

VoiceBase: Detecting Social Security Numbers

Jack Wells, Alex Berriman, Yuqing Shang,
William Davidson

COM6911 Industrial Team Project

Department of Computer Science
The University of Sheffield

December 12, 2017

Contents

1 Introduction iii

2 Literature Review iii

3 Data v

4 Rule Based System vi

4.1 Results vii

4.2 Discussion ix

5 Support Vector Machine xi

5.1 Cleaning the data xii

5.2 Results xii

5.3 Discussion xiii

6 Random Forest Classifier xv

6.1 Results xvi

6.2 Discussion xvii

7 Comparison of Methods xvii

8 Conclusion xvii

9 References xviii

1 Introduction

Social security numbers (SSN's) are an essential part of life for citizens and residents of the United States of America. They are similar to the UK's national insurance number scheme in terms of use and possessing a structured format. The nine numbers that make up the SSN are defined by the person's geographic area (A), a group number (G) and a serial number (S) in the format of AAA-GG-SSSS [1]. Since 1943 the use of SSNs has grown, now being used for a variety of reasons [1]. SSN's are commonly used as personal identification numbers when by customer service centres in the USA. Within these phone calls, the operator may ask for a partial or full social security number so that they have assurance that they are talking to the right person.

This project is about the identification of SSNs in the transcripts of the call centre conversations, as transcribed by the company, VoiceBase. The redaction of these SSNs from the transcripts is an important objective due to SSNs being private and confidential information and should only be known by the person it has been assigned to.

VoiceBase are a California based speech recognition and analytics company. They provide automatic speech recognition (ASR) transcriptions of customer service conversations. The aim of this project was to design an automated system that would pick out the SSNs from the transcripts provided by VoiceBase. The objectives were to:

- Write a rule based system.
- Design a method that involved a support vector machine.
- Test the method that implemented the support vector machine with different features.
- Design a method that implemented a random forest classifier.

The methods discussed in the objectives will be elaborated on in the literature review. The programming language used for his project was Python due to its useful machine learning modules and the experience of the group members.

This report will outline the actions that were taken to achieve the aims of this project. After the literature review there will be a discussion on what was provided in terms of raw data and truth data, which showed where the actual SSN was. Following this, the methods used will be presented, outlining how each was implemented and what was tested to see how effective the method was. The results and discussion will follow, presenting the outcomes of what was tested. The report will end with the conclusion, discussing which method was best and whether the project was a success or not.

2 Literature Review

This project involved using the transcripts produced using ASR provided by VoiceBase to attempt to find any SSNs within them. Three techniques were used in total to find these SSNs, these involved using a rule based system, a support vector machine (SVM) and a random forest classifier. All three techniques involve an algorithm that will have to decide whether a word in the transcript is part of the SSN, this is an example of a binary classification problem [2]. The latter 2 methods proposed above are examples of machine learning techniques whereas a rule based system is slightly different, this will be discussed

shortly. Classification is an essential part of machine learning and aims to automatically assign a data point to a group based on its features [3]. In an age of rapidly increasing online material, it has become more and more essential to utilise computers to sort and extract relevant information from this material. One of the main sources of information is text, and being able to classify sections of this text into opinions, facts, dates etc. is a valuable tool for many purposes. Machine learning methods have long been used for these tasks, with SVMs showing the some of the best performance results out of all of them [4].

A rule based system is one where a set of if-then statements are used to complete an action using the data if a certain condition is met [5]. Rule based systems have been a useful way of taking human intuition to create simple artificial intelligence (AI) and automated problem-solving systems [6]. A rule based system will incorporate the human gained knowledge of a problem and implement if-then statements to perform the best action, which can then be improved with further knowledge about the problem [6]. This type of system has many applications as it can be adapted to whichever situation is required provided a decision based on specific information is needed. For example, the same system produced for this report could be adapted to find the phone numbers in the transcript as opposed to a SSN.

Support vector machines (SVM) are a machine learning algorithm that look to classify a set of data into two sets by constructing a decision boundary that best separates the data. To use an SVM, the data must be projected into 'feature space' [7]. A feature is a value derived from the input data to represent a particular quality of a portion of it. The process of projecting the inputs into feature space is called feature extraction [8]. There may be features that are useful to the learning of the classification machine, but there may also be features that are not useful and may even cause the model to be overfit. Ideally, only the features that are useful to the success of the SVM should be used, this is feature selection [2].

SVMs have many benefits that make them especially well-suited to text classification problems. One is their ability to use many features to train with [9]. This is likely to be needed with text classification problems as this allows each word, deemed to be of interest, to be used to improve the prediction. These features are then plotted in multiple dimensions and a hyperplane decision boundary is drawn between them, that maximises the margin or the distance to the closest data point to ensure they are separated as best as possible [4]. The nature of how SVMs linearly classify is especially suited to text classification tasks as they can usually be linearly separated, for example, if it contains a credit card number or not. They are also effective at filtering out noise and outliers.

The library that was used for the machine learning was scikit-learn, which contains many classification and regression algorithms as well as other functions needed to train and test our predictions. Both the Support Vector Machine and Random Forest classification tools were used as a means of predicting the positions of a social security number. Input parameters to these functions can be altered to affect how the decision boundary is drawn, changing the classification of the data points. The *train_test_split* function was used to split the data into training and testing sets into a 2:1 ratio. The function *accuracy_score* was also used to compute the overall prediction accuracy, along with the recall and precision.

A random forest is another type of machine learning technique often referred to as an Ensemble method [10]. According to Benyamin [10], The main principle behind ensemble methods is that a group of weak learners can come together to form a strong learner. By combining the predictions from each separate weak learner, the ensemble can improve the performance over a single learner [10][11]. A random forest fits a number of decision tree classifiers on various sub-samples of the dataset to improve the predictive accuracy and control over-fitting by using an average. Each tree in the ensemble is built from a sample drawn with replacement from the training data set. Moreover, during construction of the tree, the split of a node is picked from a random subset of the features, instead of the best split amongst all features [11].

3 Data

The data provided by VoiceBase consisted of 3,278 JavaScript Object Notation (JSON) files. Each JSON file included, as well as file metadata, a transcript of a single call. These files could be loaded into memory using the Python module *json* and accessed as a Python dictionary. The most important part of the transcript is the 'words' list. This is a list of the words transcribed by the ASR software in chronological order, spoken by either the customer or the operator the software does not distinguish between the two. For each word transcribed, the JSON file provides:

- Position index (p-value)
- Start time in milliseconds
- End time in milliseconds.
- Confidence score
- Volume
- Frequency and energy of the dominant frequency pair

VoiceBase also provided a 'ground-truth' spreadsheet which indicated the mid-point times of words that were believed to be SSNs. This data proved very inaccurate, possibly due to operator error when tagging the start and end of SSNs in their conversation. In addition, only 328 conversations were included in the ground-truth, compared to the 1000+ that were expected to contain SSNs. A script was written to convert these mid-point times into p-values for use in the algorithms. When manually checking the words where the times matched the ground-truth it was revealed that many of them were not number words. This confirmed the lack of confidence in the accuracy of the ground-truth.

A new, revised ground-truth was then provided which instead of mid-point times, provided ranges of p-values. Approximately one third of the transcriptions provided in JSON were believed to include an SSN, and this ground-truth was far more complete. When manually inspecting the words tagged as SSNs, there was again a high proportion of non-number words. This could either be due to inaccuracies in the ground-truth, with the wrong words being labelled as SSNs, or it could be a speech-to-text error, where a number spoken in the conversation was instead transcribed as a non-number word.

A script was written to export the list of words from the JSON files into an equal number of text files. This was done to simplify the process of accessing the words using Python. Rather than loading the JSON file into memory and accessing it as a dictionary, the words could now be read one-by-one from the text files with no extra steps, which increased the speed of the algorithms.

Due to the large number of files, it is hard to know how inaccurate the data truly is. It was found that only half of the tagged words were number words, often caused by the words either side of a SSN being tagged. These words add a lot of noise when training the machine learning algorithms as they were effectively trying to predict the occurrence of words such as please and number. While these may be good indicators that a SSN is about to be said, they are not the target to be classified as a SSN. It made it hard to judge what was a good accuracy score as even if the perfect classifier was built, the accuracy of the predictions would have been skewed by the inaccuracies in the data. Many of the p-value ranges that were given as a SSN were more than 30 words, which is clearly incorrect as an SSN can only be nine digits long. These files were removed when cleaning the data.

4 Rule Based System

The procedure for making the rule based system involved looking over many transcripts to get an idea of how a conversation including a social security number developed. Around 50 transcripts were manually inspected in the first instance to get an idea how it looked. However, approximately 100 transcripts were inspected in total to try to get a better picture of how a rule based system could be implemented. The transcripts were inspected alongside the truth data to see where the social security number was tagged by VoiceBase. It was found that in most cases the call centre employee would ask the customer for their social security number, subsequently the customer would read out the section of the social security number that was requested. This element of the rule based system will be discussed later in this section of the report. In most instances that was the end of the useful part of the transcript. However, on some occasions the call centre employee would repeat the number or numbers back to the customer or it would need repeating by the customer for further clarification. Anywhere in the transcript that the social security number appears needs to be redacted as this is all private information.

It was also decided that the words that the system should pick out from the transcript should only be number words (one, two, three etc.). To elaborate on this, a list of all possible configurations of number words (for example: to, too and two are all possible versions of the number 2) was created and used in conjunction with one of the if statements in the final iteration of rule based system. This list will be referred to as the list of number words. Another list of words was made that included words that usually signified the end of where the social security number was said. In the final version of the rule based system this list only included the words of thanks and thank, as these usually indicated the end of where the social security number was said. The effect of different words in the list end words will be discussed more in the discussion section.

The method for creating and improving the rule based system was to attempt to come up with rules, in the form of if statements, that would hopefully get closer to finding where the social security number is according to the truth data.

The rule based code worked by first turning the .txt file of the transcript into a list of the words in the order they appear. The algorithm will then look through the list for the words social and/or security. When one or both of these words are discovered the system will look over the next N number of words to look for words that are also in the list of number words. If the word it comes across is in that list, then the system will return the p-value of that word. This returned position value will make up the estimated position values of the SSN. When iterating over the next N words, if the system comes across a word that is in the end words list, the search will stop, and the system will then continue through transcript looking for words of social and/or security and repeat the procedure until one is found or the entire transcript has been looked through. If both social and security are not in the transcript then the system will not perform any action on that file. The code produced can either return a list of the positions it picks out as estimates for the social security number or it can return a range of estimated position values starting from the minimum position value up to the maximum position value found. These two methods will be referred to as the individual method and the range method respectively.

The justification for only looking for the words social and security when looking over the transcript was to do with the fact that these words were the only consistent indicator of when a social security number is going to be said. Although, due to some transcriptions not being correct, there are cases where the words social and security arent transcribed but a social security number has been tagged within the transcript. It was very difficult to write a rule that would pick up these tagged social security numbers as there was no sensible way to do it without there having to be a lot of human interaction to check to see if it was correct. It was found that, when looking only at the transcripts that had social security numbers

in (according to the truth data), 85.6% of the transcripts had both the words social and security or just security in them. Whereas the remainder (14.4%) of the transcripts that were tagged, had neither word in them and there were zero occurrences of just social being said. The rule that dictated what should happen if only the word social was found was kept in for the final system as there may be a transcript added to the data set where this does in fact occur. With respect to the transcripts that were flagged as having no social security number in them, the percentage of those transcripts that included and did not include the words social and security can be looked at again. For this part of the data set, it was found that 13.9% had both social and security in, 14.4% had just security in and 71.7% had neither. Again, there were no instances of only social being said but the rule has been left in for the same reason as before. It was concluded that using social and security as the indicator of a social security number was justified, as according to the truth data when there is a social security number there, most of the transcripts have the words social and security or just security and vice versa when the truth data states there isnt a social security number there.

After the rule based system was written, tests were performed on different elements of the code to see what effect it had on the system. These tests were:

1. Whether it is better to use the individual method or the range method.
2. Effect of changing how many words the algorithm will iterate over, when the system finds the words social and/or security, to look for the number words.
3. Effect of the length (number of words) in the social security number as tagged by the truth data.

These results will be validated by looking at their scores for recall and precision. Recall is the fraction of those position values that were picked up by the system that are also relevant, or more simply put the number of correctly found position values (according to the truth data) to the total number of actual position values. Precision is again a fraction of the number of correctly found position values to the total number of position values the system picks up. The use of recall and precision for the files that do have social security numbers in them is fine since all elements of the formulas of recall and precision can be computed with the information from the truth data for those transcripts. However, there is a problem with this method for the transcripts that do not contain social security numbers as the elements of the formulas of recall and precision cannot be computed with the information from those transcripts. For the transcripts that, according to the truth data, do not possess a social security number, success was measured by how many the system got right for those transcripts (it would be correct if it picked out no numbers using the system). We will discuss the effectiveness of both these ways of measuring success in the discussion section of this report.

4.1 Results

For the first set of results we will discuss is how well the final of the version of the rule based system works when tested against the truth data. In the final version of the rule based system the end words used were just thanks and thank. The number of words that the system iterates over when one of the words social or security is found is set at 50. This first test will be asking whether it is better to return the individual position values of the suspected social security number or a range of values from the minimum position value up to the maximum position value found. The results for this are presented in Table 1.

From the results in Table 1 we can see that for the individual method, the recall and precision are quite similar whereas the results from the range method has a much better recall but a very low precision.

Table 1: Recall and precision for rule based method

| | Social security number data | | No Social security number data | |
|-------------------|-----------------------------|-----------|--------------------------------|---|
| | Recall | Precision | % correct | Total number of suspected position values |
| Individual method | 0.5646 | 0.5085 | 74.51 | 4804 |
| Range method | 0.7426 | 0.05185 | 74.51 | 58661 |

Both tests have the same number of correct transcripts for the No Social security number data but the Range method returns a lot more suspected position values than the individual method.

The next experiment looked at changing the number of times by which the system iterates when one of the words social or security is found, referred to as the N number of words. In the final rule based system this was set at 50 words. This experiment used N values ranging from 5 up to 80, ascending in 5s. No end words were used in this experiment so that the full effect of changing N could be observed. The results from looking at the effect of changing N on the individual method are shown in figure 1.



Figure 1

As the number of words, the system iterates over (N) is increased the recall asymptotically increases to a limit. As N is increased the precision decreases asymptotically. These observations are expected as the system is more likely to find the right values but may also pick up a lot more incorrect numbers whilst doing so. The results of changing N for the range method values are presented in Figure 2.

For the range method the recall also increased asymptotically with N. However, the precision is very low for all values of N and decreases slightly with each increase in N. Again, this is expected as the system is more likely to find the correct values with a bigger range but will also pick up a lot more incorrect numbers whilst doing so.

The final element of the rule based system to be investigated was considering how the length of the tagged social security number effected the recall and precision calculated. The end words used were just thanks and thank and value of N was set to 50. The results from this experiment are presented in Table 2.

The main point to make from Table 2 is that the worst result for both methods is in the social security

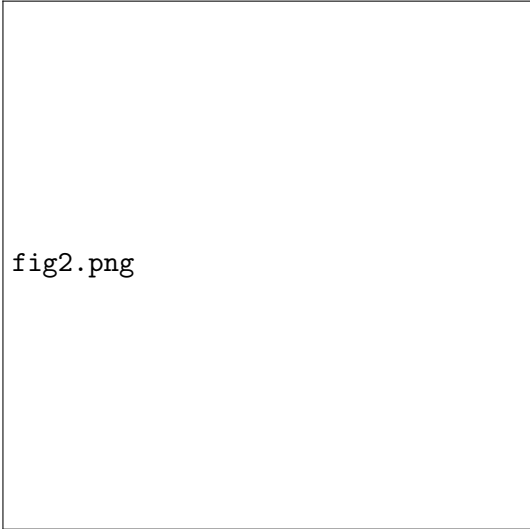


fig2.png

Figure 2

Table 2: Effect of p-value range on rule based accuracy

| Length of social security number | Effect on Individual method | | Effect on Range method | |
|----------------------------------|-----------------------------|-----------|------------------------|-----------|
| | Recall | Precision | Recall | Precision |
| 1-3 | 0.4946 | 0.2401 | 0.6064 | 0.02033 |
| 4-5 | 0.6374 | 0.4190 | 0.7033 | 0.04026 |
| 6-10 | 0.6439 | 0.4641 | 0.7369 | 0.03601 |
| 11-15 | 0.5585 | 0.5702 | 0.7454 | 0.07038 |
| 16-20 | 0.5350 | 0.7645 | 0.7390 | 0.08185 |
| 21-25 | 0.5600 | 0.7656 | 0.8470 | 0.1106 |
| 26-30 | 0.4670 | 0.7317 | 0.7811 | 0.08100 |

of length 1-3. It is believed that the poor result is due to a combination of bad transcription and human error in the word tagging. The best result for both tests cannot be categorically stated as it depends on what is deemed to be more important, recall or precision.

4.2 Discussion

A rule based system has been written that attempts to the social security numbers in the set of transcripts, the results were then tested against the truth data that was provided. Two different ways of returning the suspected positions of the social security numbers were presented. The first was to return only the position values of the words the system picks out that coincide with those in the list of number words. The second was take what was returned from the first but return a range of position values from the minimum to the maximum position value found. These two methods result in quite different values for recall and precision. For the individual method the recall achieved on the final system is 0.5646 whereas the range method achieves 0.7426. These are the results on the data that is known to have social security numbers in. The range method is expected to have a better recall as it can pick out the non-number words that are tagged as social security numbers whereas the individual method cannot do this. The trade-off of recall to precision must be considered when deciding which method to use. For the individual positions the precision achieved on the final system is 0.5085, whereas the range of positions method gains 0.05185. Again, this is expected as the range method will be selecting a lot of positions that arent

the tagged ones from the truth data. It would have to be decided what is more important, finding as many of the social security numbers as possible or to not redact too much of the non-social security part of the transcript. Both methods have the same number of correct transcripts for the data that did not have social security numbers in, but the range method returns a lot more suspected position values than the individual test. This is an expected result since the system would have not picked out social security numbers here due to there not being the words social or security in the transcript. The recall for both methods and the accuracy for the non-social security data is still lower than desired and it means that at the current moment the system cannot be fully trusted.

The results from the second experiment show that for both methods increasing N increases the recall asymptotically but will decrease the precision attained. This observation is expected for the individual method as the system is more likely to find the right values but may also pick up a lot more incorrect numbers whilst doing so. For the range method, this is also expected as when N is increased it is more likely to get more of the correct number words and the non-number words tagged by the truth data, but the precision is very low. Another point to make about what effects recall and precision is the use of the end words list. In the final rule based system the only words in that list were thank and thanks. Some investigation into other words to use, such as telephone and credit, was conducted but no useful results were gained. This is because it was very difficult to write a rule that will be applicable to all conversations. For example, using other words in the end words list may only be useful on a handful of transcripts but may inhibit the effectiveness on others. The use of the end words list had the most effect on the precision as opposed to the recall due to its ability to cut the search for number words off early. Again, this is a question of what is more important; recall or precision.

The third experiment looked at the effect of the length of the social security number as tagged by the truth data. It was found that a social security number that was tagged to be 1 to 3 words long incurred the worst results on the average for both the recall and precision. This could be due to a combination of errors from both the transcript and human error in the tagging of the truth data. This is evident from looking through what was tagged as there was quite often no number words tagged or even any number words in that region. Other results from table 2 show that there is quite a lot of inconsistency in the transcripts and the truth data. The different methods also get different relative results for different lengths of social security number. The inconsistencies are believed to be due to the proportion of number words that are tagged as part of the social security number in that region of length.

This brings us on to how the rule based system could be improved as in its current state the results gained are too unreliable. One major improvement, that would make the rule based system work a lot better, is better accuracy on the ASR transcripts. ASR will improve with more research and advances in technology in that area and would make all the methods used in this report considerably better. Another improvement for the rule based system would be to have more accurate truth data for the testing of the system. Improvements in the truth data would help to see where the system could be made better. Another approach to the rule based system would be to design a system where it returned a window where the social security could be. This alternate system would allow the user to set the range of how big they wanted the window to be, allowing the user to decide what is more important recall or precision. This system was not written due to the belief that it would not be possible to achieve a good accuracy score with it. That belief stems from the fact that with the current system the recall is not good enough, meaning that it is still missing a lot of the tagged social security numbers. This alternate system would require manual checking on almost all the transcripts which in some respects defeats the aim of the rule based system. The rule based system would have been considered a success if it could achieve a much higher recall and precision, and as discussed earlier this could have been achieved with better transcripts.

5 Support Vector Machine

The SVM was applied to this problem by using features of the voice transcripts that could be used to find correlations between them and the presence of a SSN. After training on a set of data, a classifier would be built that could be used to identify SSNs in unseen transcripts.

The first step was to identify the features that would be used as inputs into the SVM. A word list of interest, 50 words long, was manually created that contained words that would be good indicators as to whether a SSN was present. These included words such as social, security and number words like zero and twelve. It also included words such as credit and address that would likely be negative indicators of the presence of an SSN.

Another feature that was tested on was the position of the window in the text. It was found that there was a relationship between the presence of a SSN and the position in the conversation. As can be seen in Figure 3 and Figure 4, the SSNs were more likely to be said nearer the start of a conversation, which makes sense as this is likely to be information necessary for an operator to acquire before performing another task using the number. The window number was used as the input feature which provided an absolute indicator of the position, and the proportional position in the conversation was also used.

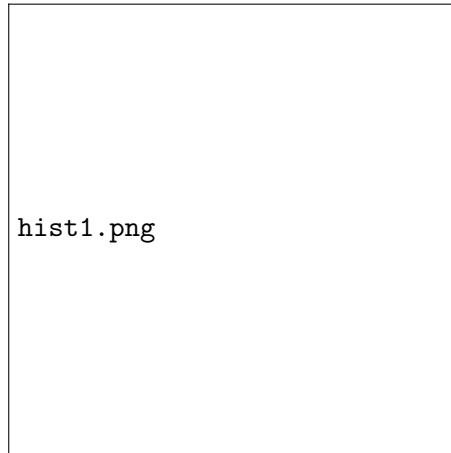


Figure 3

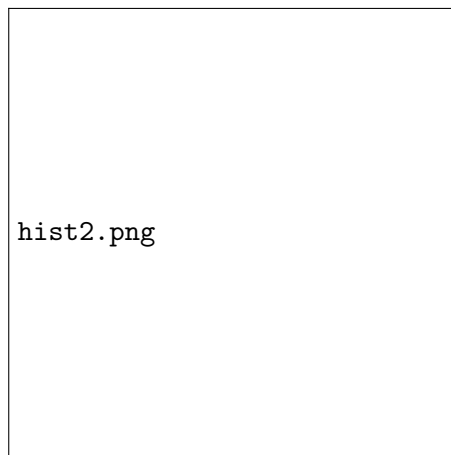


Figure 4

Another feature that was used to build the classifier was the volume of the speech. The sliding window

was used which took an average volume level across the range of words and returned this value to the window scores.

A sliding window of words was created that iterated over the text and returned a score for the frequency of each word that was in the word list of interest as well as an indicator as to whether a SSN was present. The presence of an SSN was determined by whether the window fell in the range of p-values indicated contained SSNs in the truth data. These were the two inputs needed to train the classifier, a list of features of a window and a simple yes or no whether the window contained a SSN.

5.1 Cleaning the data

Although multiple sets of data were provided and cleaned, the data that will be discussed in this section is the most recent truth data from VoiceBase entitled *ssn_training_data_11.21.2017*. This was a csv file showing the start and end position of the social security number in each file, if it was present. Upon first inspection we saw that many files had a very large range, sometimes more than 30 words, of p-values indicating the presence of an SSN. When we looked at the text in this range it was found that although there might be a social security number present, there were many irrelevant words tagged as well. This made these files unsuitable for use in a SVM as the classifier would be training on words which were not good indicators of the presence of a SSN.

Another set of data was provided by VoiceBase, entitled *cleaned_callCriteria_vb_results_11.9.2017* which gave an indication as how well the social security number had been tagged. Most of these fell into the category of having problems with the digits. About 90 of them had been tagged well with only the relevant digits being tagged. Unfortunately, the accuracy of the SVM was very low when trained on this file set. Although the inputs were correct with most of them having positive indicators such as the occurrence of the words social and security where there was one present, the SVM failed to correctly classify these windows as having SSNs in them. The most likely reason for this is that the training set size was too small. Machine learning algorithms almost always perform better with more data, and 90 is a very low number of inputs for a support vector machine.

5.2 Results

Table 3 shows the accuracy results of the SVM, when looking at the first 500 files accuracy scores, 3 iterations, with a window size of 15, moving by 5 each time.

Table 3: SVM accuracy scores for different features

| | Accuracy of SSN prediction (%) | | | |
|---|--------------------------------|-------|-------|---------|
| | Run 1 | Run 2 | Run 3 | Average |
| With only text as features | 45.2 | 46.0 | 46.1 | 45.7 |
| With absolute word position as feature | 19.2 | 20.6 | 21.8 | 20.5 |
| With propotional word position as feature | 46.3 | 47.4 | 44.8 | 46.2 |
| With volume as feature | 41.3 | 39.0 | 42.1 | 40.8 |
| False Alarms | 0.5 | 0.4 | 0.4 | 0.4 |

5.3 Discussion

Figure 5 shows how the accuracy of the predictions change with the p-value range of an SSN, and Figure 6 shows the accuracy against the file number where the files are ordered from smallest p-value range to largest. The blue line shows the accuracy of the current section, and it can be seen that it fails to predict any SSNs when the p-value range is greater than 4, or the file number greater than 500. This was an influencer in the decision to only look at the first 500 files, rather than the whole data set. It was also concluded that there was little use in including the files where there was no SSN present as there were already enough negative examples, windows with no SSN, in the SSN files to train on.

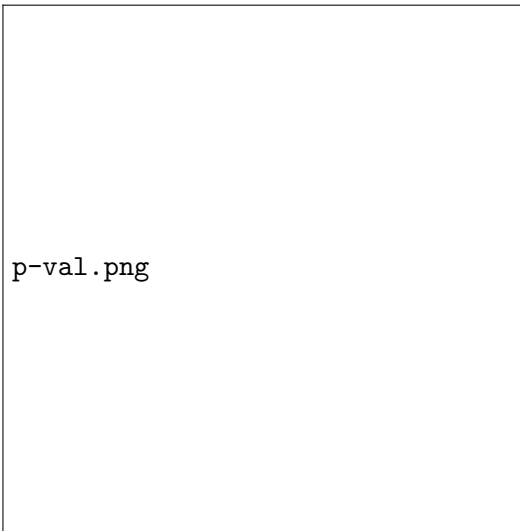


Figure 5

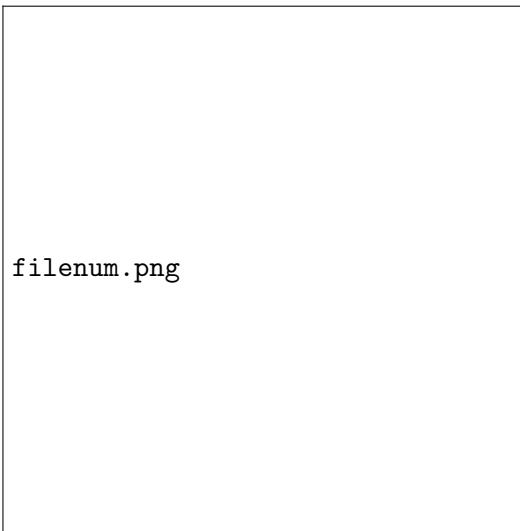


Figure 6

The accuracy of the SVM was found to rapidly decrease when the range of p-values increased, so the decision was made to filter out all of the largest ranged p-value files. There were also files with very small p-value ranges which may only have one word tagged as a SSN. Although these instances may have missed out a few other number words that it should have tagged, they proved to be the best files to train the classifier on as they had very little noise in the form of incorrect words being tagged. The SVM

accuracy scores on these files were far better than on other files.

The accuracy of the SVM is very high, at over 98%, when looking at every window. This value however, is highly skewed by the overwhelming majority of correct classifications of where a social security number is not present. The main problem with the predictions of the SVM was missing the social security numbers, predicting them as a 0 when they should have been a 1. There were very few false alarms, with 99.6% of windows that didnt contain a social security number being correctly tagged as 0. Due to the nature of the task, redacting sensitive personal information, it seems that recall should be valued higher than precision, so little work was done to reduce the small number of false alarms.

The results when including the extra features were not entirely as expected. They showed that using the absolute position of the social security number decreased the accuracy of the predictions by about half, while using the proportional positions in the conversation only slightly improved the accuracy. This was surprising as it would make sense that for example, the social security number was asked for after one minute of talking rather than halfway through the conversation, but this seems not to be the case. The average volume of speech over the window also did not improve the accuracy of the predictions.

It was expected that using more features of the recordings would help the classifier as it would have more correlations to draw from and would therefore make better predictions. However, all of the non-word features that were included decreased the overall performance of the support vector machine. It is unclear as to why this happened as it was expected that including new features would either improve or not change the prediction accuracy.

The window size refers to the number of words that the sliding window is looking at each time it moves. Each of these windows is scored based on how many words in it are also in the word list of interest, and sometimes other features such as window position and average volume. Most of the computation was performed with a window size of 15 words being used, which moved by five words each time and computed a score for that window. Figure 7 shows the results when five sets of window sizes were used, from 10 to 50, still moving by five words each time.



Figure 7

There is a noticeable reduction in speed as the window size increases, but this is accompanied by an improvement in the accuracy at predicting SSNs. This increase in accuracy must be looked at with caution however. It is harder to predict if a SSN is in a very small range such as 10 words, which would be a very accurate prediction, than it is to say it is somewhere within a 50-word window. One way to

approach this problem could be to pull out the number words from these large windows and return the position of them. Another would be to look at the window overlap and take the 5-word section that has the most positively tagged windows. There is a sharp drop off in accuracy when the window size gets smaller than 15 words, with a 10-word window being unable to predict any SSNs. One reason for this is that there is less information available for training, and also that even if the words social and number are detected, the actual social security number may not be said for another 12 words, for example, and it would then be missed by the classifier.

The code seems to be fairly slow and computationally heavy, and there are a number of parts that are surely extending the time it takes to run. However, each individual file is still computed fairly quickly and could be done automatically after each conversation has been transcribed in a few seconds. Maybe if there was a large backlog of conversations that needed to be redacted very quickly, the speed would have to be looked at and improved.

6 Random Forest Classifier

Considering the task was to recognize the social security numbers from the given data, the most relevant information should be words and word position. As for the words, after generating several conversation transcripts to observe, it was found that not all the words are relevant to social security numbers, for instance, greeting and farewell words were sometimes included. If the data set used contained these words, it would affect the models accuracy, and increase the computation time of our model. On this foundation, the relevant words were classified into four-word bags and named as: keywords, digits, noises and others. The bag of keywords contained the two words social and security, which are the most likely indicate the following numbers are going to be social security numbers. The bag of digits not only includes number words but also contains the words which are likely recorded as wrong words, for example two and to. In the bag of noises, the words of 'telephone', 'phone', 'credit', 'card', 'road', 'street', 'house', 'identification', 'address', 'addresses', 'flat', 'cards', 'contact' and 'birth' were included since these words may signify a set of digits which are not social security numbers. Finally, the bag of others, included the words 'digit', 'digits', 'thanks', 'thank', 'you', 'number', 'numbers', 'please', 'give', 'me', 'your', 'last', 'first', 'would', 'you', 'full', 'provide', 'me' and 'confirm'. These words didnt indicate social security numbers directly, but in the conversation, they could be used as supporting evidence of a social security number existing.

With the classified bags of words, the next step was to consider the form of input data set. Firstly, due to lots of redundant words, its inappropriate to set each conversation as a separate element of the data set. Secondly, since the basic social security numbers format is 9 numbers presented as AAA-GG-SSSS [1] and there are relevant words around social security numbers, it is reasonable to use a window which is composed of 15 words as an element of data set. To avoid missing any information (the window generated automatically may just contains part of the required information), the step for the sliding windows was set at 5 words. The improved truth data provided by VoiceBase was used to create the data set as presented in Table 4. In Table 4, the label *Session_id* is the name of the given JSON file, *WinIndex* and *WinPos* are the index and position of each separate sliding window in every JSON file. And under the label of *key_words*, *digits*, *noises* and *others*, are the count of words from each bag that appears in that window. As for the label *SSN*, a 1 shows that this window does contain social security number according to the truth data, compared to 0 that signifies it does not.

According to the task and the data set that has been created for the random forest method, the *RandomForestClassifier* from the *sklearn.ensemble* module were used to construct the classifier. In the training process, *SSN* was the dependent variable where *WinPos*, *key_words*, *digits*, *noises* and *others* were the independent variables.

Table 4: Example of input data set for random forest classifier

| | Session_id | WinIndex | WinPos | key_words | digits | noises | others | SSN |
|-----|--------------------------------------|----------|--------|-----------|--------|--------|--------|-----|
| 0 | 16486.39.02.17.2016 _20-21-04.106 | 0 | 0 | 0 | 2 | 0 | 3 | 0 |
| 1 | 16486.39.02.17.2016 _20-21-04.106 | 1 | 5 | 0 | 2 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 362 | 16486.3.02.19.2016 _08-13-41.101 | 76 | 380 | 0 | 1 | 0 | 5 | 0 |
| 363 | 16486.3.02.19.2016 _08-13-41.101 | 77 | 385 | 2 | 1 | 0 | 6 | 1 |
| 364 | 16486.3.02.19.2016 _08-13-41.101 | 78 | 390 | 2 | 6 | 0 | 4 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

The scikit-learn module, *RandomForestClassifier*, has many parameters that can be varied to change the way that the data is classified. For this task, the parameter which was deemed to be most influential to the performance and accuracy of classifier is the parameter *n_estimators*, which affected the number of decision trees used.

From the 600,000 windows, two thirds were used to train the parameter, with the remaining one third used to test the accuracy of the predictions. The *train_test_split* and *accuracy_score* functions from sklearn was used to achieve this. Initially, a very high accuracy score was being predicted at around 99%. This was however due to the large number of 0s being correctly predicted. The more informative accuracy is the proportion of 1s, representing SSNs, that are correctly predicted, which is a far lower but more realistic score.

To optimize for the parameter *n_estimators*, an iterative procedure was used which recorded the number of decision trees used and a corresponding accuracy score.

6.1 Results

Table 5 shows the three outcomes of the random forest classifier.

Table 5: Outcomes of random forest classifier

| | 1st attempt | 2nd attempt | 3rd attempt |
|---|-------------|-------------|-------------|
| Maximum Accuracy | 21.85% | 22.06% | 21.96% |
| Number of Decision Trees corresponding to Maximum Accuracy | 29 | 41 | 11 |
| Minimum Accuracy | 19.98% | 19.46% | 20.19% |
| Number of Decision Trees corresponding to Minimum Accuracy | 1 | 15 | 5 |

The results in table 5 show the highest and lowest accuracy and the corresponding number of decision trees. Because of the random property of the random forest method, and the differing training and testing sets for each iteration, each outcome has different value, but the accuracy of the classifier is consistent at around 22%.

6.2 Discussion

The accuracy attained was deemed to be unsatisfactory. There were several reasons for this and ways that the random forest method could be improved are discussed below.

The training data that was provided was not accurate. The errors in the given data would cause the random forest classifier to be trained incorrectly. To rectify this problem, checking the training data manually could be an option. The downside to this is the time required to do this would be excessive.

The feature selection process also needs to be improved, since there could be more relevant information about social security numbers than what was used for this method. In this case, the selected information about words and word position were used as the features to create the input data set. Potentially start time (mS), end time (mS), and word volume may be helpful to locate the social security numbers.

The arguments that were passed to the parameters of the classifier could be further optimised. The python module *RandomForestClassifier*, contains a number of parameters that could be used to alter the classification process. In this case the parameter *n_estimators* was chosen, which dictates the number of decision trees used to optimize our model. As for the other parameters, setting and optimizing *min_samples_leaf*, the minimum number of samples required to be at a leaf node whose default is 1, could be used to achieve a better outcome. However, there is a potential problem that the random forest method requires higher performance of CPU, using more arguments leads to a reduction in performance.

7 Comparison of Methods

Due to the individual requirements for each of the methods, any comparison of them will be flawed to some extent. Overall the method which performed the best was the rule based system, achieving better recall values than both the SVM and random forest. It did, however have a lot more false positives than the SVM. Due to the nature of redacting sensitive information, a small number of false positives was deemed to be acceptable if it helped increase the recall, however the precision of the range method for the rule based approach was unacceptably low at approximately 0.05%. The SVM was effective in not falsely predicting SSNs, with only one in 200 windows being incorrectly tagged. The SVM performed very poorly on p-value ranges greater than four, an issue that was not present with the other methods with the rule based method actually returning better recall values for these ranges. This is clearly a big limitation of the SVM as it consistently fails to predict a full SSN when present. The random forest seems to be more consistent than the SVM, but with a lower accuracy of around 22%. This is however looking at the entire data set, whereas the SVM accuracy is only looking at the most suitable files.

8 Conclusion

The results were a bit disappointing. It was hoped that the machine learning methods would outperform the rule based method, but this was not the case. Without some form of manual checking, neither the SVM or the random forest methods would be suitable for use. The rule based method proved to be the most effective, but due to its low precision scores it too would benefit from some manual checking of its results.

The first action that would be taken to improve the results would have to be manually inspecting the transcripts to get as close to perfect data as possible. This would most likely benefit the machine learning

methods the most as a large amount of pristine data would greatly improve the classifications.

More work would need to be done to understand why the SVM behaves as it does when looking at larger p-value ranges and extra features. There is a lack of clarity about how the SVM works, and why it performs worse with certain features such as absolute position of the window.

It would also be beneficial to try out some other classification methods such as Naive Bayes to see if they performed any better than the current techniques.

9 References

1. Caroline. P. (2009). The Story of the Social Security Number [online].
Available: <https://www.ssa.gov/policy/docs/ssb/v69n2/v69n2p55.html>
2. K.P. Murphy. Machine learning: a probabilistic perspective. Cambridge, MA: MIT Press. 2012
3. Zhao. X, Shi. Y, Nui. L, Kernel based simple regularized multiple criteria linear program for binary classification and regression, Intelligent Data Analysis, vol. 19, pp. 507-527, 2015.
4. C.D. Manning, P. Raghavn and H. Schutze. Introduction to Information Retrieval. Cambridge University Press. 2008.
5. Ireson-Pain. J. what is a rule-based system? [online].
Available: <http://www.j-paine.org/students/lectures/lect3/node5.html>
6. Hayes-Roth. F, rule-based systems, communications of the ACM, vol. 28, pp. 921-932, 1985.
7. N. Christianini. J. Shawe-Taylor. An introduction to support vector machines and other kernel based methods. Cambridge University Press. 2008.
8. C.M. Bishop. Pattern Recognition and Machine Learning. Springer. 2006
9. Joachims. T, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, University of Dortmund.
Available: <https://www.cs.cornell.edu/people/tj/publications/joachims.98a.pdf>
10. Benyamin. D. A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System [online] (2012).
Available: <http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics>
11. Scikit-Learn. (2017). Ensemble Methods [online].
Available at: <http://scikit-learn.org/stable/modules/ensemble.html#forest>.