analysis of latent semantic analysis and latent dirichlet allocation topic modeling methods using bible data. *Indian Journal of Science and Technology*, 13, 2020.

GeeksforGeeks. Naive bayes classifiers, 01 2023. URL https://www.geeksforgeeks.org/naive-bayes-classifiers/.

Ideagen. Pleasereview: simultaneous document review software. URL https://www.ideagen.com/products/pleasereview.

I. Information Commissioner's Office. What is the freedom of information act? URL https://www.gov.uk/make-a-freedom-of-information-request.

Intersoft-consulting. Art. 4 gdpr, definitions. URL https://gdpr-info.eu/art-4-gdpr/.

InterviewBit. Top 11 python frameworks you must know in 2022, 06 2022. URL https://www.interviewbit.com/blog/python-frameworks/.

Ioana. Latent semantic analysis: intuition, math, implementation, 05 2020. URL https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8.

M. Makai. Full stack python - flask, 2022. URL https://www.fullstackpython.com/flask.html.

Microsoft. Sensitivity label insights about your data in microsoft purview, 03 2023a. URL https://learn.microsoft.com/en-gb/azure/purview/sensitivity-insights.

Microsoft. Microsoft purview ediscovery solutions, 02 2023b. URL https://learn.microsoft.com/en-us/microsoft-365/compliance/ediscovery?view=o365-worldwide.

ostezer and M. Drake. Sqlite vs mysql vs postgresql: A comparison of relational database management systems, 03 2022. URL https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-syst

PostgreSQL. Postgresql: The world's most advanced open source relational database. URL https://www.postgresql.org/.

P. Quinn and G. Malgieri. The difficulty of defining sensitive data—the concept of sensitive data in the eu data protection framework. *German Law journal*, 22, 01 2022.

G. Rudusans and G. Vitols. Machine learning methods for classification of sensitive data. *Research for rural development*, 36, 2021.

N. Seth. Part 2: Topic modeling and latent dirichlet allocation (lda) using gensim and sklearn, 06 2021. URL https://www.analyticsvidhya.com/blog/2021/06/part-2-topic-modeling-and-latent-dirichlet-allocation-lda-using-gensim-and-sklear#:~:text=Latent%20Dirichlet%20Allocation%20(LDA)%20is,are%20also%20%E2%80%9Chidden%20topics%E2%80%9D.

A. Shafi. Random forest classification with scikit-learn, 02 2023. URL https://www.datacamp.com/tutorial/random-forests-classifier-python.

SQLite. What is sqlite?, 03 2023. URL https://sqlite.org/index.html.

W3Schools. Css tutorial. URL https://www.w3schools.com/css/.

W3schools. Mysql tutorial. URL https://www.w3schools.com/MySQL/default.asp.

Wikipedia. Enron corpus, 2022. URL https://en.wikipedia.org/wiki/Enron_Corpus.