**CO600 Terms and Conditions Analyzer**

Adam Preston (ap720), Euel Domingo (ejd29), Kester Gideon (kg327), Marcus Robinson-Bailey(mer25)

*School of Computing*

*University of Kent*

*Canterbury, United Kingdom*

**Supervised by**: Alex Freitas ([A.A.Freitas@kent.ac.uk](mailto:A.A.Freitas@kent.ac.uk))

**Abstract**

This report describes the stages and processes involved in creating the terms and conditions analyser. The system analyser is a user-friendly website that highlight any risky statements in the document. With the help of machine learning, the analyser will scan through the document inserted and compare with a trained set of words we deemed to be risky. The paper will begin by describing how the concept came about then move on to provide more information on the background and aims of the project. Further description will be given on the product development and testing of the final product. We will also show the languages chosen and the requirements given for risky statements.   In addition to this the report we will provide a technical description of the classification algorithms used, quality assurance and challenges faced.

**1. Introduction**

There have been countless occasions where individuals sign documents or contracts, whether they be phone contracts or social media terms and conditions with no intention of reading the terms and conditions. A recent survey concerning social networking terms says that more than 30% of the respondents said they have never read the terms and conditions when signing up to social networks.(1).In relation to this current issue, we decided to create a terms and conditions analyser. The purpose of this would be to  would allow users to insert their conditions document in plain text into the web interface which would then indicate the risky statements by highlighting them in the text area. The risky words highlighted allows the user to focus on the sentences and hence review the document before taking further action. The idea would instigate that machine learning would be involved and the use of Neural Networks. Machine learning is a method of data analysis to effectively perform tasks and it has recently received an increase in interest over the last few years. Machine learning helps in various aspects; whether it is predicting the next video on YouTube, to predicting when earthquakes could occur. The report highlights the process we took from the generation of the idea, to the completed phase. Research was undertaken in the initial stages into different terms and conditions, machine learning API, pre-processing scripts. Following this we planned our project by setting out the requirements and the design plan.

In preparation for the project, we realized that this would entail additional knowledge to be acquired in order to produce a system that is able to read through sentences and highlight the risky statements. We quickly realized that in order to create a good machine learning system, we would have to follow some required steps. We would first take into consideration data preparation capabilities. Data would be collected and words we deemed risky would be placed into a document. Terms and conditions from a few top companies were used. A pre-processing script was created to accept the words we assumed were risky and show the number of occurrences of each risky word.

**2. Background**

**2.1 Market Research**

After conducting market research on our specific topic, we came across applications that held the same attributes as the ones we wanted to include in our software (EULAlyzer, Terms of Service Didn’t Read, Polisis). Terms of Service Didn’t Read was the one main application that mirrored features that we wanted to implement. The purpose of performing market research for the project was to gather and calculate ideas for our university project. It also allowed us to brainstorm ideas but more importantly inspired us to make an application that first met user requirements and secondly push the boundaries of our data mining knowledge.

**2.1.1 Terms of Service Didn’t Read**

Terms of Service Didn’t Read [1] is our biggest competitor within the market we have chosen. This is so, because it both analyses the terms and conditions of a website but also provides browser add on compatibility. This feature would naturally within your browser rate and label website policies with class ratings ranging from Class A (good) – Class E (very bad). This would then inform the user about the rights they have on the specific website they’re signing up to. From Terms of Service Didn’t Read we acknowledged that there were aspects of the application that we didn’t want to implement for the shear reason that the feature would be very time consuming, expensive to implement and would slow down production of our main features.

**2.1.2 EULAlyzer**

EULAlyzer[2] is another software that held the same concepts as the ones we wanted to implement. Created by Brightfort the software prides itself on identifying important elements of terms and conditions. Even though EULAlyzer wasn’t our main competition like Terms of Service Didn’t Read it again helped with research and understanding what is essential for our project idea.

**2.1.2 Polisis**

A smaller advertised application was Polisis[3], created by an independent developer. An application that visualised privacy policies using artificial intelligence. It highlights information that a website is collecting from you and possibly sharing to external agencies. Researching both levels of application production shows the difference in quality and outlined how we had to be patient with implementation. Researching a spectrum of applications, it reassured that there would be limitations in our project, and we shouldn’t label the lack of features as a limitation.

**3. Aims**

[AIMS OF THE PROJECT I.E. WHAT WE WANTED TO ACHIEVE, WHAT THE PROGRAM NEEDED TO ACHIEVE TO BE A SUCCESS AND IF IT CATERED TO OUR AUDIENCE] - EJ

**4. Requirements**

[TALK ABOUT SYSTEM REQUIREMENTS, DATA MINING AND AUTOMATED MACHINE LEARNING] - EJ

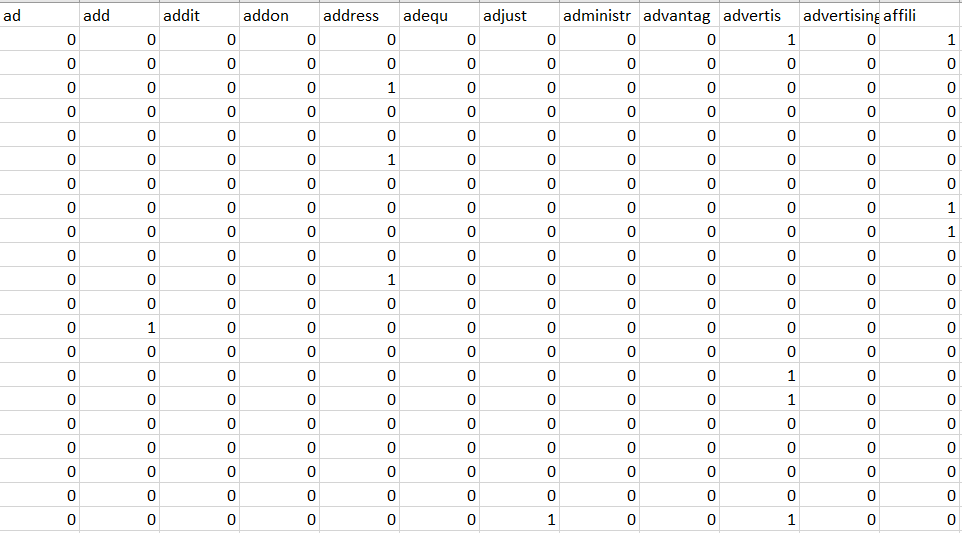
**5. Development**

[INTRODUCE OUR APPROACH TO DISTINGUISH BETWEEN RISKY/SAFE SENTENCES AND EXPLAIN HOW CLASSIFICATION ALGORITHMS WORK - ADAM]

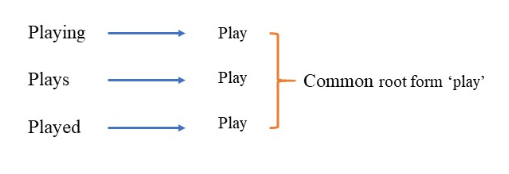
Preprocessing Script

The model responsible for classifying sentences as either risky/safe needed first to be trained by a classification algorithm using a large dataset which indicated the distinction between ‘risky’ and ‘safe’ sentences. To provide the required dataset, a preprocessing script had to be developed that transformed our interpretations of risky/safe sentences into a table that could be interpreted by a classification algorithm. In addition, the preprocessing script is also responsible for manipulating the dataset in order to enhance the classification algorithms ability to generate a suitable model. To give an example, we may program the preprocessing script to deliberately exclude certain words being included as attributes. As humans we judge that these words have no relevance to the model we wish to generate and therefore excluding them can improve both speed and accuracy results.

A sample result of the preprocessing script is given below



The columns represent words that the model would check the presence of while the rows represent one sentence extracted from the privacy policies. A cell can contain either a 1 or a 0 and indicates the presence of a word in a sentence.

The data used by the classification algorithm must be provided by humans, as the classifier in the context of this example will attempt to emulate their judgement. Before the end of each sentence a human examiner deemed as risky, the examiner would include a codeword (0cf333). After the pre processing script had used the nltk toolkit to extract the sentences from the provided documents, it would look for the codeword within each sentence. If the codeword is found, then the preprocessing script can store the corresponding sentence in a collection of risky sentences. If the codeword is not present, then the sentence is stored in a collection of safe sentences. The pre processing script then performs a filtering process on all words recorded when examining the documents to generate the attributes which will be included as the column headers. The script removes duplicates, punctuation, stopwords, excluded words, words that occurred too few times in the documents and numbers before applying stemming which groups words of similar meaning into one attribute. 

All risky and safe sentences are then re examined and the presence of attributes within each sentence recorded before being exported in a csv format to be viewable as a table.

Developing the classification algorithm

The sklearn library gives you access to hundreds of classification algorithms that can be implemented within three lines of code. However, despite this implementation friendly package, choosing the correct algorithm and providing the optimum parameters is challenging. The classification script uses AutoML to iteratively build and test models while gradually modifying the parameter values on each iteration to find the best classification algorithm for our project. Currently, the classification algorithm that generates the most accurate model is a neural network that has 2 hidden layers and 33 hidden units in each layer.

As risky sentences occur once in every 15 sentences, they are hugely underrepresented in comparison to safe sentences. This poses a problem for trained classifiers. While classifiers in this scenario are very accurate when predicting sentences as risky (high precision score), they fail to identify the majority of risky sentences (low recall score). Using sklearn’s library, we can oversample the risky sentences in the training data and rebalance the trade off between recall and precision scores. Although oversampling does reduce the precision, it forces the classification algorithm to consider the risky sentences more greatly and therefore risky sentences are identified more often. There are also many other factors that contributed to enhancing the classifiers performance. For example, by displaying the sentences which were identified as risky, we could more effectively deduce the cause of inaccuracies. We often noticed that questions were being classified as risky. By removing ‘?’ from the list of excluded words, the ‘?’ was re-introduced into the identifiable attributes and the classification algorithm could learn that when a ‘?’ is present, the question should be identified as safe.

Classification API

The motivation for developing an API is so that our website can communicate with our classifier and is essential for enabling normal users to analyse privacy policies.

When the API is started, it runs the classification algorithm to generate a predictive model. Once a request is received, the API parses the given document and extracts the sentences. Each sentence is then run through the classification algorithm to obtain a prediction (risky/safe). All risky sentences are returned to the website in JSON format so they can be displayed to the user.

**6. Quality Assurance**

[TALK ABOUT TESTING] – ADAM

**7. Challenges**

**7.1 Accuracy**

One major challenge that we experienced was producing accurate data. This involved accuracy in our pre-processing data and making sure that all the data collected followed the same guidelines. Originally, we had an issue with contrasting data where we all had conflicting views on what was considered “risky”. So, when going through our data collection process we gathered a lot of data that wasn’t consistent. This was a problem for us but in order to resolve this problem we created Terms and Conditions guidelines to keep us focus on what should be considered “risky” and prevent us from swaying in opinions.

Website

**7.2 Expectation exceeding reality**

No matter how much we planned, gathered more data or time spent on our project, we always came with the overwhelming feeling of have with set the bar too high for what we expect to complete in the year project. As the idea was based around machine learning, a quite new aspect of computer science, we generally came across the hurdles of either the knowledge scope of development for the analyser was way over what we knew as students or the time we had to implement such algorithms exceeded the time we had for the project. Even smaller features of the project like collecting data became a longer process than we first imagined. Looking back ideas we brainstormed at the start of the project could’ve been refined to help us progress further into the analyser and expand its means of functionality.

**7.3 Algorithms**

Another challenge that we faced was choosing the right classification algorithm to apply to our model. With such a wide range of classification algorithms within machine learning e.g. Linear regression, decision trees, neural network, naïve bayes etc we had a real problem with choosing an algorithm that met our model requirements. To have a model that easily could distinguish between “risky” and “non-risky” sentences we needed an algorithm that produced high accuracy and recall results. Accuracy being total number of risky statements correctly identified/total number of risky statements. Recall being the number of examples that have been classified correctly. We as a group had a real issue with low results from a lot of the chosen algorithms and were struggling to find an algorithm that produced a high enough recall result to meet requirements. The only solution to this issue was to collect more training data but due to time constraints this action wasn’t feasible.

**8. Conclusion**

**9. References**

[1] <https://tosdr.org/>

(Viewed 6/2/2019)

[2] <https://www.brightfort.com/eulalyzer.html>

(Viewed 6/2/2019)

[3] <https://pribot.org/polisis>

(Viewed 6/2/2019)

**10. Acknowledgements**

Throughout the CO600 project we were guided and supervised by Alex Freitas. We’d just like to say thank you for all the help and guidance you gave.