Identification and Classification of Sexual Predatory Behavior in Online Chat-Room Environments

Norbert Eke
UBC Okanagan
Kelowna, BC, Canada

Abdallah Mohamed
UBC Okanagan
Kelowna, BC, Canada

Jeffrey Andrews
UBC Okanagan
Kelowna, BC, Canada

norberteke@protonmail.com {abdallah.mohamed, jeff.andrews}@ubc.ca

Abstract

According to the Crimes Against Children Research Center, one in five U.S. teenagers who regularly use the Internet have received an unwanted sexual solicitation via the web. There is an increasing danger in online environments such as chat-rooms, where online predatory behaviour is more and more frequent, and creates an unsafe environment to minors. This project aims to design an approach for online communities to enhance their members safety by detecting malicious conversations. This project joins the powers of computational linguistics with statistical machine learning to decipher the insight lying in conversations, then make predictions on whether or not a specific conversation should be flagged for containing sexual predatory behaviour. The contribution of this novel approach is 2-fold: firstly, it is able to capture the contextual details by putting an emphasis on insight that lies within the conversation, and secondly it contains a 2 stage classification system, which is highly flexible and customizable for detecting and classifying other malicious textual data.

1 Introduction

Intro to problem, scary statistics, then what has been done previously by others, then mention your own contributions, and finish it with significance.

2 Research Questions and Objectives

The main goal of this research project is to design an approach that can detect and classify online conversations as containing sexual predatory or non-predatory behaviour within the conversation. Such an approach would have to take into consideration the messy and difficult to interpret nature of online chat-room conversations, such as the conversations being full of misspelled words, slang, internet acronyms, inappropriate language and broken grammar being used. One of the major tasks in this project is the reconstruction of linguistic context of words and phrases being used in a conversation. Modern computational linguistics use complex deep learning language models to capture the semantic similarity between words and concepts, thus mapping the meaning of words and concepts to high dimensional word vector spaces.

This research project aims to join the powers of computational linguistics with statistical machine learning to decipher the insight lying in conversations, then make predictions on whether or not a specific conversation should be flagged for containing sexual predatory behaviour. Such a powerful approach could be applied in online gaming chat-room environments, where children could be disposed to being manipulated or preyed on by sexual predators. This projects sole purpose is to design an approach that could help online communities keep their members safer by integrating an automated chat-room filter going through the conversations and classifying conversations to different levels of danger.

This research project initially focused on answering the following questions:

Q1: How to make modern computational language techniques and models adapt to the difficult to interpret nature of online conversations?

Q2: How to extract meaning from online conversations, such that decisions could be made about the potential danger levels that lie in the contextual details of the conversation.

Q3: What kind of classification system and what statistical machine learning models can make a difference and predict whether or not a conversation contains sexual predatory behavior with as much accuracy as possible.

3 Algorithm Proposed

The inputs of this algorithm are a parsed CSV file containing all conversation data. The CSV file should have individual columns for conversation-id, conversation line number, author/ author-id for the line, text content in that line, and a conversation label (being a binary value of 0 or 1, 0 meaning the conversation contains no predatory behavior, 1 meaning predatory behavior).

3.1 Data Manipulation and Text Pre-processing

Given a CSV file containing all conversation data, one needs to parse each conversation to combine all lines within a conversation into one textual observation, then pre-process the text, and attach the appropriate labels to each conversation. For text pre-processing a few basic text cleaning techniques have been considered, like removal of extra white spaces and html tags or links, also removal of numerical characters. All characters have been converted to lowercase and the autocorrect (McCallum, 2016) library's spelling corrector was also used. Conversations containing less than 3 words have been removed. The pre-processed text output is a text file, in which each row contains a label, then the whole conversation.

3.2 Vector Representation of Words

In order to create quantitative attributes about the content of conversations, the use of vector representation of words was needed. One of the most famous deep learning models that allows the computation of vector representations of words is Google's Word2Vec.

Word2Vec is a deep learning model takes a text corpus as input and produces high dimensional vector representations of words as output. In one of Google's blog posts the algorithm is described as "it constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications"(Mikolov et al., 2013b). These word vectors are also called word embeddings, and they are the output of the Word2Vec model. The Word2Vec model is trained to reconstruct the linguistic context of words by placing the vectors of words used in the same context close to each other. Word embeddings are also useful, since they are "good at capturing syntactic and semantic regularities in language" (Mikolov et al., 2013c), by placing the vectors of similar words closer to each other. Figure 5 offers a visual explanation of how Word2Vec works and how it is used.

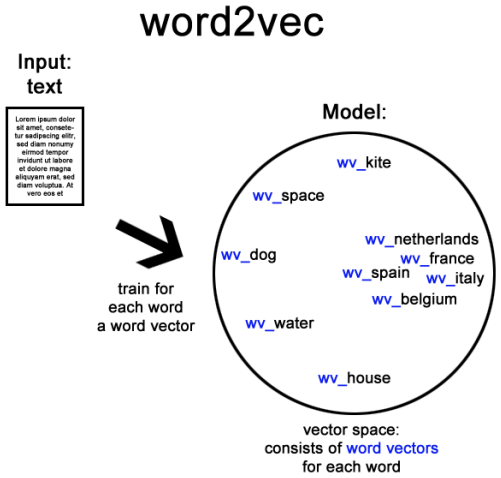

Figure 1: Detailed Overview of the Deep Learning Model Called Word2Vec. Image credits go to (L, 2015).

More details about the model architecture of Word2Vec can be found in Mikolov et al's paper. (Mikolov et al., 2013a).

If dimensionality reduction would be performed on word embeddings, these semantic regularities be visualized in so called low dimension word vector spaces. A good example of such vector space can be seen in Figure 2.

In terms of the algorithm proposed, these word embeddings were created by taking all pre-processed words used in all available conversations and train a large Word2Vec model, which converts each unique word into a high dimensional vector. Each unique word became part of a training text corpus that was used as input for the Word2Vec model. In order to train the Word2Vec model, Gensim (Řehůřek and Sojka, 2010), the python library's implementation of the Word2Vec was used. Gensim's implementation requires the text corpus and size of word vectors as parameters. $size = 400$, $workers = 20$ and a text file containing a pre-processed conversation for each line were the parameters given for the trained model. $size = 400$ was chosen on an experimental basis, as 100, 200, 300, 400 and 500 were considered, but word vectors of size 400 seemed to work best with the classifications models dis-

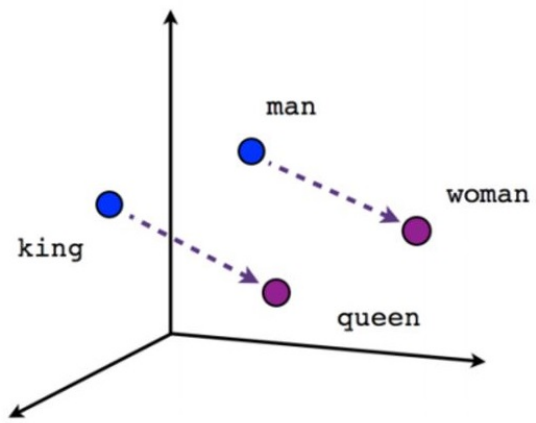

Figure 2: Photo Credit to (NSS, 2017). Three dimensional vector space, which contains the vectors 'Man', 'Woman', 'King', 'Queen'. 'Man' and 'Woman' are related to each other the same way as 'King' and 'Queen' are related to each other, since the vectors in between them describe the male-female relationship.

cussed later. The Word2Vec model created a vector representation of each unique word used somewhere within a conversation, thus a conversation can be represented as set of n word vectors, where n is the number of words present in that specific conversation.

3.3 Feature Extraction using Word Embedding Aggregation

The most important aspect of this algorithm is putting the emphasis on contextual details within the conversation. More specifically, the emphasis is on the insight found in the contextual details of a conversation. Having a 400 dimensional word vector represent each word and n word vectors represent a conversation provides the opportunity to dig deep into the contextual details of conversations and potentially find some insight about the content of the conversation.

Using the word vectors coming from the Word2Vec model is a good starting point for learning the context of a conversation, but a better approach includes an extra step, which deals with feature extraction. Traditionally, feature extraction is done with the help of dimensional reduction algorithms like Principle Component Analysis, or variable selection methods like LASSO. This project uses word embeddings to extract contextual details from textual data, therefore a word vector specific technique called "word embedding aggregation" (De Boom et al., 2016) was used. De Boom et al. noted that aggregating word embeddings through a mean, max, min function is one of the easiest and most widely used techniques to derive sentence embeddings.

In section 3.1 it was mentioned that each conversation is being concatenated into one very long sentence, which seemed like an odd choice for data manipulation. The real purpose of conversation concatenation is to set up conversations to be aggregated into "conversation embeddings", using Word2Vec embeddings and De Boom et al.'s word embedding aggregated sentence embeddings.

Based on De Boom et al.'s results, the algorithm makes use of coordinate-wise min and max functions applied to each word embedding within a conversation. This approach should work well, assuming that "conversation embeddings" behave the same way as sentence embeddings. If the vectors for the n words in the conversation are $v_1, v_2, \ldots, v_n \in \mathbb{R}_d$, then the computation of min(v_1,v_2,...,v_n) and max(v_1,v_2,...,v_n) are needed. What this means is that the coordinate-wise minimum vector and maximum vector of n word vectors within a conversation become 2 separate feature vectors.

Finally, as suggested in De Boom et al.'s work, the concatenation of these two feature vectors results in obtaining a coordinate-wise min-max feature vector for feature extraction purposes. In this project, the word embeddings are 400 dimensional, so the concatenation of coordinate-wise minimum and maximum vectors results in an 800 dimensional feature vector.

3.4 Two Stage Classification System

After performing feature extraction using word embedding aggregation, the 800 dimensional feature vectors can be fed into various classification models, to identify what machine learning model could separate conversations that contain predatory behaviour from the ones that do not contain it. Various binary classification models were considered: Linear Discriminant Analysis Classifier (?), Support Vector Machine Classifier(?), LASSO Classifier (?), Random Forest Classifier(?), Bagging Classifier (?), Generalized Boosted Classifier (Ridgeway et al., 2006), AdaBoost Classifier (?), kNN Classifier (?), Naive Bayes Classifier (?) and Ridge Classifier (?).

After fittings numerous classification models,

the results indicated that no model is able to minimize false positives and false negatives in the same time. Some models are better at minimizes one over the other, but no models can do it both, and still be predict accurately. Therefore, based on these observations a simple solution was to design two stage classification system, where two different classifiers would be trained.

3.4.1 First Stage Classifier

The first classifier reduces the number of false negatives (predatory conversations labelled as non-predatory), minimizing the Type II error. This first stage classifier is trained to predicts a conversation as containing or not containing predatory behaviour. Using the first stage classifier, the predicted groups can be interpreted the following way:

1. Conversations predicted by the first stage classifier as containing non-predatory behaviour: these are conversations that most likely do not contain predatory behaviour.

2. Conversations predicted by the first stage classifier as containing predatory behaviour: these are conversations that need to be looked at again, by a secondary classifier, just to filter out false positives.

A comparison of different first stage classification models' performance can be found in the Results section, but it is worth noting that the Linear Discriminant Analysis Classifier was chosen as the first stage classifier, purely based on cross validated error rate and precision measurements.

3.4.2 Second Stage Classifier

The second stage classifier filters through all conversations labelled by the first stage classifier as containing predatory behaviour, reducing the number of false positives (non-predatory conversations labelled as predatory), minimizing the Type I error. Using the second stage classifier, the predicted groups can be interpreted the following way:

1. Conversations predicted by the second stage classifier as containing non-predatory behaviour: these are conversations that possibly could contain predatory behaviour

2. Conversations predicted by the second stage classifier as containing predatory behaviour: these are conversations that most likely contain predatory behaviour

A comparison of different second stage classification models' performance can be found in the Results section, but it is worth noting that an AdaBoost Classifier (Freund et al., 1996) was chosen as the second stage classifier, purely based on cross validated error rate, precision and recall measurements.

Overall, the 2 stage classification system creates 3 groups, in decreasing danger levels: conversations most likely contain predatory behaviour, conversations possibly could contain predatory behaviour and conversations most likely do not contain predatory behaviour. This resulted from 2 groups found by the first stage classifier, then the second stage classifier breaks down one of the first classifier's groups into 2 separate groups. This results in a hierarchical relationship between the groups.

3.5 Overview of the Algorithm

1. Parse each conversation to combine all lines in each conversation into one textual observation, pre-process the text for each conversation, and attach the labels to it

2. Take all pre-processed words used in the whole dataset of conversations and train a 400 dimensional Word2Vec model, which converts each unique word into a 400 dimensional vector.

3. For each conversation aggregate a min and a max feature vector by taking the min/max of each dimension from all words present in a specific conversation. This will result in a 400 dim min feature vector and a 400 dim max feature vector for each conversation.

4. For each conversation concatenate the min and the max feature vectors, to obtain an 800 dim feature vector/conversation.

5. Perform Linear Discriminant Analysis on the whole dataset of conversation feature vectors, to classify conversations as predatory vs. no-predatory behaviour (First stage classifier).

6. Take all conversations that have been labelled to contain predatory behaviour and feed them into an AdaBoost model to filter out conversations that contain non-predatory behaviour,

but have been labeled as containing predatory behaviour (Second stage classifier).

7. Obtain 3 different groups of conversation, differentiated by their danger levels:
   - Group A = conversations most likely do not contain predatory behaviour: Linear Discriminant Analysis model predicted the conversation to contain non-predatory behaviour
   - Group B = conversations possibly could contain predatory behaviour: AdaBoost model predicted the conversation to contain non-predatory behaviour, but Linear Discriminant model predicted it to contain predatory behavior
   - Group C = conversations most likely contain predatory behaviour: AdaBoost model predicted the conversation to contain predatory behaviour

4 Case Study and Results

4.1 Sexual Predatory Conversation Identification Task

A controlled case study was conducted on a Sexual Predatory Conversation Identification task, similar to the Sexual Predator Identification tasks described in (Inches and Crestani, 2012). The data-set used for this controlled case study is the test set coming from Inches and Crestani's work. The test set was chosen over the training set, since the two set's ground truth labels are different, and only the test set's labels matched the controlled case study's needs.

This controlled case study differs from Inches and Crestani's Sexual Predator Identification tasks. Its aim is to focus on analyzing a whole conversation and deciding if it contains predatory behaviour, instead of identifying the predators among all users in different conversations or identifying the part of the conversations which are the most distinctive of predatory behaviour.

Inches and Crestani's test set provides an xml file, containing 155.128 conversations, alongside author and line meta-data. A ground-truth label file is also given, containing conversations id's and lines id's of those lines considered suspicious (of a perverted behavior) in a particular conversation. Given this data, the conversation xml file was processed into a csv file, with columns containing conversation id, line number, author, text and each row representing a line within a conversation. To get the data in the format described at the beginning of section 3, the conversion of predatory line labels to conversation level labels was needed. In order to do so, each conversation's line level labels were checked, and a new label of 0 (non-predatory) or 1 (predatory) was assigned if any of the conversation's lines contained predatory behaviour.

After these initial data processing steps, the whole algorithm described in section 3 was executed. The sample size of the conversations data-set is 155.128 observations, which is a large sample size. In order to speed up the computational process, a sample of 100.000 conversations was taken. In this random sample 99.450 conversations had the label 0, meaning no predatory behaviour has been identified within the conversation, and 550 conversations had the label 1, suggesting that predatory behaviour is present within the conversation. A seed number of 2017 was used in order to create reproducible results. The final results of this controlled case study were the classification results obtained from the 2 stage classification system. K-fold cross validation was used to assess the performance of various classification models on the feature vectors extracted from each conversation. $k = 10$ was chosen for the number of folds in the cross validation process.

4.2 First Stage Classification Results

In section 3.4 a list of potential classification models were provided. In the first stage classification process LASSO, Linear Discriminant Analysis, Support Vector Machine, Random Forest, Bagging, Generalized boosted models have been tested out. The Random Forest model consistently outperformed the Bagging model, and since both of them as similar, the Bagging model was dropped.

The two groups within the data-set are massively unbalanced, thus measuring the error rate of classifiers would be a mistake. Instead, recall and precision measurement for the predatory behavior labels, and F-scores are taken into consideration (Sokolova and Lapalme, 2009). F-scores, recall and precision for predatory labels is calculated as per Table 2 in Sokolova and Lapalme's work. In statistical analysis of binary classification the F-score is a harmonic mean of precision and recall and it has a parameter, β. This parame-

Classification Model	Average Recall	Average Precision	F1 Score
LDA	0.5223	0.9144	0.6648
SVM	0.7664	0.6585	0.7084
Random Forest	0.8241	0.2982	0.4379
LASSO	0.6739	0.6492	0.6613
Gr. Boosting Machine	0.8646	0.3018	0.4474

Table 1: Average Recall and Precision, and Overall F1-Score from the top 5 First Stage Classification Models.

ter is just a constant, that places more emphasis on either the precision or recall. For the purposes of this binary classification tasks a β of value 1 was chosen, which makes the F-scores F1-scores, and it equally weights recall and precision, while punishing extreme recall or precision values. Table 1 contains the average precision and recall measurements from each fold within the the cross validation process, also the overall F1-score for each model. All results are color coded based on cross-comparisons done between each model's average recall, precision and F1 scores. All values are percentage rates, and lower values within the table are red, while higher values are represented with the green color cells within the table.

As previously established, the first stage classifier needs to reduce the number of false negatives (predatory conversations labelled as non-predatory), therefore the model with the highest average precision value is the best first stage classifier. Random Forest has a large recall average value of 0.8241, but its average precision is only 0.2982, thus it is not an appropriate first stage classifier. Its F1-score is under 0.5 which makes it a really ineffective classifier for the overall classification task, but it can be concluded that the Random Forest model is good at detecting non-predatory conversations labelled as predatory. The Support Vector Machine (SVM) classifier has a lower average precision value (0.6585) than its average recall value (0.7664), therefore this model will not minimize the number of false negatives, thus it can not be used as a first stage classifier. Its F1-score is 0.7084, which is acceptable, but using this SVM classifier will result in balancing out recall with precision, without focusing on minimizing either, therefore this model is not perfect for providing the best results. The LASSO model's performance is very similar to SVM's case, having an F1-score of 0.6613. LASSO has almost equal precision (0.6492) and recall (0.6739) values, having the same problem the SVM model, not minimizing the number of false negatives, thus not being a good first stage classifier. The Gradient Boosting Machine Classifier's performance is really similar to the Random Forest model's. Both models are good at detecting non-predatory conversations labelled as predatory, average recall being 0.8646, but they both have really poor average precision performance (0.3018), thus they are ineffective at detecting predatory conversations accurately. Finally, Linear Discriminant Analysis (LDA) minimizes the number of false negatives, having the largest average precision value at 0.9144, therefore it was chosen as the First Stage Classifier. The LDA classifier is very good at identifying conversations that contain predatory behavior, but its very low recall rate of 0.5223 shows that this model can not solve all the problems at once. The low recall rate created a concern over the large number of false positives, which triggered the addition of a second stage classifier. The second stage classifier should be focused on filtering out false positives, while not creating a large number of false negatives. Figure 3 below shows the confusion matrix, after predicting for the left-out subset of observations during the cross validation process for the LDA model.

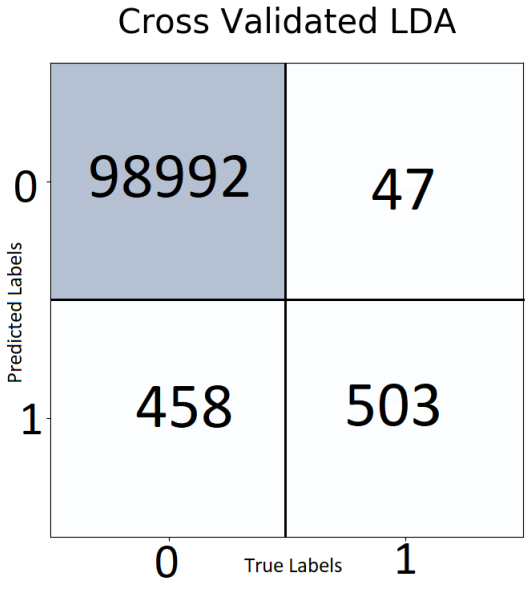

Figure 3: Confusion Matrix from the Cross Validated LDA model results.

As shown in Figure 3, the LDA model correctly classifies 503 out of 550 predatory conversation, out of a total of 100.000 conversations. This model minimizes the number of undetected predatory conversations, which makes this model so effective, with a large average precision value. However if one looks at the F1-score, it is noticeably low, only 0.6648, which is understandable, since the average recall is only 0.5223. It can be concluded that the LDA model in only good at precision, and it is ineffective at detecting non-predatory conversations labelled as predatory, where it mis-classifies conversations about half of the times. Models like Random Forest and SVM are good at this this specific task, so a second stage classifier in addition to the LDA model would help.

4.3 Second Stage Classification Results

For the Second Stage Classification process the following classifiers have been trained: LASSO, Ridge Classifier, Naive Bayes Classifier, k-NN Classifier, Linear Discriminant Analysis, Support Vector Machine Classifier, Random Forest Classifier, Bagging Classifier and AdaBoost Classifier.

The second stage classifier's emphasis is on filtering through all conversations labelled by the first stage classifier (LDA) as containing predatory behaviour, reducing the number of false positives (non-predatory conversations labelled as predatory), thus minimizing Type I error. Since the second stage classifier only filters through "flagged" conversations, there is no need to classify all 100.000 conversations, only those labelled by the LDA model as predatory (961 conversations, from Figure 3).

Table 2 contains the 5 best models built for the second stage classification process. The performance results from these 5 models are cross validated and average precision and recall measurements from each fold within the cross validation process are returned, alongside the overall F1-score for each model. All results are color coded based on cross-comparisons done between each model's average recall, precision and F1 scores.

The five best classifiers for the second stage classification process were SVM, Naive Bayes, LASSO, k-NN, and AdaBoost. Looking at the F1-scores, all 5 models have F1-values between 0.7 and 0.8, therefore all 5 models are compatitive at filtering out the mistakes that the LDA model does in the first stage classification. The Naive Bayes and k-NN classifier's performance is quite similar, by having similarly large average precision values, 0.9285 for Naive Bayes, and 0.8966 for the k-NN model, but they both lack performance in average recall, where the values are quite low, 0.5858 and 0.6279 respectively. Overall, both have over 0.7 F1-scores, but the conclusion is that both Naive Bayes and k-NN models are effective at precision, but not at recall, and that leaves quite room for plenty of mis-classifications. A good second stage classifier would need to be really effective at both precision and recall, since if the chosen classifier would be optimized to minimize just one of precision or recall, a third stage classifier would be needed. Looking at the SVM model, it has a 0.7533 F1-score, which is larger than the previous 2 models, and even the average precision is over 0.8. but the average recall value is just shy of 0.7. Unfortunately the SVM model's performance has more emphasis on precision, therefore choosing such a model as the second stage classifier would result in having the same problem as with the Naive Bayes and k-NN classifier models. Looking at the LASSO model's performance, it has an F1-score of 0.7574, which is better than all previous second stage classifier models. Its average recall is 0.7442, and precision is 0.7711, which makes it an effective and recall and precision-wise balanced model. This LASSO model would be a good choice for the second stage classifier, but there is one problem: the AdaBoost model outperforms the LASSO model. AdaBoost has an average recall of 0.7767 and average precision of 0.8091, making its F1-score 0.7926, just shy of 0.80. This model performed the best at filtering out mis-classifications that the first stage classifier made, and it achieves equally large and

Classification Model	Average Recall	Average Precision	F1 Score
SVM	**0.6934**	**0.8246**	**0.7533**
Naive Bayes	**0.5858**	**0.9285**	**0.7184**
LASSO	**0.7442**	**0.7711**	**0.7574**
AdaBoost	**0.7767**	**0.8091**	**0.7926**
k-NN	**0.6279**	**0.8966**	**0.7385**

Table 2: Second stage classification process, with cross validated results from the top 5 models, and their average precision, recall measurements, alongside F1-scores.

balanced performance on both recall and precision, making the second stage classifier a complete system for accurate classification, without needing a third classifier. Figure 4 below shows the confusion matrix, after predicting for the left-out subset of observations during the cross validation process for the AdaBoost second stage classifier model.

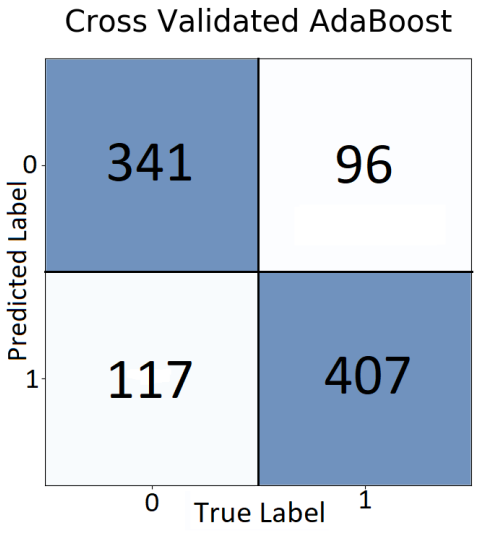

Figure 4: AdaBoost Second stage Classification Result's Confusion Matrix

As shown in Figure 4, the AdaBoost model correctly classifies 407 out of 503 predatory conversation. It also only mis-classifies 117 out of 458 non-predatory conversations. This AdaBoost model is as effective as it can be at minimizing errors made on both Type I and Type II errors, having large values for both average precision and recall. It can be concluded that the AdaBoost model is effective at filtering out mis-classifications made by the LDA first stage classifier.

4.4 System-wide Classification Results

In order to re-assure that LDA and AdaBoost are the best 2 models for the two stage classification system, Table 3 shows the Recall, Precision and F1-scores for LDA, the first stage classifier, combined with each possible second stage classifier. These values are obtained by fitting, then predicting using the first stage classifier, then using the second stage classifier to filter all observations that the first stage classifier flagged as potential predatory conversations.

The First stage classifier, LDA is the same model combined with different second stage classifiers. One can see how LDA + SVM, LDA + Naive Bayes and LDA + k-Nearest Neighbors has a lower recall value, but a much larger precision value. On the other hand, LDA + AdaBoost and LDA + LASSO have a much more balanced recall-precision ratio, since both of their values are closer to each other, meaning that both AdaBoost and LASSO are good at avoiding both Type I and Type II errors. If one only looks at the F1 scores, then LDA + AdaBoost outperforms every other combination of first stage-second stage classifier pairs, by having an F1-score of 0.7579. LDA + AdaBoost's largest F1-score is the main reason why the final 2 stage classification system is composed of the interaction of these 2 models. Figure 5 shows the overall classification system's confusion matrix, after both models have been applied, and their predictions have been merged.

System of Classifiers	Recall	Precision	F1 Score
LDA + SVM	0.6928	0.7545	0.7224
LDA + Naive Bayes	0.5859	0.8491	0.6934
LDA + AdaBoost	0.7767	0.74	0.7579
LDA + LASSO	0.7433	0.7055	0.7239
LDA + k-NN	0.6273	0.82	0.7108

Table 3: Recall, precision and F1 scores from First Stage classifier combined with each possible second stage classifier.

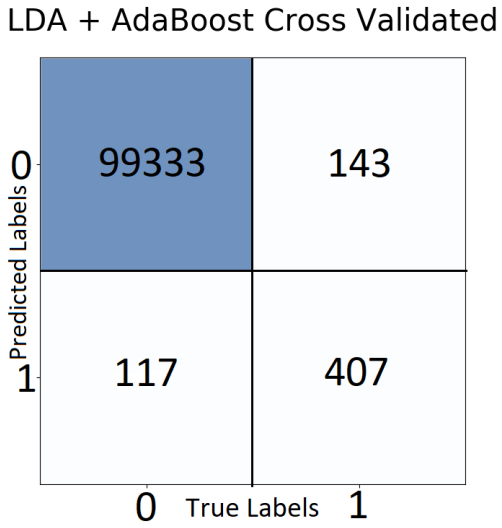

Figure 5: Overall Classification system: LDA + Adaboost

It is worth noting that the results from Figure

Group	Non-predatory Conversations	Predatory Conversations
A	98992	47
B	341	96
C	117	407

Table 4: 3 group classification of conversations, based on uncertainty levels of maliciousness

5 are only useful for calculating the final recall, precision and F1-scores for the overall two stage classifier system. Table 4 shows the proposed algorithm's 3 group classification of conversations based on uncertainty levels on maliciousness:

- Group A: conversations most likely do not contain predatory behaviour
- Group B: conversations possibly could contain predatory behaviour
- Group C: conversations most likely contain predatory behaviour

Group A comes from the first stage classifier's upper half of the confusion matrix, where the Linear Discriminant model predicted the conversations to contain non-predatory behaviour.

Group B comes from the second stage classifier's upper half of the confusion matrix, where the Linear Discriminant model predicted the conversations to contain predatory behavior, but the AdaBoost model filtered the conversations by predicting them not to contain predatory behaviour.

Group C comes from the second stage classifier's lower half of the confusion matrix, where both the Linear Discriminant and AdaBoost model predicted the conversations to contain predatory behaviour.

5 Discussion

One of the key aspects of the algorithm designed is taking into consideration the nature of data being analyzed. It is worth emphasizing that online chat-room conversations are filled with misspelled words, slang, internet acronyms, inappropriate language, broken grammar, short, messy and unstructured textual data. The most challenging aspect of this project is finding a way to interpret and analyze textual data. These linguistic challenges have been addressed in 3 different ways. Firstly, the careful selection of the adequate text cleaning techniques is considered. Secondly, special emphasis is put on the insight that lies within the contextual details of a conversation and thirdly, a domain specific feature extraction technique is being used to extract the essential details from each conversation.

The text cleaning techniques used were white-space, HTML tag, hyperlink and numeric character removal, lowercase conversion, autocorrect). Surprisingly, extensive white spaces, html tags and hyperlinks show up in Inches and Crestani's sexual predatory conversations dataset quite often, and since Word2Vec is not built to extract meaning from random html tags and hyperlinks, their removal from the conversations was obvious. When using the Word2Vec model, it is considered standard practice to remove numerical values, since their interpretation depends on domain, country of origin and other contextual details that often impossible to retain from the conversation. The lowercase conversion of all textual data is done for the sole purpose of not making discrepancies between uppercase and lowercase spellings of the same word. Conversations containing less than 3 words have been discarded, since no meaningful interpretation of such short conversation could be made. Arguably the most important text cleaning tool used is (McCallum, 2016)'s library called autocorrect, which is a spelling corrector. This tool is necessary for the as accurate as possible interpretation of broken language and misspelled words or expressions.

When it comes to the Sexual Predator Identification task, previous approaches included the identification of predators among all users in different conversations or identification of parts of conversations which are most distinctive of the predator behaviour. In order to come up with a unique approach, the algorithm is centered around detecting insight that lies within the contextual details of a conversation. More specifically, with the help of vector representation of words, and customized feature extraction, the vector representation of a whole conversation is obtained. These "conversation feature vectors" are composed of each conversation's contextual details detected by the Word2Vec model, then carefully selected and aggregated by a feature extraction process. The "conversation feature vectors" obtained in such a manner are the essential input for classification

models, which decide whether or not a conversation contains predatory behaviour. It is also worth noting that this approach considers the whole conversation as one large textual observation, without looking at discrepancies between each individual line in the conversation. To do so, the original labels from Inches and Crestani's dataset need to be parsed. The original labels are predatory line labels, which means that within each conversation, each line is labeled as predatory or non-predatory. The end goal of the label parsing process is to create conversation level labels, where each conversation contains a predatory/non-predatory label. This can be easily obtained by checking whether or not a predatory line is present within a conversation.

The feature extraction technique used in the algorithm is specific to only vector space representation of textual data. Good quote from (De Boom et al., 2016): "Short text messages such as tweets are very noisy and sparse in their use of vocabulary. Traditional textual representations, such as tf-idf, have difficulty grasping the semantic meaning of such texts, which is important in applications such as event detection, opinion mining, news recommendation, etc."

6 Future Works

7 Conclusion

Acknowledgments

References

De Boom, C., Van Canneyt, S., Demeester, T., and Dhoedt, B. (2016). Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Bari, Italy.

Inches, G. and Crestani, F. (2012). Overview of the international sexual predator identification competition at pan-2012. In *CLEF (Online working notes/labs/workshop)*, volume 30.

L, K. (2015). Graphic representations of word2vec and doc2vec.

McCallum, J. (2016). Spelling corrector python 3 library.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., and Le, Q. (2013b). Tool for computing continuous distributed representations of words. Google open source blog, Google Knowledge.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

NSS (2017). An intuitive understanding of word embeddings: From count vectors to word2vec. Retrieved October 6.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Ridgeway, G. et al. (2006). gbm: Generalized boosted regression models. *R package version*, 1(3):55.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.