Methodologies for the Management, Normalization and Identification of Sexual Predation of Minors in Cyber Chat Logs

John Sekeres

A Thesis
In the Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Computer Science
at Concordia University
Montréal, Québec, Canada

August 2022

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: John Sekeres

Entitled: Methodologies for the Management, Normalization and Identification of Sexual Predation of Minors in Cyber Chat Logs

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. René Witte

_____ Examiner

Dr. Brigitte Jaumard

_____ Supervisor

Dr. Olga Ormandjieva

_____ Co-supervisor

Dr. Ching Y. Suen

Approved by _____

Dr. Leila Kosseim

Chair of Department or Graduate Program Director

_____

Dr. Mourad Debbabi

Dean of Faculty

Date August 29, 2022

Abstract

Methodologies for the Management, Normalization and Identification of Sexual Predation of Minors in Cyber Chat Logs

John Sekeres

Neural networks based on the Transformer architecture have shown great results in tasks such as machine translation and text generation. Our contribution provides a methodology for an AI agent capable of Sexual Predator Identification (SPI) based on the classification capabilities of models built on the Transformer architecture. Results are comparable to existing state-of-the-art methods, with a $F_{0.5}$ score of 92.5% for predator identification on the PAN2012 test dataset consisting of 2,004,235 lines of text. Practical considerations require an AI agent that can evaluate large numbers of chats quickly. In that regard the Transformer based AI agent is able to evaluate over 2 million lines of text in under 6 minutes on a modestly configured workstation.

An AI agent by itself does not provide a complete solution to sexual predator identification. In an effort to give practical value to an AI agent, we address the vitally important but often overlooked issues of chat management and normalization. Our contribution provides a methodology for efficiently transforming raw chats from a native format into a consistent 'normalized' format suitable for analysis. We define a methodology to the problem of managing large numbers of chats, converting/normalizing 10,000 documents in a dataset in under 3 minutes on a modestly configured workstation. We present a software-based solution that among other things brings together chat management, normalization, and AI based analysis into a cohesive, productive environment that law enforcement can use to identify and build a case against suspected predators.

Acknowledgments

I would like to thank Dr. Ching Y. Suen and Dr. Olga Ormandjieva for their support and guidance. Their dedication to exploring new ideas, and encouraging the practical use of knowledge, has been the guiding inspiration for the work done in this thesis.

I would like to thank SunLife for sponsoring me. I am thankful that I have the privilege of working everyday with some of the finest and smartest people I have met in my professional career. A special thanks to Carmelo Mangiola, Peter Bourazanis, and Phil Gibson for their support in this endeavor.

I would like to thank Joanie Hamel from the Sûreté du Québec, who has always provided us with help and support. Joanie has gone above-and-beyond her duties towards helping protect victims of crime. Her devotion is inspirational.

Finally, I would like to give the warmest thanks and gratitude to my lovely wife Assunta and my children Cinzia, Anthony and Alexia. They provided encouragement and shown great patience with me along every step of this journey.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Social media provides a powerful platform for individuals to communicate globally. This capability has many benefits, but it can also be used by malevolent individuals for nefarious reasons.

Since 2014, the number of police-reported online child sexual exploitation and abuse incidents has generally been on an upward trend. In 2020, the overall rate of online child sexual exploitation and abuse was 131 incidents per 100,000 children and youth compared with 50 incidents per 100,000 children and youth in 2014 [1].

Reading through documents searching for sexual predation of minors is a daunting task, but this is precisely what law enforcement is faced with. With limited resources the focus is usually set to searching for evidence to build a court case against a suspect. This requires great effort, as it involves reading documents found on a suspects storage devices and on social media platforms known to be used by the suspects. This task takes away resources from identifying unknown predators before they commit future crimes.

This thesis describes the tools needed by law enforcement to be able to manage and analyze large numbers of chats, and furthermore, we present a methodology for how these tools might deal with;

1. Transforming raw chats into a normalized (i.e., consistent) format, suitable for further analysis.
2. Automatically detecting suspicious conversations and authors.
3. Bringing together all the required functionality into an application that law enforcement can use to analyze many chats.

1.1 Motivation

Technology is in some sense making it easier for predators to find and lure victims. This creates an environment where minors are often left vulnerable to predation. The motivation of our work is to help protect minors from these threats without excluding them from utilizing the benefits of the underlying technology.

Threats can come from all sorts of different mediums, however, there often exists with reasonable likelihood a digital footprint somewhere of a predator attempting to lure his/her victim. Often this interaction is in the form of a conversation, i.e., chat.

It is our assessment that having a set of tools that can quickly and intelligently identify incriminating chats offer a partial but meaningful impact at protecting victims.

1.2 Problem Statement

Chats are basically short text-based conversations between two or more people. The dialog often takes a form similar to a spoken conversation.

Online chats can come from many diverse places on the Internet. Chats will often be stored with different file encodings, for example: ASCII, Unicode. Also, within each conversation chat lines will be stored using different formats, for example, XML, HTML, plain text with different positions for the date/time, author identification and dialog positioning.

These raw formats make it difficult to analyze chats either manually, or by computational means without first transforming the chat data into a consistent format. As a result, one of the problems addressed in this thesis is the process of transforming raw chats into a consistent 'normalized' format.

Manual analysis of chats presents a big problem for law enforcement due to the amount of data they need to read through. We define and build an AI agent that can analyze and detect sexual predation of minors based on the text in a conversation, using a Transformer based model.

Normalization and automatic SPI analysis address the core functionality needed; however, it is not enough to make it practical to be used by law enforcement. There is also the issue of managing a great many documents, and of bringing together the normalization and SPI AI agent into a cohesive package. We address these issues by describing and building a software application which we name 'Resolute'.

1.3 Research Objectives

Our research objectives are founded on the issues presented in the last section *1.2 Problem Statement*, which essentially require 3 areas of exploration:

1. Normalizing chats into a consistent format suitable for analysis.
   We'd like to explore how to best address chat normalization, i.e., make it as easy and as fast as possible.
2. An intelligent system for automatically detecting suspicious conversations and authors.
   We'd like to expand on the existing body of knowledge in this area, and answer the question of how well a model can be created using an attention (Transformer) based deep learning model.
3. A practical system that allows for the management of a great many documents, and that brings together all the required functionality into a cohesive package.
   What would a system like that look like? How can it bring together the diverse

functionality into a cohesive, and productive environment? How can we organize the data so that it can handle millions of chats with lighting speed?

Aligned with the problems and areas of exploration defined above, the research conducted in this thesis is based on the following two hypotheses:

Hypothesis 1: the Transformer architecture used as a SPI classifier can obtain state of the art speed and identification of the sexual predation of minors in chats.

Hypothesis 2: a chat management system can be defined and built that enables users to manage, process, analyze and identity raw chats for sexual predation more efficiently than it takes to perform the task manually.

1.4 Research Approach

Our research included working with the cyber-crime unit of the Sûreté du Québec[1] (SQ) in order to get a sense of the challenges they face with regards to acquiring and identifying documents that contain incriminating evidence of sexual predation of a minor. The following is a partial list of the most relevant challenges:

1. Data acquisition.
   The SQ keeps details of their data acquisition procedures relatively secure. This process does however involve a lot of effort. Data acquisition is not the focus of this thesis, and it is assumed that the data has been acquired and is in need of analysis.
2. Data analysis.
   The SQ generally performs the analysis of documents manually. That is, they have a team of analysts that read through each document identifying which ones contain incriminating evidence. The evidence is later used to build a case against a suspected predator. This is a difficult and tedious task that takes a large amount of time and resources to perform.

In an effort to build a complete solution, it became apparent early on that before an AI agent could analyze the documents, raw chats would have to be normalized into a consistent format. Furthermore, because of the large number of documents that need to be processed, and based on the SQ requirement that only original documents could be used as evidence and could not be altered (e.g., normalization), this meant that some sort of document management facility needs to be considered. As a result of this line of reasoning the following list summarizes the order of the research approach taken:

1. Address the issues of document management. Details of this are given in *Chapter 6: Resolute*.
2. Address the issues of normalization. Details of this are given in *Chapter 4: Chat* Normalization.

[1] Sûreté du Québec is our provincial law enforcement agency

3. Address the issues of SPI. Details of this are given in *Chapter 5: Sexual Predator Identification.*

Our approach allowed us to work iteratively with the SQ. The idea is that by building the software early on in the process and putting it into the hands of the user, it would be possible to work out small issues before they became big problems. Also, this would give the SQ some time to use the software to build a dataset of French chats that we could later use to train the SPI model to detect French speaking predators.

1.5 Contributions

The goal of our work is to build a practical solution that will help law enforcement in their fight to protect minors from sexual predation. Part of that effort includes the analysis of many documents for incriminating evidence. This activity entails document management, document preparation and intelligent automated analysis of the documents. To that end, some notable contributions of our work include:

- A method of storing and organizing many documents in a secure, efficient manner so that original documents remain unchanged, while allowing copies of them to be edited as needed.
  This task has not been sufficiently addressed in other papers dealing with SPI.
  Our thesis describes a system that allows the data to be stored in a database management system, allowing for fast secure access to the data. Additionally, we show how to store data in a manner that allows the documents to be normalized and edited without altering the original document.
- A method towards fully automated normalization and the evaluation of the 'goodness' of text that is normalized.
  Normalization is a necessary part of SPI and has not been sufficiently addressed in other papers dealing with this task.
  Our thesis brings fully automated normalization closer to realization by addressing the issues involved with normalizing documents using templates and the scoring of normalized documents for how well they have been normalized.
- The effectiveness of a Transformer based model architecture using pre-trained word-embeddings as a classifier to automatically identify sexual predation of minors in text.

Chapter 2: Literature Review

2.1 Chat Normalization

Normalization in the context of this thesis, refers to the process of transforming the file encoding and structure of a raw chat into a text-based, standardized format.

Online chats can come from many diverse places on the Internet, for example; instant messaging, social media, chat rooms, Internet forums, computer storage (e.g., forensics), etc. In each case chats will be stored with different file encodings, for example: ASCII, Unicode, HTML, Word, PDF, XML, JSON, etc.

Also, within each conversation the chat lines will contain different positions and formats for the date/time, author identification and dialog positioning. These raw formats make it difficult to analyze the chats manually, and impossible to analyze by computational means without first transforming the chat data into a consistent format.

Once chats are normalized it is possible to present the chats for manual analysis in a pleasant and readable form (REQ9). More importantly the chats can be analyzed automatically by sufficiently intelligent Natural Language Processing (NLP) models (REQ6-REQ8).

Under normal circumstances custom programs or scripts are needed to transform the raw chat into a consistent format suitable for analysis. In our case the end-user of Resolute is not expected to know how to do this (CON1). This implies that we need to define a method that can do this work automatically, or at least provide a mechanism that can facilitate this process for the end-user (REQ3).

Unfortunately, not much research on how to normalize raw chats automatically is available. As a result, we have initiated research on the topic of normalizing raw chats automatically in this thesis.

2.2 Sexual Predator Identification

The task of SPI in chats using AI systems has been addressed in several other papers. The most successful approaches are summarized in the following table:

Approach	Precision	Recall	$F_{1.0}$	$F_{0.5}$
M. Vogt, et al. (BERT/NN) [2]			0.98	
Pastor Lopez-Monroy et al. (MulR/SVM) [3]			0.97	
M. Wani, et al. (CBOW/Random Forest) [4]	0.97	0.94	0.95	0.96
D. Liu (Sentence Vectors/LSTM) [5]	1.0	0.81	0.90	0.96
M.A. Fauzi, et al. (TF-IDF/Ensemble) [6]	0.96	0.86	0.90	0.93
E. Villatoro-Tello et al (CBoW/NN) [7]	0.98	0.79	0.87	0.93
snider12-run (no info available)	0.98	0.72	0.83	0.92
P. Borj, et al. [8] (SVM, Random Forest, Naïve Bayes)	0.92	0.89	0.91	0.91
H. K. Patrick Bours (NN/SVM/NB/Ridge) [9]	0.89	0.87	0.88	0.89
M. Ebrahimi (CNN) [10]	0.92	0.74	0.82	0.87

J. Parapar et al. (TF-IDF/SVM) [11]	0.94	0.67	0.78	0.87
C. Morris and G. Hirst (morris12, SVM) [12]	0.97	0.61	0.75	0.87
G. Eriksson and J. Karlgren (eriksson12) [13]	0.86	0.89	0.87	0.86

Table 2-1: Sexual predator identification systems

The PAN2012 competition [14] [15] opened up the doors for research in SPI by supplying an extensive labeled training and testing dataset (see *3.1.1 PAN-2012 Dataset*). In addition, the competition detailed a set of performance metrics (e.g., f-score – see section 5.8) to use as a basis for comparison between the different approaches. The SPI systems summarized in *Table 2-1* all utilize the PAN2012 dataset and scoring in one form or the other. This provides a rich environment for analysis.

Based on the existing work the following sections provide a description of the essential topics related to AI based SPI, and how our approach was influenced by the work already done.

2.2.1 Overview of the Top Approaches to SPI

Elements of a Transformer based model for early SPI detection (eSPD) has been explored in the paper '*Early Detection of Sexual Predators in Chats*' by *M. Vogt, et al.* [2]. Their approach uses a pre-trained BERT [16] language model as input to a linear classifier. In contrast, our models use traditional language models (learnt from training data, Word2Vec [17] and GloVe [18]) and a Transformer encoder for classification. Although eSPD is a form of SPI the goal of eSPD is to be able to detect the sexual predation of minors as early as possible, i.e., detection during an ongoing chat. Because of limitations of the PAN2012 dataset with regard to eSPD (e.g., chat timestamps do not include dates) *M. Vogt, et al.* built their own dataset (PANC) from PAN2012 (non-predator chats) and ChatCoder2 (predator chats). Nonetheless they did include experiments that compared their model to 'conventional SPI' (i.e., PAN2012 competition). Details of the SPI detection setup in the paper is lacking; however, their conventional SPI experiments did use the PAN2012 dataset filtered using the methods used in *E. Villatoro-Tello et al.* [7] (VTPAN). The VTPAN dataset filters out 90% of the chats from the PAN2012 dataset (see 3.1.2 VTPAN Dataset). Filtering proved to be very effective for *E. Villatoro-Tello et al.,* which makes it difficult to know what percentage of the results reported by *M. Vogt, et al.* can be attributed to the filtering and which can be attributed to the pre-trained BERT model. As a result in this thesis we will include experiments with the full PAN2012 dataset and the VTPAN dataset for comparison.

The effectiveness of good word embeddings is evident in the results reported in the paper '*Early Text Classification using Multi-Resolution Concept Representations*' by *Pastor Lopez-Monroy et al.* [3]. Their paper addresses eSPD, but unlike *M. Vogt, et al.* [2] their experiments use the PAN2012 dataset. It is not mentioned if they used all the data from the PAN2012 dataset or the filtered version VTPAN. The paper proposes and explores Multi-Resolution Representations (MulR) word-vector representations. Which is a method of enhancing existing word-embeddings with extra generalizations. The model uses a SVM with a linear kernel to process the MulR embeddings. In comparison to the highest reported result for a SVM based model *J. Parapar et al. (SVM)* [11] ($F_{1.0}$ 0.78) in which they used TF-IDF unigram encodings, the result of $F_{1.0}$ 0.97 in

Pastor Lopez-Monroy et al. (MulR(Temporal Variation Terms (TVT))) paper is exceptional. One can infer that given that the classification for both models was done by a SVM, that the substantial difference in performance must in some way be attributed to the effectiveness of the word embeddings.

The approach to SPI taken in the paper "*Sexual-predator Detection System based on Social Behavior Biometric (SSB) Features*", *M. Wani, et al.* [4] ($F_{0.5}$ 0.96) focuses on enhanced and extensive feature engineering, specifically targeting social behaviors. Input features consist of CBOW[2] encodings of the PAN2012 chat data along with encodings of social behaviors. The reported best result was from the CBOW encodings fed into a Random Forest model ($F_{1.0}$ 0.95). They also included for comparison, a SVM and Descision-Tree model. Given the same set of input features the Decision-Tree produced a result of $F_{1.0}$ 0.94, and the SVM $F_{1.0}$ 0.86. It is interesting that the performance of the SVM model is 9% less than the Random Forest model, and 8% better than the performance of the J. Parapar et al. (TF-IDF/SVM) [11] model. It is unclear from the data whether the good performance is coming mainly from the added social behavioral features or from the Random Forest and Decision Tree models. A baseline comparison would have been useful in this circumstance. Ideally, we want to keep feature engineering to an absolute minimum, and let the model learn which features are important.

Good performance was also achieved by *D. Liu (LSTM)* [5] ($F_{0.5}$ 0.96). The approach to SPI taken in the thesis "*Identifying Cyber Predators by Using Sentiment Analysis and Recurrent Neural Networks*" [5] uses a 2-phase approach where phase 1 is used to identify suspicious conversations and phase 2 is used to identify which of the authors is a predator. Similar to our 2-phase approach and that used by *E. Villatoro-Tello et al.* [7]. The models use sentence vectors to increase the training and processing speed of the LSTMs. The sentence vectors are built from using the hidden layers of a LSTM. The input embeddings are learnt from the training data. The output of the sentence vector is then used as input into another LSTM in which the output is fed into a dense linear network for classification. Approximately 20% of the PAN2012 data is filtered prior to processing. LSTMs have been proven to be very good at NLP tasks. It would have been interesting to know how a baseline LSTM based model (e.g., without filtering or sentiment scoring) would have performed, but such a baseline was not included. Also, because of the sequential nature of RNN based models, they are notoriously slow at training and evaluation, it would have been interesting to know the time that it takes for training and evaluation, and the gains achieved by implementing sentence vectors.

M.A. Fauzi, et al. (Ensemble) [6] also used a two-phase approach to identify suspect conversations followed by author identification. A different filtering strategy from VTPAN was used to filter out approximately 90% of the data. Filtering consists of using only conversations between two authors, and conversations that had at least 6 messages per author. In addition to an Ensemble model the paper evaluates eight other machine learning models; Multinomial Naive Bayes (mult nb), Bernoulli Naive Bayes (bern nb), SVM, Neural Network (NN), K-Nearest

[2] In their paper they referred to the encoding as 'CBOW (Count of Bag Of Words)', which sounds more like TF-IDF than the indicated **C**ontinuous **B**ag-**O**f-**W**ords acronym.

Neighbours (knn), Logistic Regression (lr), Random Forest (rf), and Decision Tree (dt). Input features for all models are encoded using BoW (binary), TF, and TF_IDF. The Ensemble consists of the top three performing classifiers (based on $F_{0.5}$), i.e., svm_tfidf, mult_nb_binary, and lr_binary. It is interesting that the Naïve Bayes and Logistic Regression models out-performed the Decision Trees and Random Forest models, especially considering how well they performed for *M. Wani, et al. (CBOW/Random Forest)* [4]. The Ensemble used two voting mechanisms for the final prediction; hard and soft. Hard voting consists of selecting the highest prediction, and soft consists of averaging the probabilities. The soft voting approach slightly outperformed the hard. In their paper [6] they state "*The experiment results also show that both ensemble strategies had a very impressive performance and obtained higher $F_{0.5}$-scores than any individual method.*". This does correlate with the general observation of Ensembles, as stated in the book "*Deep Learning with Python*" by François Chollet [19] "*If you look at machine learning competitions, in particular on Kaggle, you'll see that the winners use very large ensembles of models that inevitably beat any single model, no matter how good.*".

The methods and approaches to SPI discussed thus far were all performed post PAN2012 competition. The PAN2012 competition 1st place was awarded to *E. Villatoro-Tello et al*. (TF-IDF/SVM) [7]. Their approach to SPI included a two-phase method to identify suspect conversations followed by author identification. It is the same approach we take with our models and is the same as that taken by *D. Liu* [5] and *M.A. Fauzi, et al.* [6]. Filtering of the data is defined by the VTPAN dataset (see section: *3.1.2 VTPAN*). The model used is a Neural Network with input features encoded as CBoW with little preprocessing. A detailed analysis as to how the simple Neural Network using CBoW encoding is able to outperform other models with similar configurations was not provided in their paper [7]. One can infer that a good part of the performance of the model came from the filtering, based on the fact that most of the non-predator data was removed helping with the precision, and it also helped even out some of the huge imbalances in predator vs. non-predator data in the original dataset. A baseline with the unfiltered dataset would have proven useful. We will attempt to run some experiments later on to help answer the question of the effectiveness of filtering.

2.2.2 Two-Phased Approach

Some of the SPI systems summarized in *Table 2-1* use a two-phase approach in an effort to identify which authors in the PAN2012 datasets are predators [4] [5] [6] [7] [12]. Phase 1, deals with identifying which conversations contain dialog between a predator and a victim. Phase 2, deals with identifying which author is the predator from the set of conversations flagged in phase-1. Phase-1 basically acts as a filter, i.e., filtering down the number of chats and prospective authors.

Our approach uses the 2-phased approach for SPI, not only because it has shown to be effective [5] [6] [7] but mainly because we require both predications to be displayed to the user in Resolute as specified in requirements REQ6 and REQ7 (see *6.3.2 User Requirements*).

2.2.3 Preprocessing

The role of preprocessing is to transform/augment the raw input data in some fashion as to extract needed data, reduce dimensionality of the input and help the model perform and generalize better.

There are numerous ways that the input text can be preprocessed, and the decisions made form part of feature engineering. Some common preprocessing techniques include;

- Convert letter case, e.g., convert case of all letters to upper or lower case. This is very common and is used by virtually all NPL classification tasks to help reduce the dimensionality of the input.
- Stemming, i.e., removing common prefixes and suffixes from words. For example: 'works', 'worker', working' => 'work'. Experiments done by *C. Morris and G. Hirst* [12], with and without stemming shows that stemming is not very effective. The models explored in this thesis will not make use of stemming.
- Named entities and numbers to tokens. Named entities include things like: URL, email, proper names, addresses, phone numbers, ... Often it is not useful to know the exact values of entities or numerical values, in fact they may serve to disrupt classification. Transforming them to predefined tokens (e.g., #URL, #NUM) [5] [12], or remove them altogether (e.g., emoticons [5]) helps reduce the dimensionality of the input vectors. Experiments done by *C. Morris and G. Hirst* [12], with and without the transformations show that they are not very effective at SPI. Nonetheless, since named-entity recognition is already being done as part of anonymization (REQ5 – see *6.7.4 Anonymization*) we decided to include transformations of numbers and some basic named entities.
- Stop word removal. Stop words are words used in normal conversation but that do not have significance in NLP tasks such as classification. Examples of stop words include: 'an', 'the', 'to', or 'in'. In the case of chats different stop words need to be considered in addition to the standard proper stop words [10]. There is no compelling evidence that removing stop words helps and as a result our models will not remove them.

Some approaches make heavy use of preprocessing [4] [8] [12]. Others like *E. Villatoro-Tello et al*. [7] make specific statements about keeping their preprocessing to an absolute minimum. The realization is that it is difficult to know what information is important, and thus what type of preprocessing is beneficial. As a result, for our models we try to keep preprocessing transformations to a minimum (see *5.5 Preprocessing*).

2.2.4 Filtering

Filtering the is the process of removing conversations from the dataset that do not meet specific criteria. The idea with filtering is to remove noisy data that hinders the performance of the model. Deciding what data should be excluded from training and evaluation form part of the feature engineering of a model.

Filtering the PAN2012 dataset prior to training and during evaluation has proven to be a very popular method used in many of the SPI approaches discussed thus far [2] [4] [5] [6] [7] [8] [9]. *E. Villatoro-Tello et al*. [7] the 1st place competitors at the PAN2012 competition utilized heavy filtering in their approach (see *3.1.2 VTPAN Dataset*). Subsequently, it has become favored in many of the post-competition approaches to SPI.

Filtering helps balance out the big disparity between the number of predator chats vs. normal chats in the PAN2012 dataset. This can have a positive effect on recall. It also reduces the likelihood that a model will erroneously predict a normal chat as a predator (i.e., reduce false-positives helping precision).

In an effort to determine the effectiveness of using word embeddings with a Transformer (and Neural Network) without obscuring results with filtering we will train, optimize and test the models on the PAN2012 dataset. Subsequently, we will attempt to boost the results using VTPAN to get an idea of the impact that filtering has on the final test results.

2.2.5 Lexical, Behavioral & Social Features

"*Lexical features are those that can be derived from the raw text of the conversation.*" [14]. Some examples of lexical features: bi-gram [7] [11] [12] [13], TF-IDF [6] [7] [8] [10] [11], Named entities [12] [13], numbers to tokens [5], features obtained by the LIWC[3] tool [11]. More examples of lexical features can be found in the section *2.2.3 Preprocessing*.

Our models make use of some lexical features as defined in section *5.5 Preprocessing.* In terms of token positions (e.g., bi-gram), the Transformer architecture gets positional information from the attention mechanism.

"*Behavioural are all those features that captures the "actions" of a user within a conversation*." [14]. Some examples of behavioral features: "*the number of times a user starts a dialogue, the response time after a message of the partner in the conversation, the number of questions asked, the frequency of turn-taking, intention (grooming, hooking, ...), etc."* [14].

In the paper "*Identifying sexual predators by svm classification with lexical and behavioral features*" [12], *C. Morris et al.* go on to define a set of behavioral features to help deal with issues of the imbalance between the number of normal vs. predator chats, and to what they refer to as "symmetry-breaking", which is the tendency of predators and victims to use similar language when conversing. Incidentally, we noticed this symmetry in our experiments as well, i.e., our models having difficulty on some chats to unequivocally identify victims vs. the predators. This symmetry is both interesting and unexpected. Are these anomalies a result of an adult posing as a victim?

There is another category of features explored by *M. Wani, et al.* in the paper "*Sexual-predator Detection System based on Social Behavior Biometric (SSB) Features*" [4]. In their paper they

[3] "*calculates the degree to which people use different categories of words across a wide array of texts*" [14] - http://www.liwc.net/

attempt to capture the Social Behavioral Biometrics (SBB) traits of users with regard to SPI. SBB focuses on analyzing the social/emotional interaction and activities of users. Features are defined based on an emotion behavioral-based lexicon that are categorized as: fear, anger, sad, joy, surprise, disgust, trust, and anticipation.

Our models will not rely on behavioral or social features, since the intension is to also build a model based on French text (see REQ8), that would require a new set of behavioral/social features. Also, deep learning models are generally good at figuring out features of importance, so the expectation is that the Transformer based models will figure out which features are relevant.

2.2.6 Embeddings

Word embeddings are vector representations of words that map language to points in a hyperplane (see *3.3 Word Embeddings*).

The use of proper word embeddings can provide significant improvements in model performance. This is evident from the two top performing approaches explored by *M. Vogt, et al.* [2] and *Pastor Lopez-Monroy et al.* [3]. The top performing model *M. Vogt, et al.* uses a simple NN as a classifier, but incorporates embeddings pre-trained on a BERT [16] language model. BERT is a language model that uses Transformers to build rich pre-trained embedding vectors. *Pastor Lopez-Monroy et al.* use a technique (MUL) to augment the representational expression of existing embedding vectors.

With the exception of some experiments done by *Pastor Lopez-Monroy et al.* [3] with Word2Vec[4] and P. Borj, et al. [8] with GloVe, not much is known about the use of Word2Vec and GloVe embeddings with respect to using them for SPI. We hypothesize that Word2Vec and GloVe embeddings can produce superior results with our models, and will be explored in our experiments.

2.2.7 Classification

Many of the top performing approaches to SPI use Neural Networks, SVM, Naïve Bayes or Random Forest architectures for classification. Two of the approaches use deep learning models *D. Liu* [5] (LSTM) and *M. Ebrahimi* (CNN) [10].

The Transformer is a deep neural sequence-to-sequence (encoder-decoder) type network model that does not use recurrent or convolutional layers. Section *3.4 Transformer Architecture* describes the Transformer architecture. It has seen great success in machine translation (MT) [20, 21], and in AI generated text (GPT-3 [22]). *M. Vogt, et al.* [2] used a language model based on the Transformer (BERT [16]) for the pre-trained embeddings, followed by a linear neural network for classification. In this thesis we will explore a different configuration of the

[4] The pre-trained Word2Vec vectors used in their experiments were manipulated by MUL

Transformer from *M. Vogt, et al.*, using word embeddings fed directly into a Transformer encoder.

Chapter 3: Background

This chapter describes background information on some of the topics discussed in this thesis.

3.1 Datasets

The following 3 datasets are used as part of the work done and described in this thesis.

3.1.1 PAN-2012 Dataset

The PAN-2012 dataset was taken from the PAN Sexual Predator Identification 2012 [23] competition. It is widely used in other papers concerned with sexual predator identification, and provides a good basis for comparison of the effectiveness of the different approaches taken to identifying predators in on-line chats. The dataset is extensive, consisting of;

	# Conversations	# Chat Lines	# Authors	# Predator Authors
Training	66,830	879,136	97,448	142
Testing	154,955	2,004,235	218,213	254

Table 3-1: Properties of the PAN-2012 dataset

The number of normal authors to predators is highly imbalanced; however, this was done by design to reflect what is believed to be a realistic proportion of the true number of predators to normal users in on-line chats. The paper "*Overview of the International Sexual Predator Identification Competition at PAN-2012*" [14] details characteristics of the how the dataset was built. From a training and evaluation perspective however; this imbalance presents several challenges that we address throughout this thesis.

The PAN2012 competition defines two basic tasks:

1. Identifying the predators.
2. Identifying the distinctive chat lines of the predator behavior.

This thesis addresses task #1, and reserves task #2 for future exploration.

The PAN-2012 dataset consists of structured chats embedded in XML. Here is an example of a chat defined in the PAN2012 dataset:

```
<conversations>
  <conversation id="e621da5de598c9321a1d505ea95e6a2d">
    <message line="1">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:20</time>
      <text>Hola.</text>
    </message>
    <message line="2">
```

```
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:20</time>
    <text>hi.</text>
  </message>
  <message line="3">
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:20</time>
    <text>whats up?</text>
  </message>
```

Figure 3-1: Excerpt of the PAN-2012 training dataset

The training and testing data are stored in two XML files respectfully.

3.1.2 VTPAN Dataset

VTPAN is a filtered version of the PAN2012 dataset. The filtering is the same as that done by *E. Villatoro-Tello et al.* [7], which includes the following processing:

1. Conversations that had only one participant
2. Conversations that had less than 6 interventions per-user
3. Conversations that had long sequences of unrecognized characters (e.g., spam)

	Training		Test	
	Original	Filtered	Original	Filtered
Conversations	66,928	6,588	155,129	15,330
Authors	97,690	11,038	218,702	25,120
Predators	148	136	254	222

Table 3-2: VTPAN data counts

The filtering removes approximately 90% of the data from the PAN2012 dataset. As a result, some of the predators are removed from consideration as well. This ultimately impacts what can be achieved as a final score. It is a justified compromise, with which we agree with the statement stipulated by *Villatoro-Tello et al.* [7] "*Nevertheless, we think the information from interventions of removed predators was not enough to effectively recognize them as predators anyways*", i.e., some predators in the PAN2012 had so little information in them that it would be even impossible for a human to identify them.

3.1.3 Sûreté du Québec Dataset

The SQ cyber-crime unit assembled a dataset of local French based chats for use in our research. It consists of;

# Conversations	# Chat Lines	# Authors	# Predator Authors
962	35,610	155[5]	32[6]

Table 3-3: Properties of the SQ dataset

The chats were anonymized manually by the SQ, however, except for a small proportion of chats there was no consistency maintained with the author names across chats. That is, most of the authors for a given chat were either randomly named or in most cases simply named 'X' and 'Y'. This meant that conversations between authors spanning more than one chat sequence/file were lost. This configuration does not work well for the training of our SPI models, and so the dataset could not be used for training. The SQ could not justify the time needed to fix the existing dataset or build another one. Consequently, we took the approach of including an anonymization feature into Resolute (see section: 6.7.4 Anonymization). The idea was to provide the SQ tools that they could use to quickly assemble a French dataset, to be ready when we started training our models. This approach did not work out as planned, and so our work on the SPI models is based solely on the PAN-2012 dataset.

The SQ dataset did prove useful for our work on normalization. The dataset included a diverse set of documents in several different styles and formats, which as it turned out was perfect for what we needed in our normalization research.

Each chat is stored in its own file. The majority of the files are encoded as text and Microsoft Word. There are 8 different unstructured chat formats provided. The following is an example of 2 chats with different formats:

```
2011-05-11 12:20:18 Jacky : Bonjour a toi ça va bien??
2011-05-11 12:20:44 Doly : Oui je vais bien et toi ??
2011-05-11 12:21:07 Jacky : très bien merci
2011-05-11 12:21:19 Jacky : tu es jolie tu sais
2011-05-11 12:21:34 Doly : Merci
---------------------------------------------------------------
14:14
Jean X
allo ca va

14:14
Jeanne Y
oui toi

14:14
Jean X
pas si mal
ke fais tu de bon dans la vie
```

Figure 3-2:Sample of two chats from the SQ dataset with different formats

[5] Most chats anonymized the author names to 'X' and 'Y'
[6] A large proportion of the chats had authors named 'X' and 'Y'

3.2 Feature Engineering

It is often not reasonable to expect an NLP machine-learning model to be able to learn something useful from arbitrary data. The data needs to be transformed in a way that will make the models' job easier. Feature engineering is the process of applying non-learned transformations to raw data in order to use as input to a machine learning algorithm.

Care must be applied with feature engineering, as it is difficult to know what assumptions should be made about the data. Doing so usually requires understanding the problem in depth.

In early attempts prior to deep-learning, feature engineering was a critical part of the machine learning workflow. Modern deep learning architectures such as Transformers do not require extensive feature engineering, because they are capable of extracting useful features from raw data automatically. However, some feature engineering is useful, as it potentially allows you to use less resources, and less data. For example, correcting misspelled words in text will reduce the dimensionality of your input vocabulary. Also, learning features from raw text can potentially require a lot of data. In the instance where a large corpus is not available, some feature engineering can augment the value of whatever data you do have.

3.3 Word Embeddings

Word embeddings are vector representations of words that map language to points in a hyperplane. Words with similar meanings have similar representations.

Word embedding vectors are dense representations of words, versus, one-hot vectors for example, that are high-dimensional and sparse. Typically, word embeddings are anywhere between 25 to 1,024 dimensions, whereas one-hot vectors are the size of the vocabulary, for example 20,000 or more. Word embeddings pack a lot of information into much lower dimensions.

Word embedding vectors contain information about the structure of the language that is learned from the data, i.e., similar words map closely in the hyperplane, and directions in the embedding space are meaningful. For example, suppose we map the words; ["dog", "cat", "wolf", "tiger", “table”, “chair”] and map them on a 2D plane for simplicity:

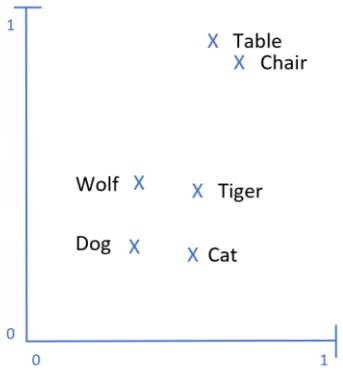

Figure 3-3: Example word embeddings on a 2D plane

The vector that goes from *dog* to *wolf*, when applied to *cat* gives us *tiger.* This can be interpreted as the vector that goes from pet to wild animal [24]. Similarly, the vector that goes from *dog* to *cat*, when applied to *wolf* gives us *tiger.* This can be interpreted as the vector that goes from canine to feline [24]. Arithmetic operations can be performed on vectors to give very interesting results. For example, *king – man = queen.* The distance of a vector (e.g., cosine similarity) between words can also show which word is semantically closer to a given word. For example, *dog* is semantically closer to *cat* than it is to *table.*

Word embeddings are generally computed by training a neural network in an unsupervised way on a large corpus. The concept was first explored by Yoshua Bengio in his seminal paper 'A neural probabilistic language model' [25]. The first to formalize a word embedding algorithm was Word2Vec, which was developed by Tomas Mikolov at Google in 2013 [17]. Since then, other schemes have been developed, for example GloVe, which was developed by Jeffrey Pennington, Richard Socher, and Christopher D. Manning at Stanford University [18]. GloVe enhances word embeddings by including word-word co-occurrence statistics into the representations.

Word embeddings can be trained as part of model, or pre-trained word embeddings can be incorporated into a model.

Training new embeddings as part of a model can be done in Keras using the *Embedding* layer for example. The benefit of training an embedding layer as part of a model is that the embedding representations are tailored to the domain specific task at hand. This however might not be optimal, since building effective embeddings usually requires a large amount of data.

An alternative to training new embeddings in a model is to use pre-trained embeddings built using Word2Vec or GloVe for example. You may choose to train from your own domain specific corpus, or download existing pre-trained embeddings built on other corpuses. These pre-trained models can be included into a new model. Note that if a new model incorporates pre-trained word embeddings, the new model should not alter the existing vector representations during training. The models used in this thesis use pre-trained GloVe word embeddings.

3.4 Transformer Architecture

The Transformer architecture was first introduced in the seminal paper "*Attention is all you need*" by *Vaswani et al.* [21]. It is a deep neural sequence-to-sequence (encoder-decoder) type network model that does not use recurrent or convolutional layers. It has seen great success in machine translation (MT) [20, 21], and in AI generated text (GPT-3 [22]). Neural attention has become one of the influential ideas in deep learning [19].

Attention is based on a powerful idea, that not all information processed by a model is equally important to the task at hand. The concept is rather simple, models should pay more attention to some features and less attention to others. Attention is a concept that has been used in NLP applications before, for example TF-IDF normalization where important tokens are assigned greater importance and irrelevant ones get assigned less importance [19].

Words have different meanings depending on the context in which they are being used. Self-attention fine-tunes the representation of tokens by considering the other tokens in a sequence. This creates context-aware token representations. That is, it essentially alters the word embedding representations (see *3.3 Word Embeddings*) so that the vectors are changed according to the context in which the words in a text are used. The paper "Attention is all you need" refers to this as '*Scaled Dot-Product Attention*'. We will refer to it as the *attention layer* and is essentially computed by:

```
embeddings = sum(values * distance(query, keys))
```

That is, for each token in *query*, compute the distance (see *3.3 Word Embeddings*) to every token in *keys,* and sum the weighted *values* to give a new set of embeddings. The *query, keys* and *values* can be different input sequences, but in our models they will all refer to the same input values. The following describes the formula in detail:

1. For a *given token* take the distance (e.g., dot product) between the token vector (e.g., word embedding) and every other token vector in the sequence. This serves as the strength of the relationships between tokens.
2. Process the distance scores through scaling and Softmax functions.
3. Compute the sum of all token vectors in the sequence, weighted by our distance scores. Tokens that are semantically closer to a *given token*, will contribute more to the total, while distant tokens will contribute less.
4. Repeat steps 1-2 for all tokens in the sequence. The end result is a matrix of scores, in which every token is scored against every other token in the sequence.

The resulting sequence of vectors represents the new token embeddings with attention included in the embeddings for each of the tokens.

The paper on "Attention is all you need" goes further to describe "*Multi-Head Attention*" which essentially describes *h* independent *attention layers* (referred to as a 'head') with an additional dense projection on the inputs to each head. The output of each head is concatenated back into a single output sequence. The dense projections enable each head to learn different groups of features for each token. Figure 3-4 illustrates a Multi-Head Attention layer with two heads:

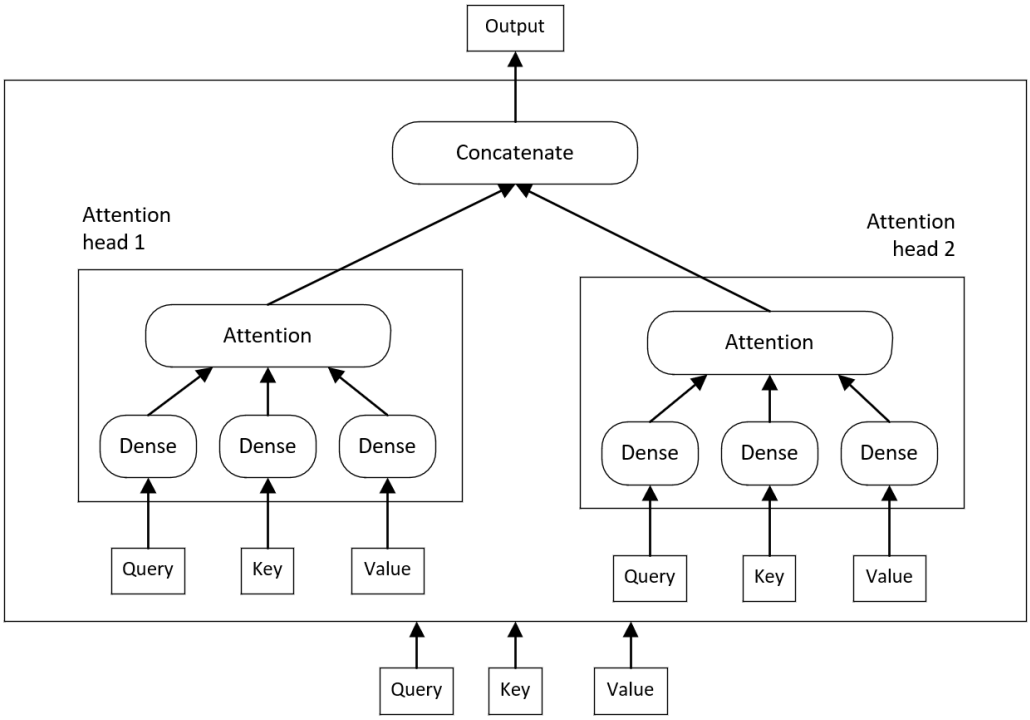

Figure 3-4: Multi-Head Attention layer with two heads

Another issue that needs to be addressed is that Transformers are sequence models, i.e., they process sequences without regard for the positions of the tokens. But, word order in text matters. The way Transformers handle this is to include token positions in the token embeddings and let higher order networks figure out what they mean. The positions cannot just be integer ordinals, since neural networks do not process large values well. There are different ways one can choose to encode the position of the tokens, as long as the values can be interpreted as the ordinal position of the token in the sequence and that are small enough not to overwhelm the network. For instance, the authors of "Attention is all you need" added to the token embeddings a vector containing values ranging from [-1, 1] that varied based on a cosine function.

The end-to-end Transformer architecture consists of two major parts: an encoder and decoder, as depicted in Figure *3-5: The Transformer - model architecture*.

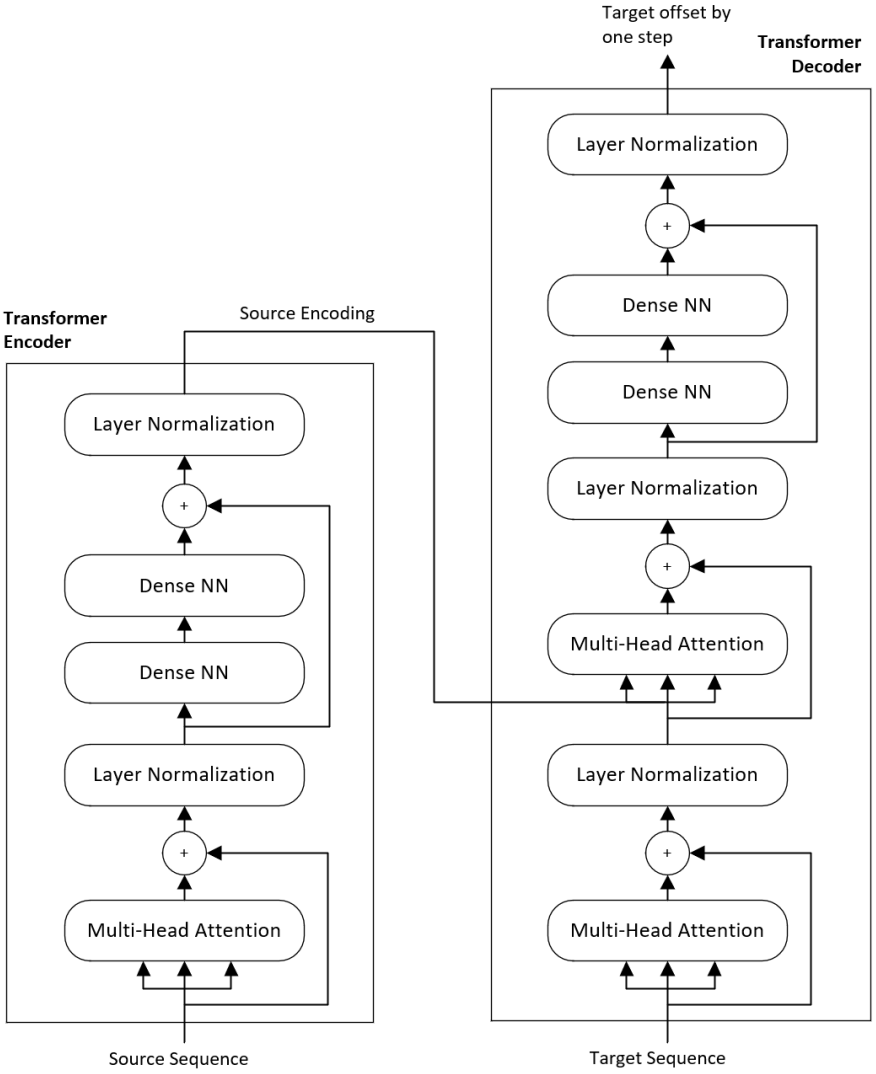

Figure 3-5: The Transformer - model architecture

The encoder is responsible for processing the source sequence into an encoding as described above. The decoder uses the source encoding to generate a translated version of the sequence. Important to note that the encoder can be used on its own for classification. Our models designed for SPI use the encoder in such a way. The encoder and decoder look very similar, except the decoder has an extra attention layer. The output of the decoder is the target sequence one step in the future, i.e., the next sequence from the original sequence.

As mentioned above, our SPI model makes use of the Transformer encoder for classification. It does not use the decoder part. Section *5.7 Classification* details the construction of the SPI model.

Chapter 4: Chat Normalization

Chat normalization refers to the process of converting the format and structure of a raw chat into a standardized text-based format. It is an essential process if we are to be able to automatically detect sexual predation of minors in chats.

In this chapter we discuss the issues with chat normalization and provide a practical process for normalizing without the need for writing custom programs.

Chats once converted from their native file encoding into text, come in 2 basic flavors:

1. *Structured chats*, i.e., chats that contain embedded meta-data, or data that describes the structure of the chat, but that is not related to the data in the header or dialog. An example of structured chats includes chat-data embedded in XML or JSON.
2. *Unstructured chats*, i.e., chats with no meta-data in them. Basically, chats that have only the required chat line header and the authors' dialog.

Each basic type of chat has its own set of normalization challenges, and will be discussed separately below.

4.1 Motivation

The ability to facilitate the normalization of raw chats is required for Resolute (see REQ3). Without normalization Resolute cannot function as a practical system (see NFR1). To the best of our knowledge a methodology for normalizing chats without having to write programs or scripts has not been thoroughly explored in other papers.

In an effort to make the process of normalization practical, this chapter presents a method of normalizing large numbers of chats while minimizing the need for writing custom programs or scripts.

4.2 Contribution

A systematic method of normalizing chats in a practical and efficient way, that provides users with the ability to analyze large amounts of chat conversations.

4.3 Workflow

There are essentially 4 steps in chat normalization (each step maps to REQ3):

1. Convert raw chats, i.e., natively encoded non-text-based chats (e.g., PDF, MS Word) into text encoded files.
2. Extract the chat header and author dialog. With unstructured chats this involves locating the positions and format of the date/time, chat line author and dialog. Often this can be

challenging as headers and dialog can span many lines, and often contain different formats from one line to another. For example, this is often the case when chats are acquired by copy/pasting from a web-site. In other scenario, structured chats that contain meta-data (e.g., XML, JSON), whereby the meta-data needs to be excluded from the final normalized chat.

3. Score the correctness of a normalized chat. If we are to make the process of normalization practical, it is important to be able to detect how good a chat was normalized. It is also a key part of being able to fully automate normalization.
4. Store the chat data into a standardized format that be retrieved for analysis later on.

In order to build a system that can streamline the SPI process and make it practical to work with, it is important that at a minimum these steps can be performed in a semi-automatic way (see NFR1).

4.4 Converting Raw Chats to Text

Raw chats refer to chat documents acquired before any processing is done on them. These documents can come from anywhere, for example: web-sites (copy pasted, screen scrapped), computer equipment (forensics), IRC chat logs, etc.

Raw chats also come in many different file encodings, for example; ASCII, Unicode, Microsoft Word, PDF, Excel, Google Docs, etc. Before any processing can be done on the documents, they need to be converted into text. This would in itself be a challenge to normalizing chats, but fortunately there are existing software utilities that can convert from various encodings into text, for example: Pandoc [26], Apache Tika [27]. Resolute makes use of Pandoc to convert raw chats into UTF-8 text based documents as part of the normalization process.

We refer to *raw chats* as those documents that have not yet been converted to text and *text chats* as those that have been converted into text but that not yet been normalized.

4.5 Normalizing Structured Chats

Structured chats refer to text chats that include meta-data or structure information about the chat embedded in the text. For example, XML, JSON, CSV (with headers), HTML, etc. Refer to *Figure 3-1: Excerpt of the PAN-2012 training dataset* for an example of a structured chat.

Normalizing structured chats consists of identifying and keeping the chat headers and text, and ignoring the rest of the data. The complication is that the information of interest (i.e., author, date-time, chat dialog) is interspersed with other data/meta-data that needs to be filtered out.

It is generally not possible to build a fully generalized automated normalizer for structured chats unless something is known about the structure; therefore, some form of manual intervention is required.

There are a number of ways to extract chat data from structured text chats:

- Manually remove the meta-data. This might be appropriate in a one-off situation, but is not practical for a large dataset.
- Utilize an existing utility program. For example, Pandoc supports the extraction of text from HTML, and JSON files amongst others. This only works for basic structures like the PAN2012 dataset. It will not work for more complicated structures like a HTML page, where you may find other text not related to the chat.
- Create a custom program or script that extracts the chat data from a complicated structure.

Resolute supports the normalizing of structured chats by the following features:

- REPLACE directive. This directive provides a simple and quick way of removing unwanted structure related text from a chat. For example, some of the chats in the SQ dataset contain simple one line session information inserted at different points in the chat. For Example: '*Session Start: Fri May 29 18:24:50 2009*' that need to be removed.
- PREPROCESS directive. This directive signals Resolute to invoke an external program to extract the chat information from a complex structured chat. For example, the PREPROCESS directive was used to execute an external program that removed the XML tags from the PAN2012 dataset.

The output of the REPLACE and PREPROCESS directives is an unstructured chat, which as we will see in the next section allows for a more systematic way of normalizing chats.

4.6 Normalizing Unstructured Chats

Unstructured chat types refer to chats that do not contain meta-data or information about the chat structure within the chat. We define an unstructured chat as:

- A text chat that contains only chat headers and conversation-based text.
- The chat header (date-time and/or author) for a given chat line precedes the authors text.

```
2019-08-13T02:29:57 <ekyle> we can see ES ingestion slowly falls behind later in the week
2019-08-13T02:30:49 <ekyle> ES ingestion may be unusually slow because the cluster is yellow
2019-08-16T13:10:12 <ekyle-pto> ES ingestion is still slow:
2019-08-16T13:10:14 <ekyle-pto> https://irccloud.mozilla.com/file/0fldkYX0/image.png
2019-08-16T13:11:20 <ekyle-pto> it seems ES lost some number of spot nodes, and ingestion
2019-08-16T13:27:32 <ekyle-pto> which requires more moves, and takes longer before the recovery
2019-08-16T13:31:57 <ahal> ekyle-pto: are you asking us to do something?
2019-08-16T13:32:05 <ahal> or just wait it out
```

Figure 4-1: Example of an unstructured chat

Figure 4-1 shows an example of an unstructured text chat. The header is composed of a date and time in a particular format, followed by the author id of the chat line enclosed in <>. The chat line text associated with the author who wrote the text follows the authors name, and ends with the beginning of the next header.

Figure 4-2 shows the chat in Figure 4-1 in normalized form.

```
08/13/2019 13:29:57, ekyle: we can see ES ingestion slowly falls behind later in the week
08/13/2019 02:30:49, ekyle: ES ingestion may be unusually slow because the cluster is yellow
08/16/2019 13:10:12, ekyle-pto: ES ingestion is still slow:
08/16/2019 13:10:14, ekyle-pto: https://irccloud.mozilla.com/file/0fldkYX0/image.png
08/16/2019 13:11:20, ekyle-pto: it seems ES lost some number of spot nodes, and ingestion
08/16/2019 13:27:32, ekyle-pto: which requires more moves, and takes longer before the recovery
08/16/2019 13:31:57, ahal: ekyle-pto: are you asking us to do something?
08/16/2019 13:32:05, ahal: or just wait it out
```

Figure 4-2: Unstructured chat in normalized form

Figure 4-3 shows an unstructured chat from the SQ dataset followed by its normalized form in Figure 4-4.

```
On September 3, 2012 10:54:01 PM PDT, Isabelle X wrote: Teer quii
?? :O

On September 3, 2012 10:54:51 PM PDT, Isabelle X wrote: Ahh Oui jme
rappelle dtoi xd



On September 3, 2012 11:02:54 PM PDT, Jean wrote: Okkk



On September 3, 2012 11:03:18 PM PDT, Isabelle X wrote: Speuux tu
que tu sort aek Pascale ?



On September 3, 2012 11:03:20 PM PDT, Isabelle X wrote: :)



On September 3, 2012 11:03:54 PM PDT, Jean wrote: Oausi
```

Figure 4-3: Example 2 of an unstructured chat

```
09/03/2012 22:54:01, Isabelle X: Teer quii ?? :O
09/03/2012 22:54:51, Isabelle X: Ahh Oui jme rappelle dtoi xd
09/03/2012 23:02:54, Jean: Okkk
09/03/2012 23:03:18, Isabelle X: Speuux tu que tu sort aek Pascale ?
09/03/2012 23:03:20, Isabelle X: :)
09/03/2012 23:03:54, Jean: Oausi
```

Figure 4-4: Normalized version of chat in example 2

Normalizing a chat consists of knowing the position and format of the date-time, the author id and the text. This can be done by applying a template to the chat data that describes the format of the chat header. As the system parses each token, it is compared against the template. If each token matches the template (which is also tokenized) then those series of tokens become the header. The remaining text after the header becomes the chat-line dialog until the next header is uncovered. If at any point there is a mismatch between the template and the current token, then the comparison resets to the 1st token of the template and the cycle repeats.

Resolute supports templated normalization of unstructured chats. For example, we can define the following templates:

```
1. NORMALIZE-HEADER-TEMPLATE: on #LONGMONTH #DAY, #YEAR #HOUR:#MINUTE #AMPM, #AUTHOR wrote:
2. NORMALIZE-HEADER-TEMPLATE: #NL #DAY/#MONTH/#YEAR #HOUR:#MINUTE #NL #AUTHOR #NL
3. NORMALIZE-HEADER-TEMPLATE: [#HOUR:#MINUTE:#SECONDS] #AUTHOR:

where:
#LONGMONTH = {January, February,..., December}
#DAY = 1-31
#MONTH = 1-12
#YEAR = 1990-3000
#HOUR = 0-59
#MINUTE = 0-59
#SECONDS = 0-59
#NL = new line
#AUTHOR = the author of the current line (the exact value is unknown at this stage,
          but tokens are concatenated until the terminating delimiter is detected)
```

Figure 4-5: Example unstructured chat normalization templates

Applying each template defined in Figure 4-5 to the following unstructured text chat:

Unstructured raw chat:

```
On April 17, 2013 6:35 AM, Rita B wrote:
Hello, how are you?
On April 18, 2013 8:37 AM, Fred S wrote: good and you
...
```

Normalized to:

```
04/17/2013 6:35:00, Rita B: Hello, how are you?
04/18/2013 8:37:00, Fred S: good and you
...
```

Figure 4-6: Sample of a normalized chat using template 1

Going through the sequence of applying each token in the chat to each token in the templates, we find that template #1 in Figure 4-5 matches the chat header of our example chat in Figure 4-6.

The following details the steps taken in order to conclude that template #1 is the best and proper match for the chat:

1. The header is expected to begin with the word ‘on’, so each token in our example chat is scanned until it comes across the 1st token ‘on’.
2. The next tokens expected are ‘#LONGMONTH #DAY’, which in our example is ‘April 17’ for the 1st header and ‘April 18’ for the second.
3. The next token expected is a comma.
4. The next set of tokens expected are: #YEAR #HOUR:#MINUTE #AMPM which match ‘2013 6:35 AM’ for the 1st header and ‘2013 6:35 AM’ for the second. Note that the colon separating #HOUR:#MINUTE in our template is expecting a colon in the chat header. If it’s anything but a colon the template is considered a mis-match.
5. The next token expected after “#AMPM” is a comma.

6. The next token expected is '#AUTHOR' which can contain more than 1 word. In our example the authors match 'Rita B' and 'Fred S'. Note that the authors name ends with the template defined next token. Which in our example is the token 'wrote'.
7. The next token expected is 'wrote' followed by a colon.
8. The remaining tokens following the header are considered the text associated with the current header, until the token 'on' is found. If the word 'on' is part of the text the next sequence of tokens will not match the template and thus will be considered as part of the text. If the token 'on' is the beginning of the next header, it will match the template and the steps above are repeated.

Note that when parsing for the author in step 6, the assumption is that the header ends with some sort of delimiter. In our example that delimiter is 'wrote:'. If a template ends with #AUTHOR (i.e., no ending delimiter), then the system does not know when to stop parsing tokens as part of the name, so it assumes the author name to be one token.

Template based normalization achieves very good results; however, it does require a user to specify a template. This makes template-based normalization impractical if it needs to be done for every chat, however; that can be mitigated by:

- For any given set of chats acquired from the same source, the format of the chats remain rather consistent. That is, one template can serve to normalize multiple text chats coming from the same source.
- A system that can apply a set of templates one-by-one against a given text chat and be able to choose (i.e., score) the template that gives the best results.
- Automating the creation of the templates. This refers to building the normalization template from the chat by analyzing the chat for the header. Technically it should be feasible, but this topic is reserved for future research.

A prerequisite for each of these mitigating points is that the system should be able to score the quality of the normalization done to a chat. For example, the system should be able to highlight which chats did not normalize properly, or take the highest normalized score to match the best template from a set of templates. The next section describes a method of how to score the quality of a normalized chat.

4.7 Scoring Normalized Documents

In this section we will be discussing how we propose to quantify the quality of a chat that has been normalized. Scoring the quality of a normalized file is important for a number of reasons:

- We may want to fully automate the normalization process by generating a template for a given chat.
- When normalizing many chats with a given template it is useful to know which chats a template did or did not apply to.
- When applying more than 1 template to a chat it is useful to know which one produced the best results.

Resolute uses scoring to select the best template from a set of templates, and it uses the score to let the user know when a chat did not normalize properly.

The scoring method applies the following rules when evaluating a chat after it is normalized. Each chat starts with a normalization score of 1.0, and is penalized a certain percentage depending on which rule is broken:

1. The date-time fields should contain only date-time data.
2. Penalize the score for every formal date-time specification in the text.
3. Every line in the chat should have an associated author.
4. Penalize the score the more authors there are from the expected average of 2 authors in a conversation[7].
   For example, a penalty of 0.0 if there are 2 authors, 0.01 if there are 3, 0.03 if there are 4, 0.08 if there are 5, etc.
   If the normalization is bad then we would expect the system to parse part of the text for #AUTHORS. This would generate arbitrary author names thus associating the chat with a large number of authors.
5. Penalize the score for every mention of the authors id in the text.
6. Subtract a small score if the line starts with a number.
   This is useful when a bad template places the date/time as part of the text.
7. Subtract from the score deviations of line lengths greater than the average length of 40 characters[8]. The more the deviation the greater the penalty.

Some rules can end up penalizing the score even though they are valid. For instance, rule #2 '*Penalize the score for every formal date-time specification in the text*'. It is perfectly acceptable for an author to specify a date in the text; however, you would not normally expect to see many dates in the text. In this case the net effect on the final score is minimal. If, however normalization is bad, there is a good probability that the date-time specification that belongs in the header would be placed in the text. The net effect on the final score would be substantial.

The consequence of these scoring methods on a normalized chat produces good results. From a set of 9 templates the process is able to identify the proper template for all 990 of the 997 chats in the SQ dataset. The 7 chats that did not normalize were conversations that did not specify a complete header. These were basically non-chat documents without an author or date-time entries.

4.8 Experiments and Results

We began our experiments by classifying the chats in the PAN-2012, and SQ datasets by their internal structures. A class is a set of documents that are of the same internal format, thus they all share the same normalization template.

---

[7] The PAN2012 dataset gives an average of 2.28 authors per conversation
[8] The PAN2012 dataset gives an average of 33.39 characters per line

The PAN-2012 dataset consists of 1 class of structured chats. Since this dataset consists of structured chats, it required pre-processing to extract the chats from the XML meta-data. Refer to *Figure 3-1: Excerpt of the PAN-2012 training dataset* for an example of a chat from this dataset. The pre-processing was done by a custom program invoked from within Resolute. Here is an example of a PAN-2012 chat after pre-processing in Resolute:

```
[2021-10-01 18:59] "edb259c0e0038f38bb200bc20c8cbf7e": guess we-spam-wikis took a
week off or so
[2021-10-01 19:07] "06cb330920ae58e1614c9145d983b3d6": just delete any old account
that's never done anything?
[2021-10-01 19:11] "e9fe2a8ed6a64844a5c024b6f688d024": Happy New Year, Whatwg!
[2021-10-01 19:12] "a11aabeeceeae6b8cb5d12ea06b56554": Eh, so there are two dates
that
```

Figure 4-7: PAN-2012 chat after pre-processing in Resolute

Note that the format of the pre-preprocessed output matches closely the standardized format for a normalized chat in Resolute but still requires a normalization template for the chats to be normalized properly.

The SQ dataset consists of 8 different unstructured chat formats (i.e., classes), file encoded in either ASCII text or Microsoft Word. Refer to *Figure 3-2:Sample of two chats from the SQ dataset with different formats* for an example of two chats with different internal formats.

We imported all the chats into Resolute from the PAN-2012 and SQ datasets, and converted them from their native file formats into text. The following table describes the 9 classes of documents with their corresponding normalization templates. More information on the template syntax can be found in section *6.7.2 Directives*.

Class	Format Template	# Documents
SQ1	`#NL #AUTHOR:`	1
SQ2	`#NL #DAY/#MONTH/#YEAR #HOUR:#MINUTE #NL #AUTHOR #NL`	4
SQ3	`#NL #HOUR:#MIN #NL #AUTHOR #NL`	3
SQ4	`#NL (#HOUR:#MINUTE) #AUTHOR :`	13
SQ5	`#YEAR-#MONTH-#DAY #HOUR:#MIN:#SEC #AUTHOR:`	3
SQ6	`[#HOUR:#MIN] <#AUTHOR>`	917[9]
SQ7	`[#HOUR:#MINUTE:#SECONDS] #AUTHOR:`	1
SQ8	`on #MONTH #DAY, #YEAR #HOUR:#MINUTE:#SECONDS #AMPM PDT #OR PST, #AUTHOR wrote:`	28
PN1	`[#YEAR-#MONTH-#DAY #HOUR:#MIN] "#AUTHOR":`	100[10]

Table 4-1: Datasets Class Templates

4.8.1 Normalization with Template Specification

In this experiment we applied the proper template to each class of documents and manually verified the score against the normalized chat.

[9] Our experiments used 100 randomly selected chats from the 917 chats
[10] Our experiments used 100 randomly selected chats from the PAN-2012 training dataset

Class	# Documents	Min Score	Max Score	Average Score
SQ1	1	0.96	0.96	0.96
SQ2	4	0.94	0.98	0.96
SQ3	3	0.99	0.99	0.99
SQ4	13	0.87	0.98	0.93
SQ5	3	0.92	0.98	0.96
SQ6	100	0.82	0.99	0.93
SQ7	1	0.98	0.98	0.98
SQ8	28	0.46	0.99	0.91
PN1	100	0.00	0.99	0.90

Table 4-2: Normalization with Template Specification Results

The results showed a proper correlation between the scores and the normalized chat when verified manually.

Of interest are the results for the lowest 10 conversations in the PAN-2012 sample dataset of 100 chats:

0.00	0.26	0.51	0.54	0.63	0.68	0.72	0.76	0.77	0.81

The score of 0.00 was given to the following chat:

```
[2022-07-17 05:31] "0a39f78bcb297ab0ebe8a29c28bfed89": bugmail: [Bug 10605] Typo: Replace
'the alt attribute's value may be omitted' with '@alt may be omitted'
<http://lists.w3.org/Archives/Public/public-html-bugzilla/2011Jan/0677.html> ** [Bug 10618]
Use "unmapped" rather than "no role" in the weak/strong ARIA tables
<http://lists.w3.org/Archives/Public/public-html-bugzilla/2011Jan/0676.html> ** [Bug 10066]
replace section 3.2.6 with the alternative spec text provided (ARIA) <http://lists.w3.o
```

This conversation was composed of 1 very long line. It normalized properly, however the scoring was heavily penalized by the fact that there was only 1 line in the chat and that it was very long.

The chat with a score of 0.26, is as follows:

```
[2022-07-17 15:21] "0a39f78bcb297ab0ebe8a29c28bfed89": bugmail: "[Bug 12171] Define "resolve
an address" here until there's a useful spec to reference. The lack of specification
seriously hurts any attempt to improve interoperability for anything related to URLs" (2
messages in thread) <http://lists.w3.org/Archives/Public/public-html-
bugzilla/2011Feb/0960.html>
```

This chat is very similar to the previous one, and was penalized for the same reasons. It achieved a higher score because the line length was shorter.

The chats with a score of 0.51 and 0.54 were penalized in the same way.

4.8.2 Normalization with Template Selection

In this experiment we defined all the templates as Resolute directives and allowed Resolute to select the best template for a given chat. Resolute does this by applying each template directive

to a chat and calculates the normalization score of that template. The template with the best normalization score is then selected as the template for that chat.

Class	# Documents	# Matching Template Found
SQ1	1	1
SQ2	4	4
SQ3	3	3
SQ4	13	13
SQ5	3	3
SQ6	100	100
SQ7	1	1
SQ8	28	28
PN1	100	99

Table 4-3: Correctly Selected Templates

With the exception of one chat in class PN1, all the correct templates were identified for every chat. The document that failed in class PN1 was the document with a score of 0.00 discussed in the previous section.

4.8.3 Normalization with Missing Template

In this experiment we removed a template from the set of 9 templates setup in the previous section, and normalized the chats with the missing template. The idea is to evaluate how well the scoring works in the absence of having a proper template available. We did this in turn for all 9 classes with the following results:

Class	# Documents	Wrong Template without Warning	Wrong Template with Warning	Error No Template
SQ1	1	0	0	1
SQ2	4	0	0	4
SQ3	3	0	2	1
SQ4	13	0	4	9
SQ5	3	0	2	1
SQ6	100	0	15	85
SQ7	1	0	0	1
SQ8	28	0	0	28
PN1	100	0	0	100

Table 4-4: Scoring without a proper template available

The column “wrong template without warning” indicates that Resolute selected a template for a chat even though the correct template was not specified, i.e., it selected a wrong template with high confidence. The column “wrong template with warning” indicates that Resolute selected a template but issued a warning to the user that the normalization score was low, i.e., a normalization score less than 50%. The column of ‘error no template found’ means that Resolute returned an error indicating that a template was not found, i.e., all templates returned a normalization score of 0.00.

Ideally, we would like to see all chats without a template be identified as ‘error no template’, however; some chats are small and don’t contain enough data for the normalization procedure to properly detect with enough certainty that the chat did not normalize properly.

4.9 Future Work

The work described in this chapter requires that a normalization template be manually specified prior to normalization. The scoring mechanism allows one to apply many templates to a chat and allow the system to choose the best one. This provides a productive environment for users to be able to process many chats with little effort. However, we can bring this process one step further by supporting fully automatic normalization, where a system is able to build templates automatically from a given chat.

Building a template automatically requires one to discover the format of the chat header, which in most cases is composed of a date-time and author id usually followed by a delimiter that separates the header from the chat line text. The idea would be to scan a chat and iteratively look for different date-time formats. Once a re-occurring date-time format is identified the author id usually follows it; however, discovering the author id is a rather tricky task, as it can contain more than one token and is usually followed by the chat line text. This can be mitigated, as a chat line header usually ends with some sort of delimiter. Once these entities are identified a normalization template can be built. The scoring mechanism described in this chapter can further help quantify how good the generated template applies to the chat.

4.10 Concluding Remarks

The process of transforming a corpus of text into a format that is suitable for analysis is an important and common task. Often custom programs are created to perform the task. This method of normalization is not always an option for users who do not know how to program, or that do not find that taking the time required to do so provides a suitable return on investment.

Since it is our goal to build a practical system that law enforcement can use to identify predators from chats acquired from many diverse places and in many diverse formats, we address in this chapter the issue of how to generalize normalization so that a non-technical user can efficiently process a large number of documents with many different formats. We break up the complexity of normalization by introducing templates and scoring to create a normalization methodology that sets the groundwork for a fully automated way of normalizing text.

With a suitable solution for normalization in place, we take an important step towards being able to put together a practical system that law enforcement can use to identify suspected sexual predators (see requirements: REQ3, REQ6, REQ7, NFR1).

In the next chapter we define and build an AI agent that is capable of analyzing our normalized chats to identify sexual predators.

Chapter 5: Sexual Predator Identification

In this chapter we design and build a system that is capable of performing SPI by computational means. It is composed of an AI agent that is based on a deep neural network, which utilizes NLP deep learning techniques.

The foundation of the AI agent is based on the Transformer, which is described in *section 3.4 Transformer Architecture*. In this chapter we test hypothesis 1 and show how the Transformer can be adapted to work as a SPI classifier.

5.1 Motivation

Resolute is built around the premise that an AI agent can help law enforcement in the analysis of many chats (see REQ6, REQ7, REQ8, NFR1). In that regard, the research described in section *2.2 Sexual Predator Identification* in the literature review substantiates that premise.

In addition to defining and building an application in this thesis, we are exploring in detail two other fundamental issues; normalization of raw chats, and AI based analysis of SPI. As mentioned in section 2.2 in the literature review, the PAN2012 competition provides a rich landscape for the exploration of SPI. Because of that rich landscape research papers continue to be published on the topic. Our motivation in this chapter is to add to that body of knowledge by testing hypothesis 1, while at the same time acquire a comprehensive understanding of SPI.

5.2 Contribution

The predictive performance of well-known word embeddings; Word2Vec and GloVe with regard SPI classification.

The performance (prediction and speed) of a deep learning model based on the Transformer architecture for the purposes of SPI classification.

5.3 AI Agent

The AI agent performs SPI in two distinct phases (Figure 5-1). Phase 1 is responsible for identifying suspected conversations (see REQ6), and phase 2 is responsible for identifying predators (see REQ7).

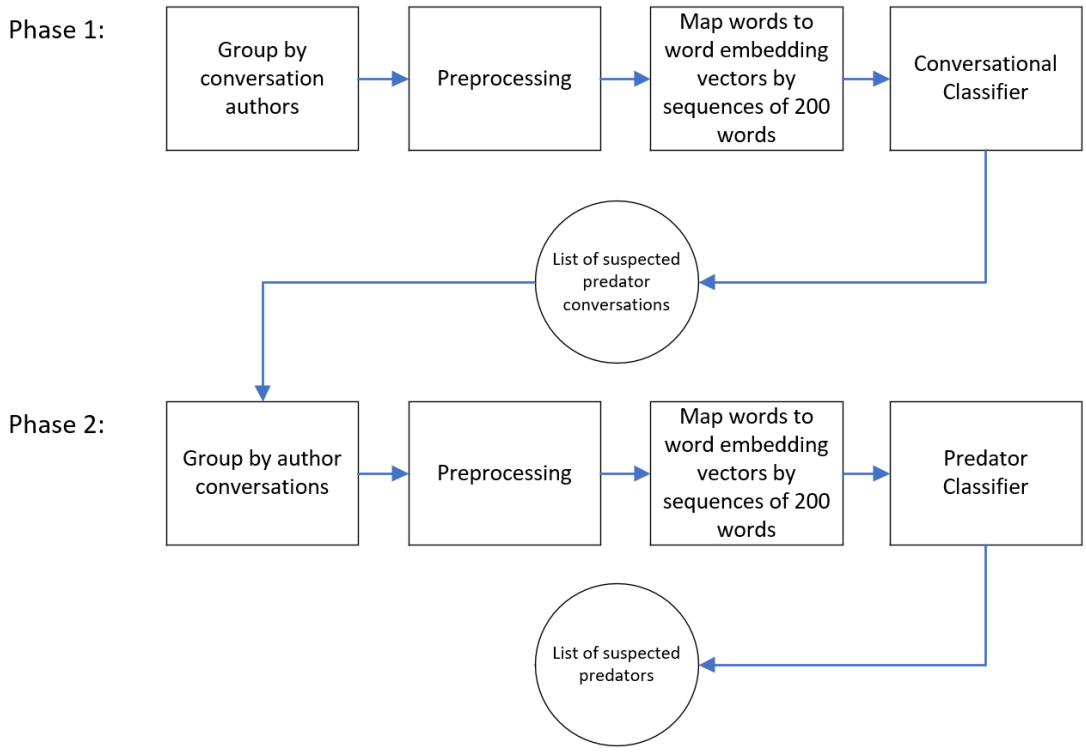

Figure 5-1: SPI System Architecture

The grouping operations are responsible for grouping the chats for a given corpus in a manner that provides the classifiers with the data needed for the type of classification required. Without grouping (i.e., feeding the chats one-by-one to the classifiers) information about a conversation between people is scattered in other chats throughout the corpus. For example, given the complete conversation of a predator and victim being divided into several different chats, one chat can hold information about the victims' age, while another holds information about sexual intent. Grouping increases the ability of the classifiers to detect SPI more accurately. Section *5.4 Chat Grouping* details how the chats are grouped for each classifier.

Preprocessing consists of cleaning up the chat text, for example the removal of non-alphabetic symbols. This serves to reduce the dimensionality of the classifiers' input vectors, and standardizes words with symbols in them. Section *5.5 Preprocessing* details the preprocessing we apply to the text.

Neural networks accept numbers as input, so the preprocessed text needs to be numerically encoded. There are many ways to do this, and which method is used weighs heavily on the performance of the model. Section *5.6 Encoding* details how the preprocessed text was encoded as input to the classifiers.

The classifiers perform the task of calculating the probability that a given chat contains a suspicious conversation, and who is the predator in those chats. The classifiers are based on the Transformer model architecture (see *3.4 Transformer Architecture*). Transformers are great at

machine translation and text generation tasks. Section *5.7 Classification* shows how we adapted the Transformer architecture for classification.

The output of the 1st phase which assigns a predatory probability to each chat, is then used to build the input to the 2nd phase which in turn assigns a predator probability to each author. The end result of the SPI AI agent is;

- A list of chats with their corresponding probabilities of a sexual predatory conversation occurring between an adult and a minor (REQ6).
- A list of authors with a corresponding probability of being a sexual predator (REQ7).

5.4 Chat Grouping

Grouping chats consists of consolidating chats from a given dataset in a certain way in order to increase the effectiveness of the classifier. In this section we will describe how chats are grouped and how grouping provides benefits over working with individual chats.

For a given dataset of chats, the AI agent uses two distinct types of groupings: all chats by the same set of authors (see Figure 5-2) and all the text written by an author (see Figure 5-3). The grouping maintains the chronological order of the chat lines.

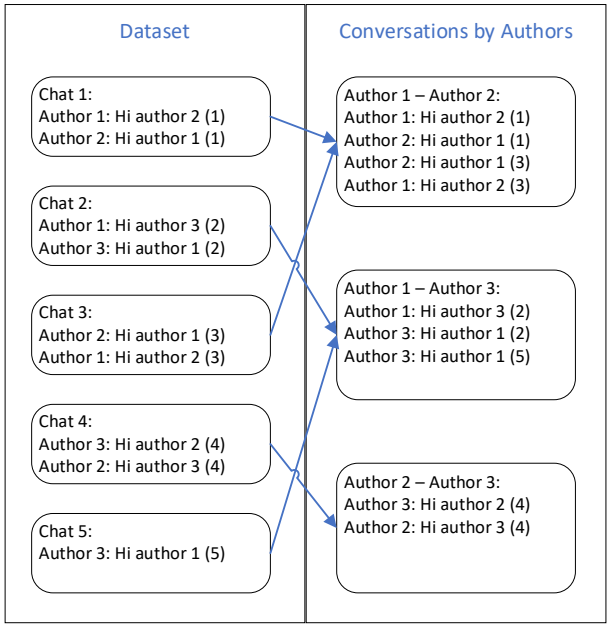

Figure 5-2: Chat grouping by the same set of authors

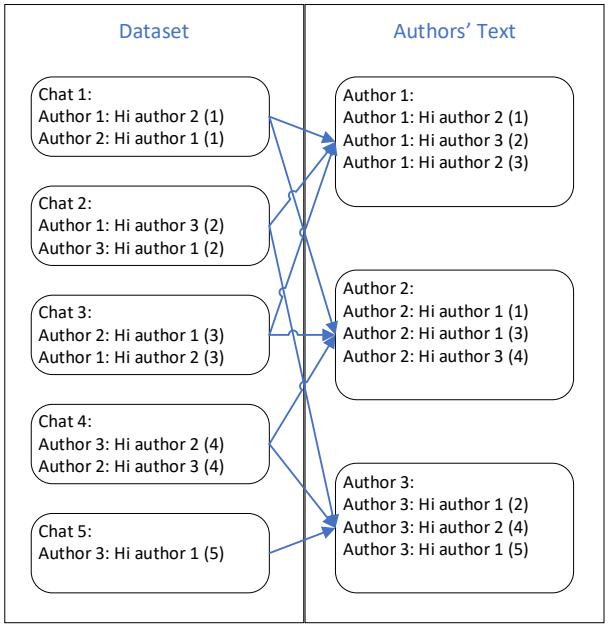

Figure 5-3: Chat grouping by all authors' text

An issue arises with the grouping of 'conversations by authors' for chats with one author, in which that author has participated in conversations with other authors. A chat with one author is not a conversation, so where does it belong? We place the chat in the group in which that author has participated and that matches the closest with regards to time.

Small chats when taken as a unit, usually do not contain enough information in them to be classified properly and usually add little value to classification. This is evident with the approach to SPI taken by Villatoro-Tello [7] who came in 1st at the PAN-2012 competition. Their filtering strategy excluded small chats entirely.

Our approach is to not filter out any chats. Small chats when grouped may provide important information to the classifier.

Grouping also provides benefits in situations where a particular conversation between a predator and victim for example span several chats. One chat may contain information that one of the authors is a minor, while another chat may reveal that there is sexual intent involved. Grouping enables the classifier to see this data consolidated as one conversation.

5.5 Preprocessing

Preprocessing is where the chat text is transformed in some way to make it conducive to being used as input to the classifier. There are essentially 3 tasks that are addressed in preprocessing:

- Standardize the text so that it is easier to process.
- Split the text into tokens.
- Convert the tokens into numerical vectors.
  This involves encoding tokens which is covered in the next section *5.6 Encoding*.

Standardizing the text is part of feature engineering (see *3.2 Feature Engineering*). The goal is to transform the text in a way that makes our Transformer model generalize better, and require less training data. The type of encoding used also plays a role in the type of transformation decisions made. The following is a list of the transformations we did for each chat:

- Remove all characters except: alphanumeric characters, and the single quote.
  The idea is to clean up the text of unnecessary noise. Note that the single quote is kept only if it is in-between two characters.
- Remove emoticons (this is by extension of the previous point).
  Emoticons could provide important information, however; based on an analysis of the PAN-212 dataset it is unlikely for a conversation involving a predator to have many emoticons. Also, the expected syntax of emoticons is not always respected, for example, :-) is also written as :-))), and supporting them would require complex processing.
- Convert all characters to lower case.
- Numerical values are transformed into either ~L if the number is less than 18 and ~H if greater. The threshold of 18 is the defining age in Quebec for a person to be considered a minor.
- Emails, and URL addresses are transformed into ~EMAIL and ~URL respectfully.
  This type of data was deemed important enough to identify and tag as it is often used by predators to exchange pictures for example.
- Postal codes are transformed into ~PCODE.
  Sometimes used by predators to meet with victims.

- Dates and times are transformed into ~DTTM.
  Dates and times are often used in coordinating a meet up.
- Repetitive characters in words are shorted to 2 characters, for example: sooorrrryyy = soorryy.
  The idea is to reduce the dimensionality of the input features, and increase the likelihood that a token maps to an embedding vector.
- Words greater than 30 characters are transformed into ~LW (i.e., Long Word).

These transformations aim to increase the likelihood that a given token will map to a word vector defined in the vocabulary table (see *5.6 Encoding*). The transformed tokens (e.g., ~EMAIL, ~URL, …) are included in the pre-trained vocabulary table. This was done by manually editing the dictionary of pre-trained tokens and replacing the definition of an existing token with the transformed token. For example, the token '0' was edited to ~L, and the token '18' was edited to ~H.

The other responsibility of preprocessing is to split the text into tokens. Each token maps to a corresponding encoding vector as discussed in the next section. There are several ways to tokenize the text, the following lists some of the more common ones:

- Character level tokenization.
- Word level tokenization.
  This is the tokenization used by the models described in this thesis.
- N-gram tokenization.
  Tokens are groups of N consecutive words. For example, "Hi there" would be a 2-gram token.

The final step in preprocessing is to convert the tokens into numerical vectors, i.e., map the tokens to an encoding. The format and values for numerical vectors or encodings depends on the model. This topic is covered in detail in next section *5.6 Encoding*.

5.6 Encoding

Artificial neutral networks of the type that we use for classification cannot take raw text as input. They can only process numerical vectors (or tensors). After grouping and preprocessing the chats, the next step is to encode the tokens into a numerical tensor to be used as input to the model.

There are essentially 2 types of text processing models; sequence models, and bag-of-words models. Sequence models care about the order of the words in a text. Word level tokens are often used for sequence models. Bag-of-words models treat input as a set of words. N-gram tokens are often better suited for bag-of-word models.

Transformers are sequence-to-sequence models, and our encoding uses word level tokens. There are numerous ways to encode tokens. For example, the tokens can be hashed to a unique value. More commonly, and the method used in our model, is to start by representing tokens by an

index to a vocabulary table. The vocabulary table can be built from the training data, or from an external corpus (e.g., pre-trained word embeddings) (see *3.3 Word Embeddings*). The following diagram depicts the relationship between the tokens and the vocabulary table:

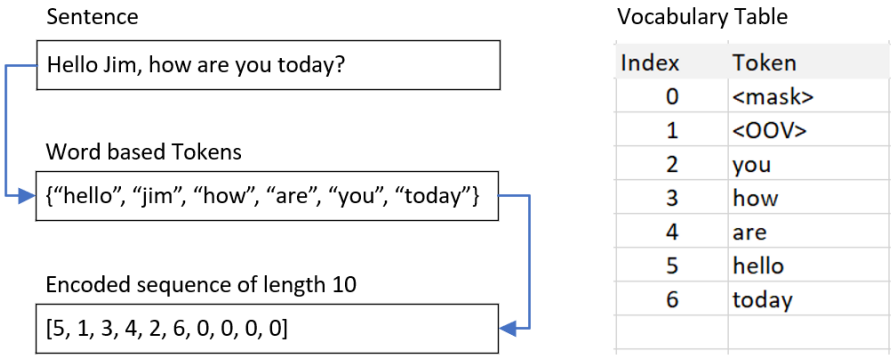

Figure 5-4: Token to vocabulary index encoding

The <OOV> token in the vocabulary table signifies an 'Out Of Vocabulary' token, and is used to mark tokens that are not in our vocabulary. The <mask> token is used to indicate that there is no token. The model learns to ignore these tokens.

A list of encoded tokens is known as a sequence, and our model expects the input encodings to be in contiguous batches of fixed length sequences. This poses a problem since chats are of variable length. Sequences that contain chats smaller than a given sequence length are padded with the <mask> token. Chats that are larger than a sequence are broken into 2 or more sequences.

There is a potential problem for chats that span more than 1 sequence, since our model essentially works with one sequence of data at a time. A large chat that is broken into multiple sequences can potentially hide information about a predator or victim from the model. For example, suppose a chat is broken into 2 sequences. The first sequence may contain information about the victims age, and the second sequence about the predators' intent. We can mitigate this limitation by overlapping (window) the sequences for a given large chat with the previous sequence for that chat.

There is another issue with the encoded sequences we need to address. The sequences contain indexes to the vocabulary table that are represented by large integers. Neural networks do not work well with large numbers. One possible solution is to one-hot encode the values into a sequence of one-hot encoded vectors. There are several problems with this approach as well. Since we can expect our vocabulary table to be very big (greater than 100,000 tokens) the dimensionally of our model (which matches the number of tokens in our vocabulary for every token in our sequence) would be exceedingly high. An alternative to one-hot encoding is to use word embeddings.

Word embeddings are dense low-dimensional (e.g,. 25-1024 dimensions) vector representations of words. Refer to section *3.3 Word Embeddings* for details. The vector representations contain contextual information about each token. This provides a rich set of low dimensional features as

input to the classifier. With the exception of some experiments done by *Pastor Lopez-Monroy et al.* [3] with Word2Vec[11] and *P. Borj, et al.* [8] with GloVe, not much is known about the use of Word2Vec and GloVe embeddings with respect to using them for SPI. In this thesis we explore embeddings learnt from the training data and pre-trained Word2Vec and GloVe embeddings.

5.7 Classification

There are two models defined in a SPI AI agent as described in section *5.3 AI Agent*. One model is responsible for identifying conversations that contain sexual predation of minors. The other model is responsible for detecting which author in the set of detected chats is the predator. This process of partitioning the problem in two has been explored before from SPI attempts of the PAN-2012 dataset with good results (see *2.2.2 Two-Phased Approach*). Also, the independent scoring of conversations and authors allows us to support the following requirements in Resolute:

- REQ6: Provide the likelihood that a chat contains predatory behavior
- REQ7: Provide the likelihood that an author is a predator

The models are built using Keras[12] and Tensorflow[13]. Keras is a deep learning API for Python, built on top of TensorFlow.

For our experiments we built and evaluate two AI agents;

- Based on a simple neural network for classification (NN-Agent)(see *Figure 5-5*)
- Based on a Transformer encoder followed by a simple neural network for classification (TR-Agent)(see *Figure 5-6*)

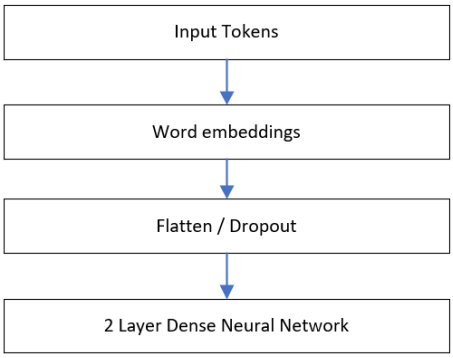

Figure 5-5: NN-Agent, conversation and author model architecture

[11] The pre-trained Word2Vec vectors used in their experiments were manipulated by MUL
[12] https://keras.io
[13] https://tensorflow.org

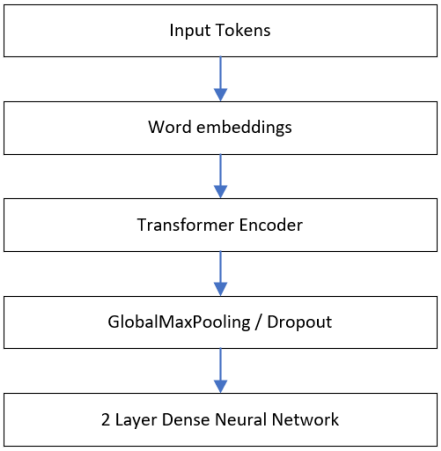

Figure 5-6: TR-Agent, conversation and author model architecture

The TR-Agent uses a Transformer encoder to enhance the token representations which are fed into a neural network for classification.

The NN-Agent will be used as a baseline for comparison with the TR-Agent. This will allow us to get a sense of the performance of the Transformer encoder.

5.8 Experiments and Results

The performance of our model is calculated using precision (P), recall (R) and weighted harmonic mean (F), which are based on the same metrics defined for the PAN-2012 competition [14]:

$\text{Precision (P)} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$
$\text{Recall (R)} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})}$
$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad (0 \leq \beta \leq \infty)$

Table 5-1: PAN-2012 competition model performance metrics

At the PAN-2012 international competition, both β=1 and β=0.5 were used to score the results of the competitors. With β=1 both precision and recall are treated equally, and with β=0.5 precision is given a higher weight. The organizers of the competition deemed $F_{0.5}$ as more relevant to the task at hand. According to the organizers "*what is more important is the fact that the retrieved authors are relevant (Precision). This to optimize the time of the police agent towards the "right" suspect rather than "all" the possible suspects. For this reason we used a measure of F with the β factor equal to 0.5, in order to emphasize Precision.*" [14].

Our experience with the SQ is different, they had indicated that they would rather have a higher set of false-positives than false-negatives. This translates to a greater importance on recall vs.

precision. The SQ envisioned a system when given a large number of documents, that an AI based agent could filter out the irrelevant chats down to a manageable set, without sacrificing documents that can be used as evidence. In this section we report on β=1 and β=0.5 to keep consistent with the results of the PAN2012 competition. This allows for a comparison of performance between the different approaches to SPI.

The following tables summarizes the hyperparameters that will be evaluated:

Sequence grouping (group)	Conversations can be evaluated as individual chats or they can be grouped by the same set of authors as described in section *5.4 Chat Grouping.* Note, that for author-based models grouping is implied.
Classification type (ct)	i.e., cost/loss function Binary cross entropy and categorical cross entropy are both evaluated. Note that the notation below will use 'PN' to signify categorical classification and 'P' for binary. Also, the classification type for the conversational and author models will be separated by a '/', e.g., PN/P.
Embedding types (em)	The following word embedding types will be evaluated: • None: embeddings are learned from the training data • Glove 1: 50, 100, 200, 300 dimensional vectors pre-trained on 6 billion tokens taken from Wikipedia • Glove 2: 25, 50, 100, 200 dimensional vectors pre-trained on 27 billion tokens taken from Twitter • Word2Vec: 100, 300, 500 dimensional vectors pre-trained on tokens taken from Wikipedia
Embedding dimension (emdim)	The dimension of the embedding vectors
Sequence length (seqlen)	The models are setup to evaluate fix length sequences of tokens as described in section *5.6 Encoding*.
Sequence evaluation (seqeval)	The text for a given conversation or author can span more than one sequence as described in section *5.6 Encoding*. Two methods are used to calculate the probabilities for a given conversation or author: • max(): the probability is assigned based on the sequence with the highest value. • weighted_average(): the probability is calculated from the weighted average of all the sequences. The weight comes from the number of tokens in a sequence.
Epochs	The number of epochs used for training
Final_nn_nodes (fnodes)	The number of nodes used in the final 2-layer dense neural network
Heads (heads)	The number of attention heads configured in the Transformer model.

Tr_dense_nodes (trnodes)	The number of nodes used in the Transformer intermediate dense neural networks.

Table 5-2: Summary of model hyperparameters

The following sections detail different hyperparameter configurations, with the goal of building an AI agent that achieves the best β=0.5 f-score for author detection. All experiments are based on 75% of the training data used for training and 25% used for validation. Each experiment reports on the average value of at least three runs.

5.8.1 Base Models

The first set of configuration parameters are given values based on processing simplicity and a best guess based on some initial experimentation. The following results are based on training with the full PAN2012 dataset:

Model	Configuration	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	group=false, ct=PN/PN, em=none, emdim=100, seqlen=200, seqeval=wavg, fnodes=32, epochs=15	0.923	0.693	0.791	0.865
Transformer	group=false, ct=PN/PN, em=none, emdim=100, seqlen=200, seqeval=wavg, fnodes=32, epochs=15, heads=2, trnodes=128	0.839	0.654	0.732	0.792

Table 5-3: Base model results

To address the question of what impact filtering has on the performance of the SPI models, the following table shows the results of training the same models with the VTPAN dataset:

Model	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	0.98	0.851	0.908	0.949
Transformer	0.978	0.826	0.894	0.942

Table 5-4: Base model with VTPAN training dataset results

Given an unoptimized model, the experiment shows that filtering has a significant impact on the results. Should filtering be included with the final version of Resolute? That question needs to be considered carefully, because, filtering by its very nature discards data. In a competition such as the PAN2012 competition where achieving the best score is the objective, filtering makes perfect sense. In a production environment where the well-being of people is involved, filtering needs to be assessed more carefully. As a result, our experiments and model optimizations will use the full PAN2012 dataset, but an additional test at the end will be included with VTPAN for comparison.

In the next section the conversation grouping and classification type will be examined.

5.8.2 Conversation Grouping and Classifier Type

The input for the classification of conversations can incorporate either no grouping of chats or grouping as described in section *5.4 Chat Grouping.* Since the type of classification (P=binary cross entropy/PN=categorical cross entropy) can influence the final results based on the type of grouping, the classification type is also considered.

The following graph shows experiments altering the chat grouping for the conversation classifier and the classification type for both the conversation and author classifiers. The x-axis labels indicate 'conversation-class-type, author-class-type, grouping', where P=binary classification and PN=categorical. Other parameters are fixed at: em=none, emdim=100, seqlen=200, seqeval=wavg, fnodes=32, epochs=15, heads=2, trnodes=128:

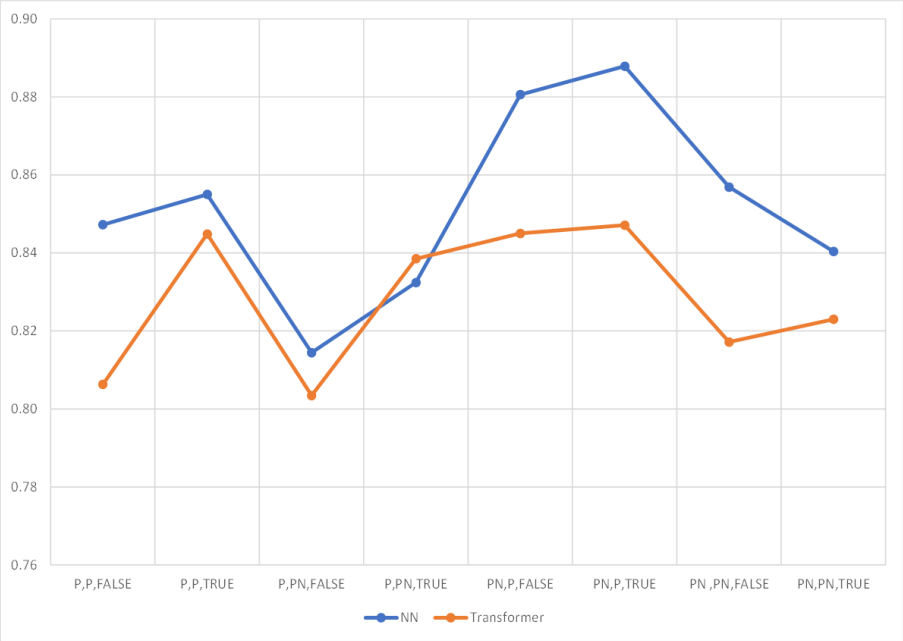

Figure 5-7: Conversation grouping and classifier type results

Best results for each model are when the conversations are grouped and classified categorically, and the author classifier is configured for binary classification (i.e., PN, P, TRUE):

Model	Configuration		Precision	Recall	$F_{1.0}$	$F_{0.5}$
	grouping	Binary/Categorical				
NN	True	Conversation: PN (i.e., categorical) Author: P (i.e., binary)	0.931	0.749	0.830	0.888
Transformer	True	Conversation: PN (i.e., categorical) Author: P (i.e., binary)	0.906	0.666	0.767	0.844

Table 5-5: Conversation grouping and classifier type experiments best results

Grouping the conversations improves the results for all combinations of classifier types, except for a slight decrease on the NN agent for the PN/PN classifier types. Better results with grouping are expected as the conversation classifier has more data to work with.

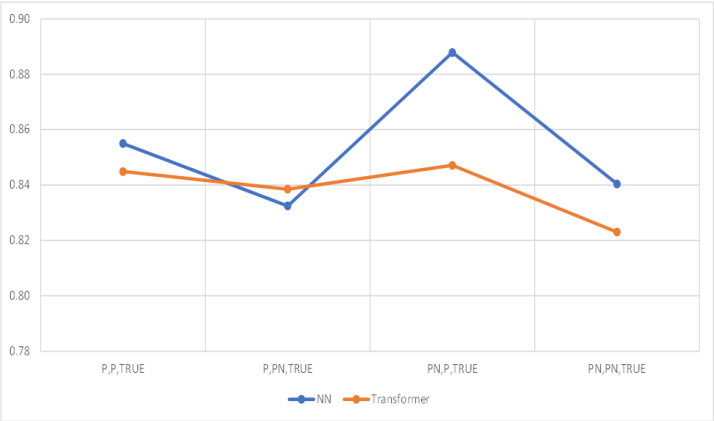

Figure 5-8: Classifier type with conversation grouping

Figure 5-8 shows the classification types for the grouped conversations. The author classifier performs slightly better as a binary classifier (*,P). This could be because of the difficulty in detecting the difference in symmetry between a predator and victim as discussed in the literature review (*see 2.2.5 Lexical, Behavioral & Social Features*). With this setup a suitable threshold can be used to decide between a predator and victim. The conversation model performs best as a categorical classifier (PN). This is expected if we consider the observed symmetry between victim and predator, i.e., the classifier performs well deciding between a predator/victim, vs. a normal conversation.

In the next section the word embeddings and dimensionality of the word vectors will be evaluated.

5.8.3 Word Embedding

The proper type of word-embedding is crucial to the performance of any NLP based model. This has been demonstrated from the work reported by *M. Vogt, et al.* [2] (BERT/NN) and *Pastor Lopez-Monroy et al.* [3] (MulR/SVM). Their state-of-the-art results were achieved mainly by the type of word embeddings that were used, i.e., BERT and MUL(TVT[14]). Our contribution will explore other types of word embeddings. The following embedding types will be evaluated:

- None: embeddings are learned from the training data using the following word vector dimensions: 25, 50, 100, 200, 500
- Glove 1: 50, 100, 200, 300, dimensional pre-trained GloVe vectors based on 6 billion tokens taken from Wikipedia
- Glove 2: 25, 50, 100, 200, dimensional pre-trained GloVe vectors based on 27 billion tokens taken from Twitter
- Word-2-Vec: 100, 300, 500, dimensional pre-trained Word2Vec vectors based on tokens taken from Wikipedia

[14] Temporal Variation Terms (TVT)

Note that higher dimensional vectors provide more input into the final linear network layer. In an effort to keep the playing field fair, the same experiment is performed varying the number of nodes in the final linear network with the following values: [32, 514, 1024, 2048, 4096]. All experiments are performed at least three times and results are averaged.

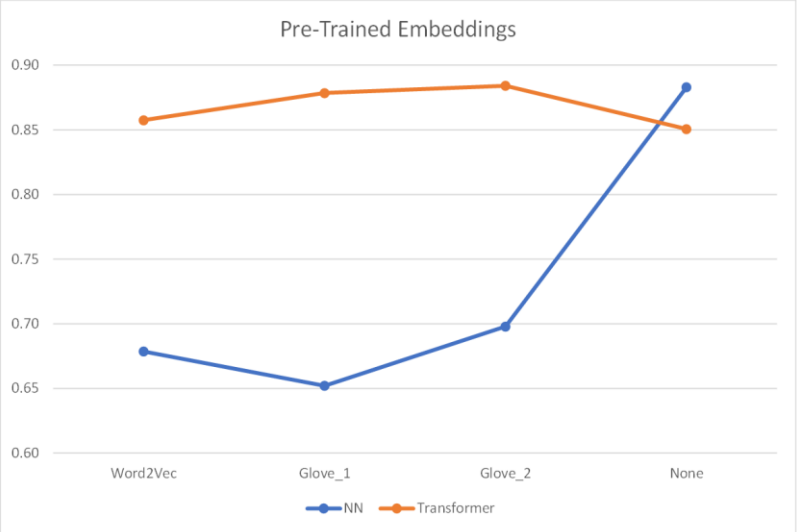

Figure 5-9: pre-trained embedding results

The best embedding for the NN model is 'None', which are embeddings learnt from the training data. This result is surprising as it is reasonable to expect that at least one of the other pre-trained embeddings would outperform embeddings learnt from the training data. *Pastor Lopez-Monroy et al.* [3] experienced the same situation with their Word2Vec embeddings experiments. The inferior results of the pre-trained embeddings might be due to the limited depth of the model, i.e., the pre-trained vectors remain static during training, only the last neural network is actually trained. Whereas the learnt embeddings offer the NN model a deeper architecture in which to learn, i.e., the learnt embedding vectors are updated during training and provide the model with a deeper set of layers to work with.

The best results for the Transformer based model is Glove_2, i.e., pre-trained GloVe embeddings based on 27 billion tokens taken from Twitter. The Transformer model produced significantly better results for all the pre-trained embeddings as compared to the NN model. In contrast to the NN model the Transformer model has a deep neural architecture, which supports our proposition above for why trained embeddings in the NN model significantly outperform the pre-trained embeddings.

For each respective 'best embedding', the following diagram shows results for different embedding vector dimensions:

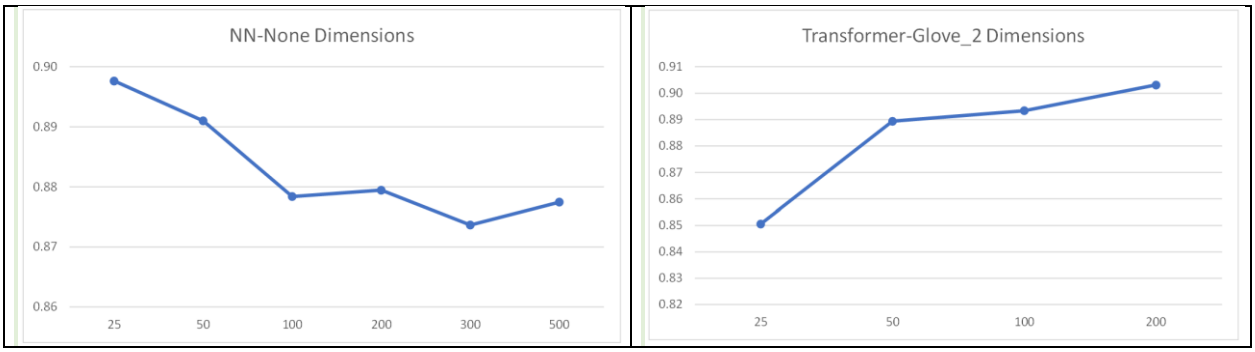

Figure 5-10:best embedding dimension results

The best embedding dimension for the NN model is 25. Graph '*NN-None Dimensions*' shows that lower dimensions produce better results. This can be explained by the fact that the model is building the embeddings from the limited training data, and that there is not enough data present to properly build the complex word representations required for higher dimensional embeddings.

The best embedding dimension for the Transformer model is 200. This result is in line with our expectation as the GloVe_2 embeddings were trained on a huge corpus, and the higher dimensions represent a more robust token representation.

Best results for the best embedding configuration:

Model	Configuration	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	group=false, ct= PN/P, em=none, emdim=25, seqlen=200, seqeval=wavg, fnodes=<var>, epochs=15	0.936	0.773	0.846	0.898
Transformer	group=true, ct= PN/P, em=glove_2, emdim=200, seqlen=200, seqeval=wavg, fnodes=<var>, epochs=15, heads=2, trnodes=128	0.962	0.729	0.828	0.903

Table 5-6: Word embedding type experiment best results

In the next section the input sequence length and the number of nodes in the final two-layer neural network will be evaluated.

5.8.4 Two Layer Dense Neural Network and Sequence Length

A sequence is a fixed length buffer containing text that is used as input to the model. The sequence length refers to the number of tokens (words) used as input to the model. Text that contains more than the maximum sequence length is overlapped (i.e., windowed) over onto the next sequence as described in section *5.6 Encoding.* Text that is shorter than the maximum length is padded with a special mask token.

During evaluation the model calculates a probability for a given sequence. If the text spans more than one sequence a weighted average (*wavg*) for the sequences is computed to give a probability for the complete text. The weight for a sequence is taken from the number of tokens in a given sequence. Another common method is to assign a probability for a given text by taking the

probability of the sequence with the highest probability (*max*). In the experiments below we compare both methods.

The other parameter we will be evaluating is the number of nodes in the final two-layer dense neural network. The following graphs summarize the results for the NN and Transformer models, varying the sequence length and number of nodes in the final layer:

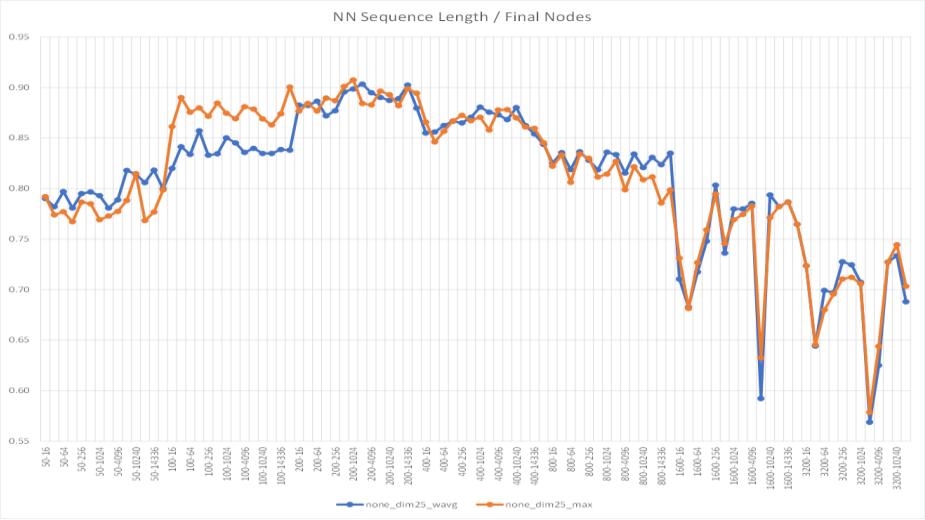

Figure 5-11: NN-agent results of sequence lengths of 50 to 3200

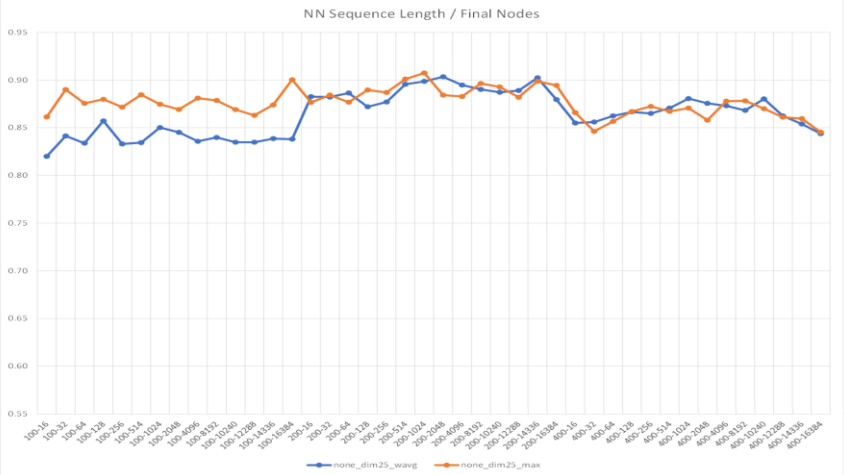

Figure 5-12: NN-agent results of sequence lengths of 100 to 400

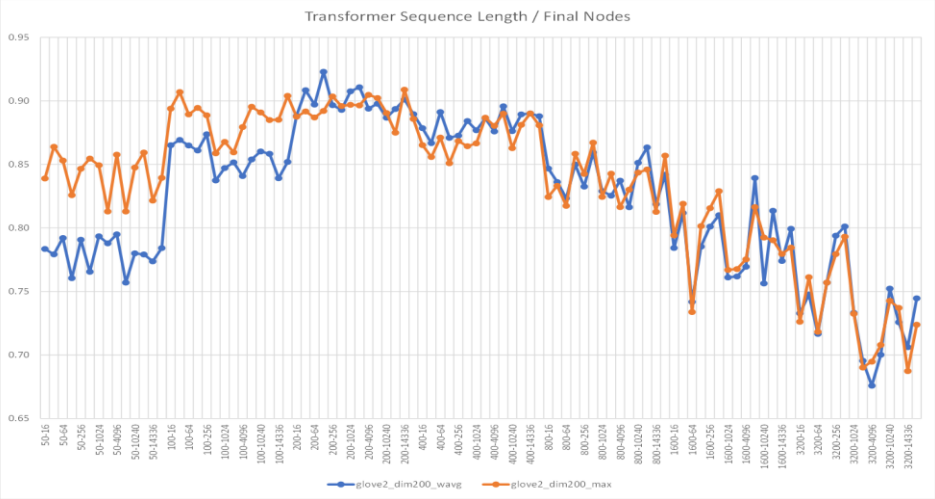

Figure 5-13: TR-agent results of sequence lengths of 50 to 3200

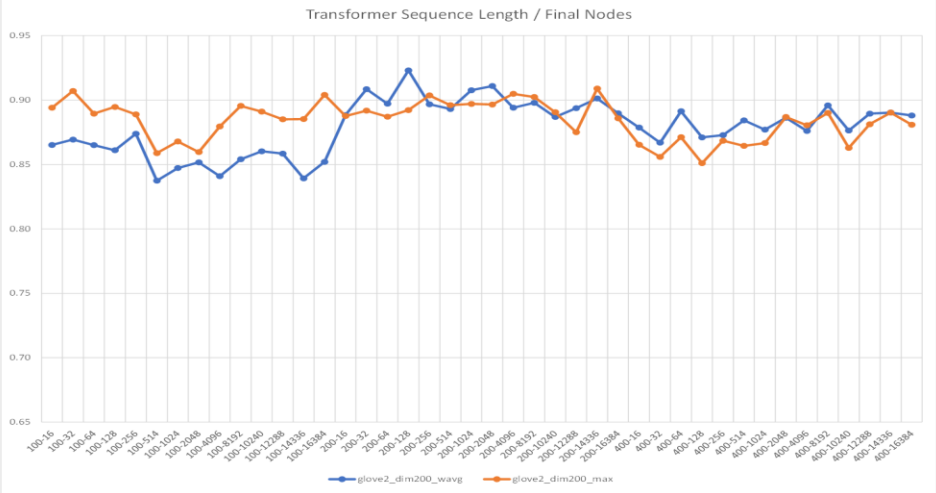

Figure 5-14: TR-agent results of sequence lengths of 100 to 400

The NN model performs best with a sequence length of 200 and with 1024 nodes in the final two-layer dense network. The Transformer model performs best with a sequence length of 200 and with 128 nodes in the final two-layer dense network.

Both agents show that sequence lengths between 200 and 400 provide the most stable results. Smaller and larger sequences have a negative effect on the results. The data indicates that sequences smaller than 100 do not encapsulate enough information to make a definitive prediction and sequences greater than 400 contain too much information such that the important markers in the data are lost. Perhaps more training epochs might help stabilize sequences greater than 400, however the variability of the results in concert with their poor performance does not indicate that we could achieve results better than those achieved with a sequence length of 200.

More nodes in the final 2-layer dense neural network do not have a significant impact on the results. This could be an indication of the representational effectiveness of the embeddings used

as input to the dense neural network, e.g., even a dense neural network of 16 nodes offers results close to those with 16,384 nodes.

Regarding the methods of calculating the overall probability of a text, maximum (*max*) does better for sequence lengths less than 200. This is expected as probabilities calculated using weighted averages (*wavg*) for small sequences would contain more sequences to average across giving less weight to any individual sequence that contains predatory text. Interestingly, the weighted average (*wavg*) and maximum (*max*) methods of calculating the overall probability both produce similar results for sequence lengths greater than 100. Using the visual data in the graphs above, the NN seems to perform slightly better with *max* and the Transformer with *wavg*.

Model	Configuration	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	group=false, ct= PN/P, em=none, emdim=25, seqlen=200, seqeval=max, fnodes=1024, epochs=20	0.962	0.741	0.837	0.907
Transformer	group=true, ct= PN/P, em=glove_2, emdim=200, seqlen=200, seqeval=wavg, fnodes=256, epochs=20, heads=2, trnodes=128	0.978	0.711	0.823	0.909

Table 5-7: Two-layer dense neural network and sequence length experiments best results

In the next section some specific Transformer parameters will be evaluated.

5.8.5 Transformer

The Transformer architecture defines multi-head attention layer(s) and a series of hidden dense neutral networks (see section *3.4 Transformer Architecture*). In this section we will be running experiments altering the number of heads and nodes in the hidden dense neural network layers.

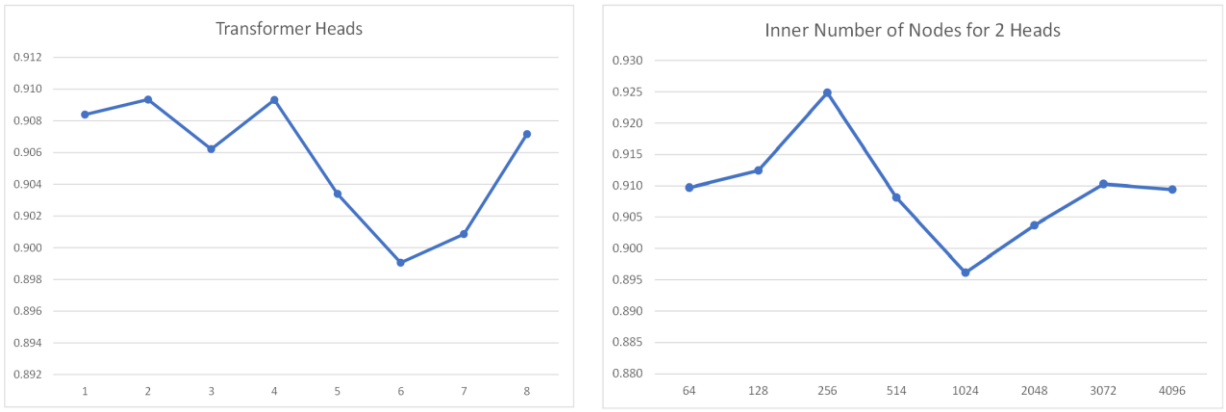

Figure 5-15: Number of Transformer heads and inner-network nodes

Two and four heads produced the best results. Two heads will be used going forward as it requires less resources. With 2 heads the internal dense network works best with 256 nodes.

Model	Configuration	Precision	Recall	$F_{1.0}$	$F_{0.5}$
Transformer	group=true, ct= PN/P, em=glove_2, emdim=200, seqlen=200, seqeval=wavg, fnodes=128, epochs=20, heads=2, trnodes=256	1.0	0.812	0.915	0.925

Table 5-8: Transformer heads and inner-network nodes best results

Each head helps the model learn different groups of features for each token. In our setup the data shows that increasing the number of heads does not impact the results in any significant manner. This could be due because the input embeddings we are using (GloVe) already encode a rich set of features in their own right. If this is true then is using a Transformer encoder for classification in our model redundant? The results being achieved by the NN model seem to suggest this.

In the next section we will run a series of experiments looking for the optimal number of epochs to use for training.

5.8.6 Epochs

In this section we run experiments altering the number of epochs each model is trained with. The following graph summarizes the results:

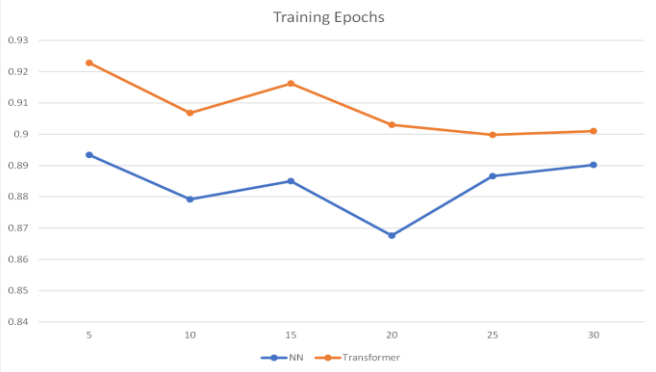

Figure 5-16: Number of epochs experiments

Best results at 5 epochs for both the NN and Transformer models:

Model	Configuration	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	group=false, ct= PN/P, em=none, emdim=25, seqlen=200, seqeval=max, fnodes=1024, epochs=5	0.934	0.761	0.838	0.893
Transformer	group=true, ct= PN/P, em=glove_2, emdim=200, seqlen=200, seqeval=wavg, fnodes=128, epochs=5, heads=2, trnodes=256	0.981	0.746	0.847	0.923

Table 5-9: Number of epochs best results

In the next section we will be evaluating the AI agents against the PAN2012 and VTPAN test datasets.

5.8.7 Test Results

The optimal configuration based on experiments:

Model	Parameter	Configuration
NN	Conversation grouping	False
	Conversational classification (loss function)	CategoricalCrossentropy Predator/Normal
	Author classification (loss function)	binary_crossentropy Predator
	Embedding	Learnt from training data
	Embedding dimension	25
	Sequence length	200
	Sequence evaluation	Maximum, i.e., highest sequence probability
	Number of nodes + activation function of final dense linear network	Conversation = 1024 (Softmax) Author= 1024 (Sigmoid)
Transformer	Conversation grouping	True
	Conversational classification (loss function)	CategoricalCrossentropy Predator/Normal
	Author classification (loss function)	binary_crossentropy Predator
	Embedding	GloVe pre-trained on 27 billion tokens taken from Twitter
	Embedding dimension	200
	Sequence length	200
	Sequence evaluation	Weighted average
	Number of nodes + activation function of final dense linear network	Conversation = 128 (Softmax) Author= 128 (Sigmoid)
	Number of Transformer heads	2
	Number of nodes in Transformer internal dense linear networks	256

Table 5-10: Test configuration based on experiments

The models are built using Keras and Tensorflow. *Table 5-11* shows the test results taking the average of 10 runs, i.e., training with the PAN2012 training data, with evaluation on the test dataset:

Model	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	0.959	0.768	0.852	0.913
Transformer	0.965	0.761	0.85	0.915

Table 5-11: SPI test results

The average time for each evaluation of the test dataset is approx. 5 minutes for the NN-agent, and 6 minutes for TR-agent on a modestly configured workstation.

The NN and Transformer models performed at the same level both in predictions and speed of training and evaluation. It is interesting that the Transformer configured as a classifier performed

the same as the NN, especially given that the NN did not have token positional information to work with, as the Transformer did. Perhaps the results reflect the quality of the data and a convergence toward the upper limit of what is detectable without filtering out the noisy data. *J. Parapar et al.* [11] the highest performing approach that does not filter the PAN2012 dataset reported a $F_{0.5}$ result of 0.87.

In order to get an idea of how filtering might affect the performance of our models, the following table shows the test results on the VTPAN dataset taking the average of 10 runs, i.e., training with the VTPAN training data with, evaluation on the VTPAN test dataset (note: the models were not optimized for VTPAN):

Model	Precision	Recall	$F_{1.0}$	$F_{0.5}$
NN	0.972	0.753	0.848	0.918
Transformer	0.98	0.757	0.854	0.925

Table 5-12: Base model with VTPAN test dataset results

The Transformer model performed slightly better because of an increase in the precision, which to some extent is expected when most of the non-predator data is removed. The $F_{0.5}$ results for the NN were less than *E. Villatoro-Tello et al.* [7] (0.93) which used a similar approach to our NN model. The difference could be because our NN encodings do not include token positional information in it, whereas *E. Villatoro-Tello et al.* used n-grams and a TF-IDF weighing scheme which provides a limited form of token position.

Based on the results of the experiments, filtering helps greatly with weak models (Table 5-4) but does not add much with strong ones (Table 5-12).

M. Vogt, et al. [2] achieved good results using pre-trained BERT [28] embeddings over a linear network. The BERT architecture is based on a Transformer but it also incorporates several additional processes tailored to building highly representational embeddings, for example bidirectional encoder representations using a "masked language model" (MLM) pre-training objective (*J. Devlin at al.* [16]). For comparison, the role of the Transformer in our models perform a similar function in that the output from the Transformer encoder is fed into a 2-layer neural network for classification, but it does not include the extra processes defined in BERT. The idea is that the Transformer in our models should be able to enhance the input GloVe embeddings in a manner that would provide the final 2-layer network layer with enough representational expressiveness that the final 2-layer neural network could provide superior classification results over the NN model. This however did not prove to be the case, i.e., the results of our Transformer agent are very close to the NN agent.

The results achieved by our NN model on the full PAN2012 dataset suggest that the word embeddings contributed a great deal to the final results, and that the Transformer encoder layer in our model is not able to enrich the input embeddings in any significant manner as to produce superior results.

5.9 Future Work

The following are some areas of research that might be worth future exploration:

- Creation of an AI agent architecture that can support the analysis of many different languages. The biggest problem we had with building an AI agent for French was the training and testing dataset. This problem extends to many other languages as well. What if we could use the power of the Transformers' ability to do language translation, coupled with a pre-trained model on the English dataset to be able to perform SPI in multiple languages without having to create an extensive set of training and test data for each language.
- One of the features the SQ mentioned that would be useful to them is to have an AI agent that could identify the lines in a chat that are predatory in nature. This was also one of the two events of the PAN2012 SPI competition. Unfortunately, the PAN2012 organizers did not provide the classification labels for the task in the training dataset, but did provide the labels for the test dataset. The test dataset can be converted into a training and test dataset for this task. Considering that the first rank for this event was by 'grozea12-run-2012-06-14-1706b' with $F_1 = 16.16\%$, $F_3 = 47.62\%$ [14] [29], there is more exploration of this task required.

5.10 Concluding Remarks

We have shown that in support of hypothesis 1 (see *1.3 Research Objectives*) the Transformer architecture used as a SPI classifier can obtain good results and speed for the identification of sexual predators in chats. In addition, our contribution shows that when using trained or pre-trained word embeddings, the Transformer encoder used for SPI text classification does not perform much better than a Neural Network.

The results achieved by our NN model on the full PAN2012 dataset suggest that the word embeddings contribute a great deal to the final results, and that the Transformer encoder layer in our model is not able to enrich the input embeddings in any significant manner as to produce superior results. GloVe word embeddings provide good results but more recent work done by *M. Vogt, et al.* (BERT) [2] and *P. Lopez-Monroy et al.* (MulR) [3] show that recent advancements in word embeddings can offer significant increases in SPI detection.

Filtering data in a dataset provides a boost in performance, especially with regard to weaker models. Filtering also helps balance out the big disparity between predator and non-predator however we did not see an expected increase in recall. As a result, our final test models did not benefit greatly from filtering.

Chapter 6: Resolute

Resolute is an application we built as part of this thesis that essentially allows for the management, processing and analysis of online chat documents, with specific focus on identifying sexual predation of minors in chats.

Some of the features and benefits of Resolute include:

- Document management:
    - A secure environment to store sensitive documents.
    - The ability to store, access and manage a large number of documents efficiently.
    - Original documents (i.e., imported chats) cannot be modified. That is, any work done to a chat (e.g., normalization) never changes the original document. Instead, any changes or meta data is linked to the original. This is important for law enforcement when building a case against a suspected predator, since the original documents need to be presented unaltered to the court as evidence.
- Document processing:
    - Document conversion: provides tools to easily and quickly convert documents from their native encoding into a text-based document. (e.g., MS Word to text).
    - Document transformation: incorporates tools to transform chats from native formats into consistent, normalized form that can be used for manual and automatic analysis.
    - Anonymization: tools that obfuscate named entities.
- Document analysis:
    - Manual analysis tools: chat viewer: chats are presented to the reader in a legible manner. Authors can be color coded or excluded from a chat making it easier for the reader to isolate points of interest.
    - Chat summaries: important chat information and statistics are presented to the user to quickly identify potential chats of interest.
    - Automatic analysis tools: a computer AI agent that scans documents looking for interactions between sexual predators and minor aged victims.

Resolute offers a GUI interface that is responsible for consolidating all functionality in a cohesive, productive manner. This is where the work of managing, normalizing and analyzing chats is done.

6.1 Motivation

Resolute is an application that helps manage a large number of chats and brings together the processes and tools required in order to be able to make effective use of the data. An application such as Resolute is essential for SPI analysis if it is going to be utilized on a day-to-day basis to catch predators.

Our motivation in building Resolute is to build an application that law enforcement can use to on a day-to-day basis to identify and help prosecute suspected sexual predators.

6.2 Contribution

Resolute provides a method of handling big data, in a high-performance environment. It provides data management facilities, that help organize chats in a secure manner. Resolute brings together a set of tools that can normalize, anonymize, and analyze the chats in a cohesive and productive environment.

6.3 Specification

Interaction with the SQ was done through a series of in-person meetings, emails and video conferences. Initially discussions centered around getting an understanding of their current operations and what we could do to facilitate the process. The following activity diagram summarizes their business process:

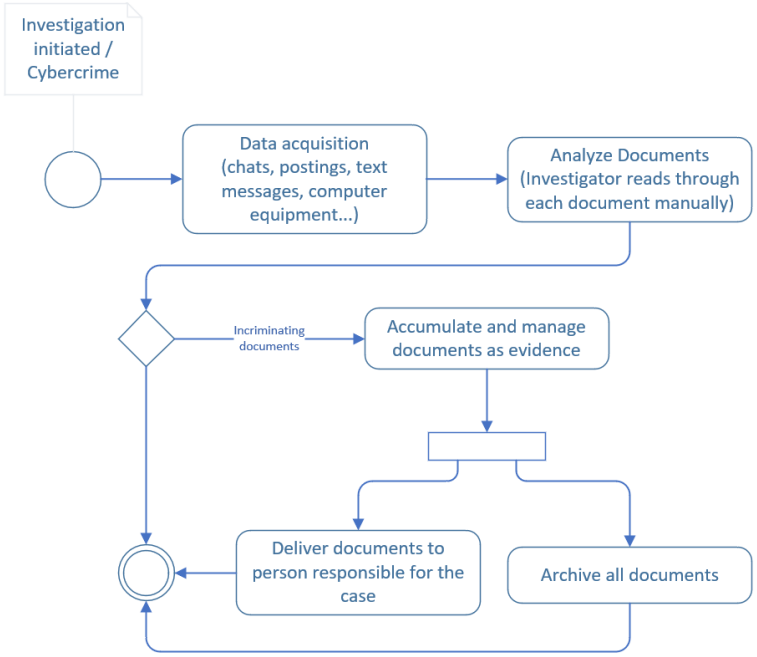

Figure 6-1: SQ cyber-crime investigation unit business process

There are three major parts to consider in the business process; data acquisition, document analysis and document management activities. In terms of data acquisition, the process used by the SQ is very dynamic, involves many different platforms and technologies, and changes so often that the activity was deemed to be better off to remain as it is for the near future. In this thesis we address the two other parts; document management and analysis.

In terms of document management Resolute offers a series of functions that enable the SQ to organize, categorize, archive, and annotate many documents by folder, dataset or dataset collection as they see fit (see REQ1-REQ4).

In terms of document analysis Resolute offers features that assist in the normalization of chats, sexual predator identification at the conversation and author level, chat anonymization and a chat viewer (see REQ3, REQ5-REQ8).

One of the requirements identified with the SQ is the ability to detect SPI in French chats (REQ8). Since a dataset similar to the PAN2012 dataset is unavailable for French chats it was decided that the SQ would build one. In an effort to help the SQ in this endeavor we include a chat anonymizer in Resolute (REQ5). Unfortunately, the SQ was not able to put together a sufficient dataset for use with our SPI agent (see *3.1.3 Sûreté du Québec Dataset*). Although the SQ can still use Resolute for English chats the missing ability to process French chats is a drawback to its useability. The intention is to continue to work with the SQ past the work done in this thesis to find a proper solution to the French chats.

6.3.1 Constraints

The following section lists the constraints on the functional and non-functional requirements of Resolute.

CON1: End user is not expected to have programming proficiency

- System should not require the user to have more than a basic knowledge of programming

CON2: Concurrent users in a dataset not supported

- System will not support two or more users working concurrently on the same dataset.

CON3: Chat only input documents

- System will support only conversational chats as input for normalization and analysis.

CON4: Analysis of French chats

- The analysis of French chats is contingent on acquiring enough good data from the SQ for the training and testing of a French based SPI AI agent.

6.3.2 User Requirements

The following section lists the functional requirements of Resolute. They were identified and accumulated during the course of interacting with the SQ.

REQ1: Original documents must remain unmodified

- User level Requirement: The system shall keep the original document that was imported into the system unchanged and in its original file format and encoding.
- Use case scenario 1: User imports an original document acquired from some unknown source. User/system requires changes to the original document (e.g., convert to text, clean

up meta-data, …), system automatically creates a copy of the document. User/system modifies copied document. User decides that document can be used as evidence. System allows user to export the unmodified original document.

- Use case scenario 2: User wishes to view original document at any time. User selects option to view original document. System automatically exports the original document and instructs the operating system to execute the appropriate program (based on filename extension) to view the original document.

REQ2: Document management

- User level Requirement: The system shall allow a user to organize documents by case identifiers and provide the ability to archive documents.
  Use case scenario 1: User creates a dataset with a chosen name. User imports a set of documents related to a case into the dataset. User works with documents in dataset. When work is completed, system allows user to move dataset to an archive folder.
  Use case scenario 2: User wishes to move, rename, copy or delete a dataset. System provides facilities to move, rename, copy or delete a dataset.

REQ3: Convert original documents to text and normalize them

- User level Requirement: The system shall provide facilities to convert original documents from their native encoding to text. The system shall also provide facilities to normalize the structure of a text document without the need to write special programs.
  Use case scenario 1: User imports a set of documents of different types of encodings and formats into a dataset. User initiates system function to convert the original document to text. System takes a copy of the original document and converts it to text. System automatically imports the converted text document into the dataset. User has the option to edit the converted text document.
  Use case scenario 2: User is required to normalize the format of a chat prior to analysis. System provides facilities to normalize a converted text chat into a consistent format as efficiently as possible. System will utilize normalization information from other documents when possible. A scoring mechanism will inform the user when chats have not been normalized properly.

REQ4: Export documents

- User level Requirement: The system shall provide an export facility for the original document and all intermediate documents.
  Use case scenario 1: User selects document(s) and activates system export function. System prompts user for which directory to place the files. System copies to specified directory all files related to the selected document(s): original document, converted to text document (if available), normalized document (if available).

REQ5: Chat Anonymization

- User level Requirement: System shall have the capability of obfuscating data that can be used to determine the identity or location of an author in a chat.

- Use case scenario 1: User converts a document to text or normalizes it using system features. User selects chats that require anonymization, and invokes the system anonymization feature. System prompts user for a directory in which to place the anonymized chats. System displays message when process is complete.

REQ6: Provide the likelihood that a chat contains predatory behavior

- User level Requirement: System will calculate the likelihood that a given chat contains a conversation between a predator seeking a sexual encounter with a minor.
  Use case scenario 1: User selects normalized chats for analysis. User initiates the analyze feature. System initiates an analysis of the chats using an AI agent. System displays the likelihood of predation as a value between 0.0 and 1.0 to indicate the confidence level of the prediction.

REQ7: Provide the likelihood that an author is a predator

- User level Requirement: System will calculate the likelihood that a given author is a predator seeking a sexual encounter with a minor.
  Use case scenario 1: User initiates actions defined in REQ6. User selects suspect chat and invokes the system chat viewer. System chat viewer displays all authors in the conversation along with the likelihood of being a predator as a value between 0.0 and 1.0 to indicate the confidence level of the prediction.

REQ8: Analysis of English and French chats

- User level Requirement: The system shall be able to analyze English and French chats (see REQ6-7). Note that for French chats this requirement is contingent of constraint CON4.
- Use case scenario 1: User initiates actions defined in REQ6. System detects language of chat and invokes the appropriate AI agent to calculate the likelihood of predation for the conversation and author.

REQ9: Chat viewer

- User level Requirement: The system shall enable a user to view a normalized chat with the following capabilities:
  - Each chat line formatted so that the reader has access to the complete text
  - Color coding for each author with the ability to remove an author from the conversation
  - If analysis has been performed (REQ6) the system will display the likelihood of each author exhibiting predator behavior.
- Use case scenario 1: User selects normalized chat. User activates system chat viewer. System chat viewer displays normalized chat with each line formatted so that the reader has access to the complete line of text (e.g., line wrapping). User can select from the list of authors to color code or remove chat lines authored by the selected author.
- Use case scenario 2: User initiates actions defined in REQ5. User selects normalized chat. User activates system chat viewer. Each author is given a likelihood of exhibiting predatory behavior. User uses chat viewer as defined in the previous scenario.

6.3.3 Non-Functional Requirements

The following section lists the non-functional requirements of Resolute. They were identified and accumulated during the course of interacting with the SQ.

NFR1: Performance and Usability

- System does not impede current productivity.
  The system should not take longer or take more effort than it currently takes to process documents manually.

NFR2: Capacity

- Store, access and manage a large number of documents quickly and efficiently.
  The expectation is that the system should be able to store an upper limit of one million documents.

NFR3: Security

- In addition to infrastructure security the data management facilities should be able to store and protect the data from unauthorized access.

NFR4: Scalability

- The ability to store and access the data from a local or remote data store, with the optional capability to distribute the databases across multiple servers.

6.4 Iterations of Resolute

There are 3 versions of Resolute:

- A prototype that was built early on to establish requirements and expectations with the SQ.
- Resolute version 1.0, which contained all functionality except SPI analysis, however; it had issues with useability and performance that required fundamental architectural changes.
- Resolute version 2.0, is the version described throughout this thesis.

This following sections briefly describe the different versions of Resolute and what each part played in defining the final version.

6.4.1 Prototype

The Resolute prototype served as a useful tool in generating ideas, discussions, and establishing expectations with the SQ. It allowed us to discover requirements that would otherwise have been missed. It also allowed us to establish common ground with regard to the overall structure of the application.

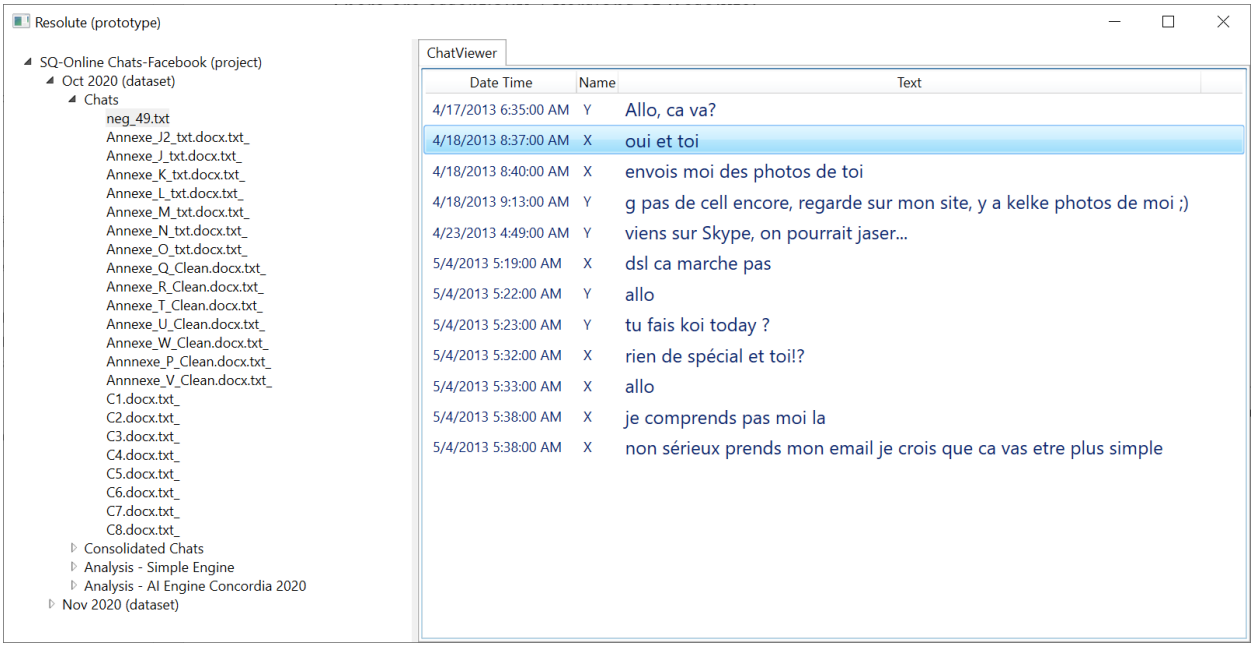

Figure 6-2: Resolute Prototype Main Screen

Development efforts for the prototype were kept at a minimum. The prototype did not make use of a database. A few chats were manually normalized for demonstration purposes. The GUI used basic functionality of the WPF common controls.

The prototype proved useful during development of version 1.0 as well. It gave us a picture of all the features which allowed us to think about how to bring these features together in a cohesive manner.

Refactoring of version 1 was reduced because we could see further down the development pipeline and plan appropriate development patterns early on. It also allowed us to make better architectural decisions early on.

The effort and time taken to build the prototype paid off many times over by serving as a very valuable tool in gathering requirements, setting expectations, and reducing the overall time and effort required in designing, and building the actual application.

6.5 Workflow

The main workflow for a given document (i.e., chat CON3) in Resolute is:

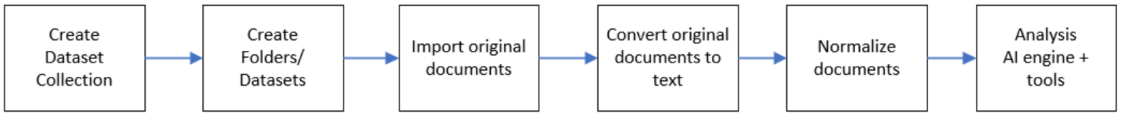

Figure 6-3: Resolute Workflow

- *Dataset Collection*: Resolute provides an administration utility that allows for the creation of Dataset Collections. A dataset collection translates into a MongoDB Collection (i.e., database). Dataset collections are designed to be able to hold large amounts of data (NFR2).
  One or more collections can be created and are intended to be used to partition datasets as an organization sees fit (REQ2). For example, law enforcement may choose to partition datasets based on forensics and cybercrime agencies.
- *Datasets*: Datasets hold sets of documents. This is where documents are imported, normalized and analyzed (REQ3). Datasets can be created and deleted within the Resolute GUI interface (REQ2).
- *Folders*: Folders hold one or more folders or datasets. They allow for simple grouping of datasets. Useful for organizing, folders can be created and deleted within the Resolute GUI interface (REQ2).
- *Import*: Import original documents in their native formats (REQ1, CON3). For example, Microsoft Word, PDF, text documents with different encodings, XML, JSON, etc. Imported original documents are never changed by Resolute, and can later be exported as exact copies of the originals. This is important for agencies building a court case against a suspect.
- *Convert to Text*: Converts the original document to text (REQ3). The original document is not changed. Instead, a text version is created and linked to the original document by Resolute (REQ1).
- *Normalize*: The converted text document is normalized into a standardized format (REQ3). The associated original and converted text documents are not changed, but a new normalized document is created and linked (REQ1).
- *Analysis of the chats*. This includes an AI agent that automatically scans the normalized chats and assigns a probability that a given chat contains dialog of sexual predation of a minor (REQ6-REQ9).

6.5.1 Resolute Version 1.0

Resolute version 1.0 is an almost full featured application including support for; anonymization, and normalization.

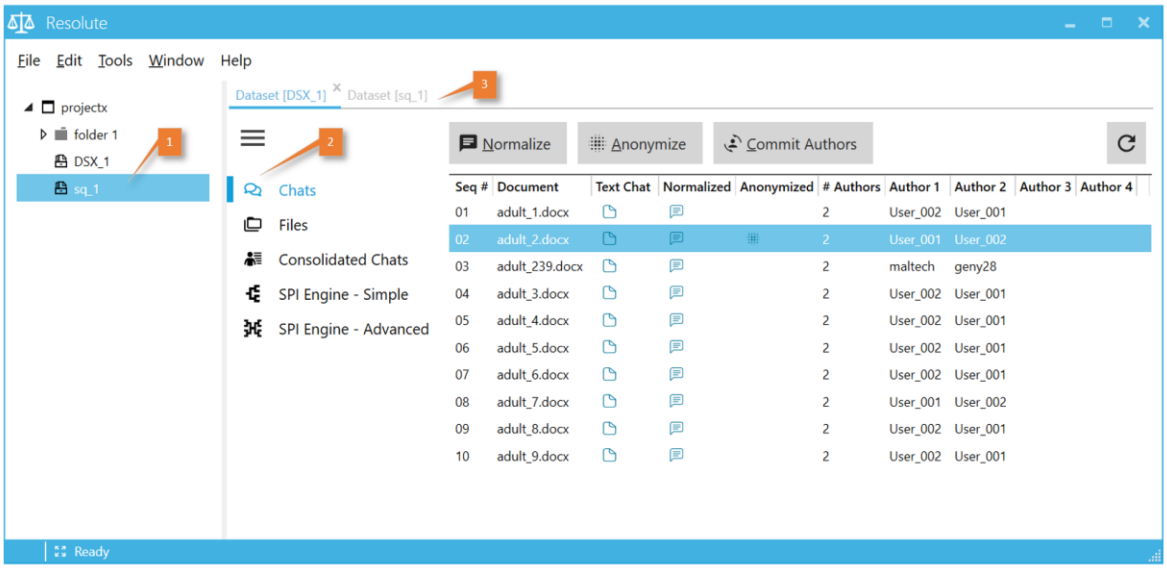

Figure 6-4: Resolute version 1.0

Figure 6-4 shows a screen capture of Resolute version 1.0. Point #1 highlights a dataset. Double clicking a dataset brings up the main dataset screen in a tabbed document window (point #3). Point #2 shows a menu for the different parts of a dataset. Note that the menu was collapsible to show only the icons to save on screen space.

The sections below highlight some of the more interesting characteristics and architecture of version 1, and we discuss the problems with it. Resolute version 1.0 is a functional application, but because of the problems we encountered, it was not a practical system to use on a day-to-day basis. The changes required to fix the problems lay deep in the architecture, and as a result could not easily be refactored. As a result, the hard decision to abandon most of the work done in v1 in favor of starting over in Resolute v2 was necessary. Some of the code from version 1 was brought over with minor changes but the majority of the code in v2 is new.

Data Management

Resolute v1.0 does not have a backend database to manage chats. Chats were processed as stored by the filesystem.

A dataset would require a path to a folder, and Resolute would read the documents from that folder. Original documents were not altered, instead Resolute would create sub-directories in the document folder to hold the converted text chats, anonymized chats, and the normalized chats.

We had considered using SQL as a database, but SQL requires some technical expertise to manage, and it enforces a rigid structure on the data, which may in the future complicate updates to the software.

Later in the development of v1 when we started working with thousands of files, we started noticing significant slowdowns in performance. Operations that should have taken seconds to

perform where taking minutes. The more files in a folder the worse the performance. There was no way of optimizing this as the performance issues were coming directly from the operating system (Windows). This resulted in non-conformance to the non-functional requirement of performance (NFR1).

It was clear that allowing the operating system to manage the documents was not going to be sufficient. In addition to slow processing in Resolute, the burden of managing the documents was being shifted to the users of the software after work on a particular set of documents was done. They would have to concern themselves with what to do with the files once work on them was complete, and moving a great number of files outside of Resolute was still very slow.

For version 2.0 we revisited using a database management system to store and manage the documents. We looked for options other than SQL. MongoDB turned out to be a good alternative.

MongoDB falls under the category of a NoSQL database management system. That is, data schemas are not rigid structures. We could choose to add or remove fields from the database without having to require complicated processes to updates of Resolute. Also, MongoDB is very performant. It allows for fast access to data, and allows a great many documents to be stored.

A basic local/server installation of MongoDB is easy to install, not requiring much technical expertise to manage. It is also good to know that MongoDB is not limited to a basic setup. It allows for more complicated setups such as clustered distributed architectures.

MongoDB was chosen as the back-end database in Resolute v2.0, and it turned out to live up to our requirements and expectations (NFR1-NFR4).

User Interface

One issue we discovered with the UI after using it for a while, is that because the tabbed windows (point #3 in Figure 6-4) were not detachable, it required a user to switch back and forth between tabs when working with two or more documents. This proved to be disorienting and unproductive, and we decided that it did not adhere to the non-functional requirement of useability (NFR1).

We wanted to make the tabbed documents detachable. This would allow a user to view more than one document at a time. The following is an example of detached windows in Resolute v2.0:

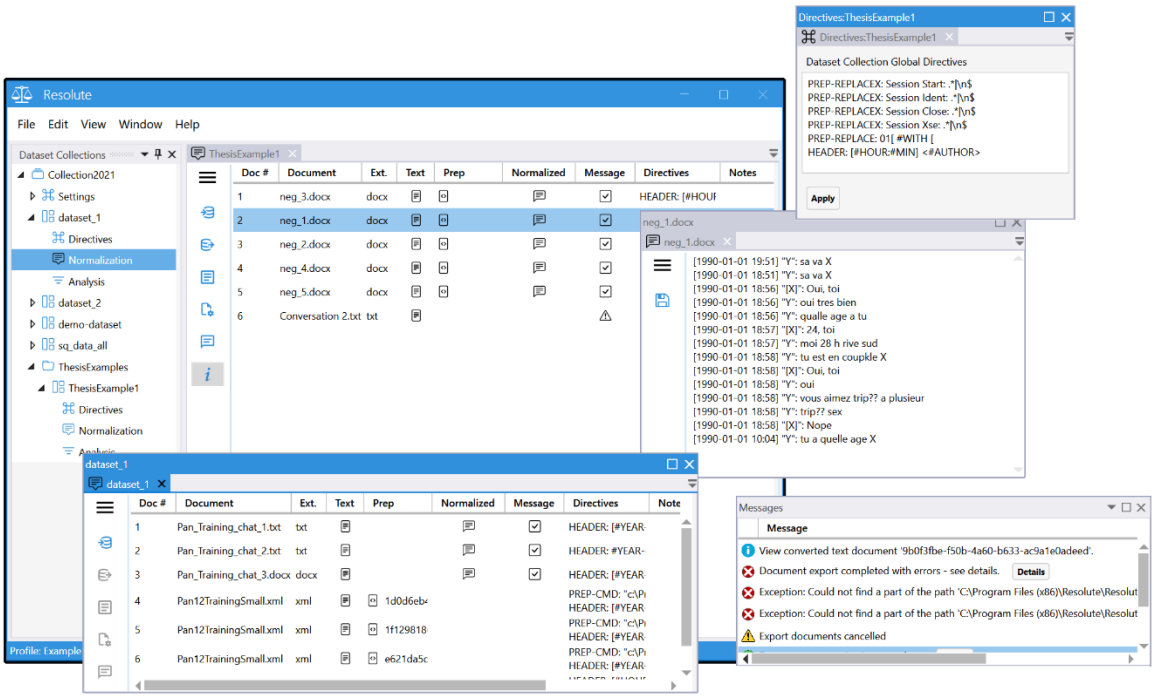

Figure 6-5: Detached document windows in Resolute v2.0

In Resolute v2, a user can detach a window by simply selecting and sliding a window from the tab. Detached windows provide even more benefits for systems with multiple screens (NFR1).

Another issue with the UI in v1 was with the way we supported the different options of a dataset in the dataset window. Clicking an option from the dataset menu (point #2 in Figure 6-4, and #1 in Figure 6-6) would instantly change the current screen to the selected option screen. For example, in Figure 6-6, clicking on the Files option replaced the current view (e.g., chat list) with the dataset file management screen. A user would have to click back and forth between two screens to see the effect on the dataset. This design after working with the software for a while

was found to be disorienting and unproductive, and it did not adhere to the non-functional requirement of useability (NFR1).

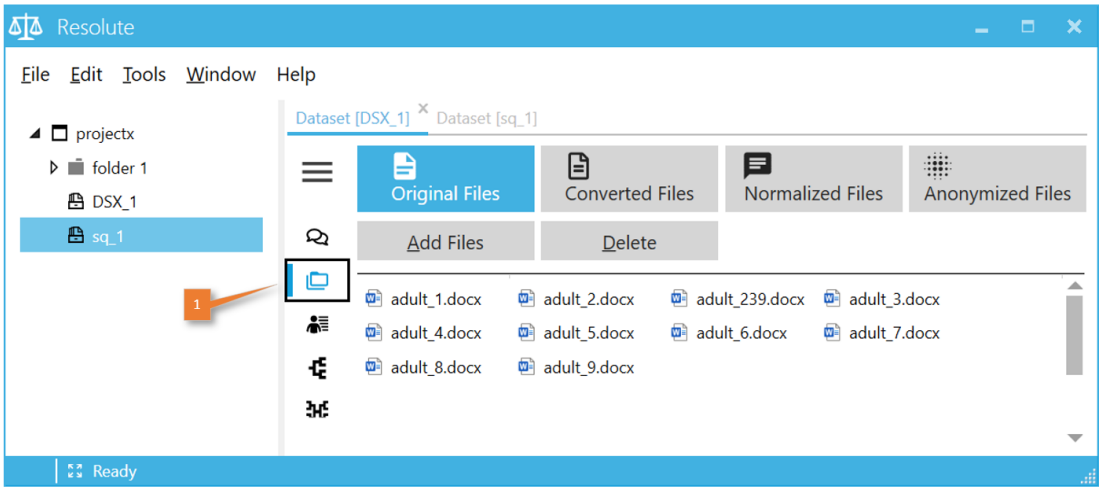

Figure 6-6: Resolute v1.0 dataset file management

We wanted to make the system open up a new window instead of switching the current screen on the user. Doing so would mean we would have to remove the menu altogether and come up with a new design of how to handle the different parts of a dataset. Also, the top buttons on the screen were taking too much screen real-estate.

The changes we wanted to do to the UI in v1 could not be done via refactoring of the code alone. The code associated with the view had to be redone.

Templates and Directives

Normalization templates are a necessary part of normalization. Resolute v1 did not provide a proper facility for managing them. Instead, it would expect to find the templates either as directives specified in the converted text document or in an external file stored in the folder containing the dataset files.

Furthermore, it was discovered during development that other directives were needed, for example the preprocessing directives.

While working with the SQ on the usability aspect of Resolute, it became clear that the lack of directive support from the UI confused users, and it did not adhere to the non-functional requirement of useability (NFR1). Resolute needed to provide a proper facility to manage directives at the dataset collection, dataset and document level.

6.5.2 Resolute version 2.0

Resolute v2.0 is the latest version of Resolute that was shipped to the SQ.

The Resolute prototype and version 1.0 played an essential part in the design of version 2.0. Taking the decision to stop development on version 1, and rewrite large parts of the code in version 2 was not taken lightly but, it did not adhere to many of the non-functional requirements (NFR1-NFR4). In retrospect, however, it turned out to be a sound decision.

6.6 Architecture

The system illustrated in *Figure 6-7: System Architecture* defines the architecture we propose to define and build.

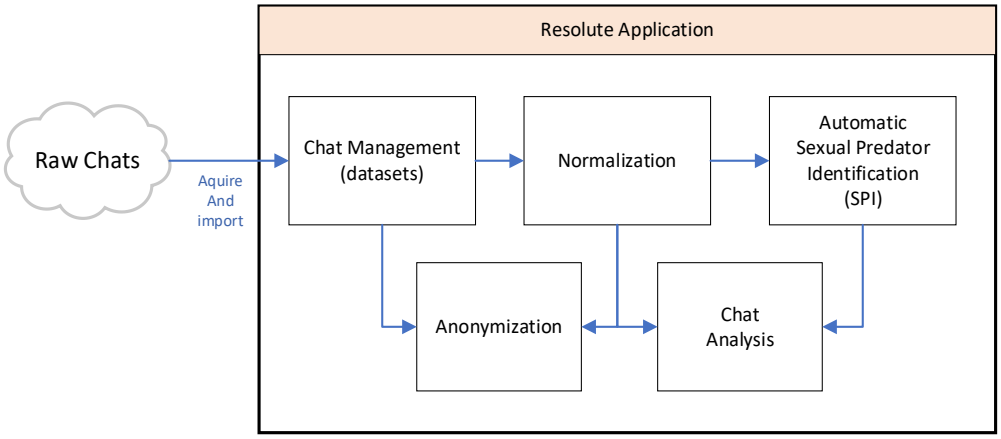

Figure 6-7: System Architecture

- Resolute is an application that is responsible for providing a user interface, and to bring together the different functions in a cohesive, productive environment (REQ2, NFR1).
- Chat Management is an integral part of Resolute. It is capable of managing potentially millions of chats. How Resolute allows a user to organize the data, and how performant it is at storing and retrieving chats are essential requirements (NFR2).
- Normalization is the function of transforming raw chats into a standardized format suitable for Sexual Predator Identification (SPI) and analysis. Ideally this function should be a totally automatic process (REQ3). It also needs to be performant in order for it to be practical (NFR1).
- Automatic SPI is the function of evaluating chats and calculating a probability that a given conversation and author contain evidence of sexual predation of a minor (REQ8). How well and how quickly chats can be evaluated are essential requirements (NFR1).
- Anonymization is the function of obscuring named entities in the chats. This is a function that provides law enforcement the ability to distribute chats to outside personnel if required (REQ5). Section *6.7.4 Anonymization* briefly describes anonymization.

- Chat analysis is an integral part of Resolute. It is responsible for presenting the user with information about a chat, for example, the probability of SPI, a viewer that presents the chat in a readable form including color coding of authors, etc. (REQ8).

6.6.1 Architectural Patterns

Resolute is built using Microsoft C# and the .NET framework. The GUI utilizes the .NET 'Windows Presentation Foundation' (WPF) framework, which supports a well-defined separation between the presentation layer and the model.

The high-level architecture of Resolute follows the Model-View-ViewModel (MVVM) pattern [30]. MVVM is a pattern that defines a clear separation of the business and presentation logic of an application. It is very close in nature to the traditional Model-View-Controller (MVC) pattern [31], but is more tailored to a GUI application using WPF.

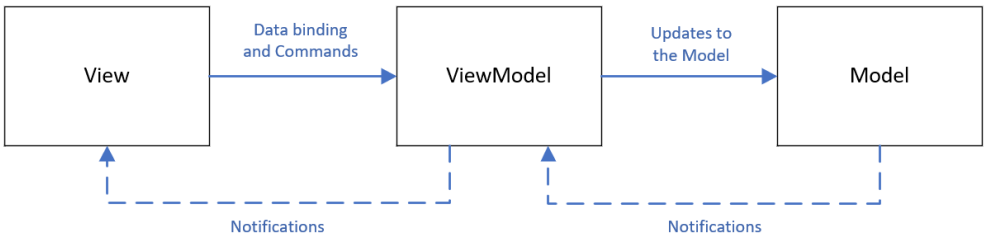

Figure 6-8: MVVM Pattern

The view in WPF has 2 distinct components;

- Extensible Application Markup Language (XAML) which is a declarative language that is based on XML. This is where the GUI screens are defined, i.e., XAML is used to define and place the controls on the GUI screen. There is no C# code in XAML, it is just XML that defines the GUI screen.
- Code-behind component which is the C# code that WPF invokes in response to events from the GUI controls that are defined in the XAML. This code is used mainly to control aspects of the UI visual behaviors based on events. Business logic is not implemented here. In fact, this layer implements very little code.

The view-model interacts with the view via properties and commands, to which the view contains data-binding definitions in XAML to the view-model properties and commands. The view-model notifies the view of state changes using notification events. The properties and commands of the view-model define the functionality offered by the UI, whereas the view determines how that functionality is to be presented in the GUI.

The view-model is also responsible for coordinating events from the view that require interaction with the models. Each view-model provides data from a model in a form that the view can easily consume.

The model contains the code associated with the applications' data and business logic, i.e., the applications' domain model. As we will see in the next section, the model layer will be further divided into other layers.

Resolute adheres strongly to the MVVM pattern, and it has proven to have worked very well in managing the code complexity during development.

6.6.2 Application Architecture

The general architecture of Resolute is based on the MVVM pattern as described in the previous section, but it also implements other patterns in the model as shown in the following diagram:

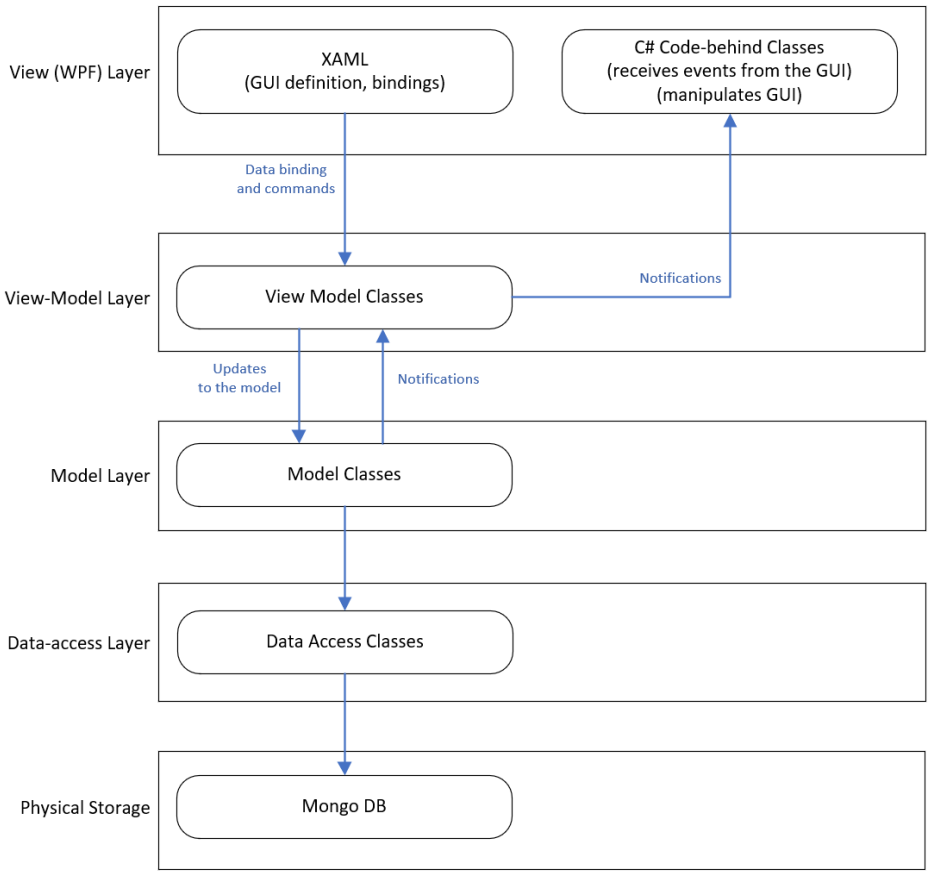

Figure 6-9: Resolute General Architecture

The Model layer contains the business (i.e., domain) logic of the application. The model classes go through the Data Access layer to retrieve, create, update and delete data from the Mongo DB.

Mongo DB was chosen as the data repository because:

- It's free and open source.

- It's a NoSQL, i.e., non-relational database. Records are stored as documents, and allows fields to vary from document to document, i.e., record structure is not rigid as it is in SQL for example. This allows Resolute to evolve iteratively without complications with deployments. Also, documents map naturally to C# objects.
- Provides high-performance data I/O, is a distributed database, provides high availability, horizontal scaling, and geographic distribution (NFR1-NFR4).
- Easy to install and maintain, which simplifies the installation and setup of Resolute.

Tests were conducted in Resolute where over one million chats were imported without any noticeable degradation in performance (see *6.8 Experiments and Results*).

6.7 User Interface

The following sections describe the Resolute GUI interface. Not all functions of Resolute are described in this section. We focus mainly on relevant topics found in this thesis. Full details about all the functions are provided in the Resolute Users Guide.

6.7.1 Main Window

The following is an example of the Resolute main GUI window after it's been setup and loaded:

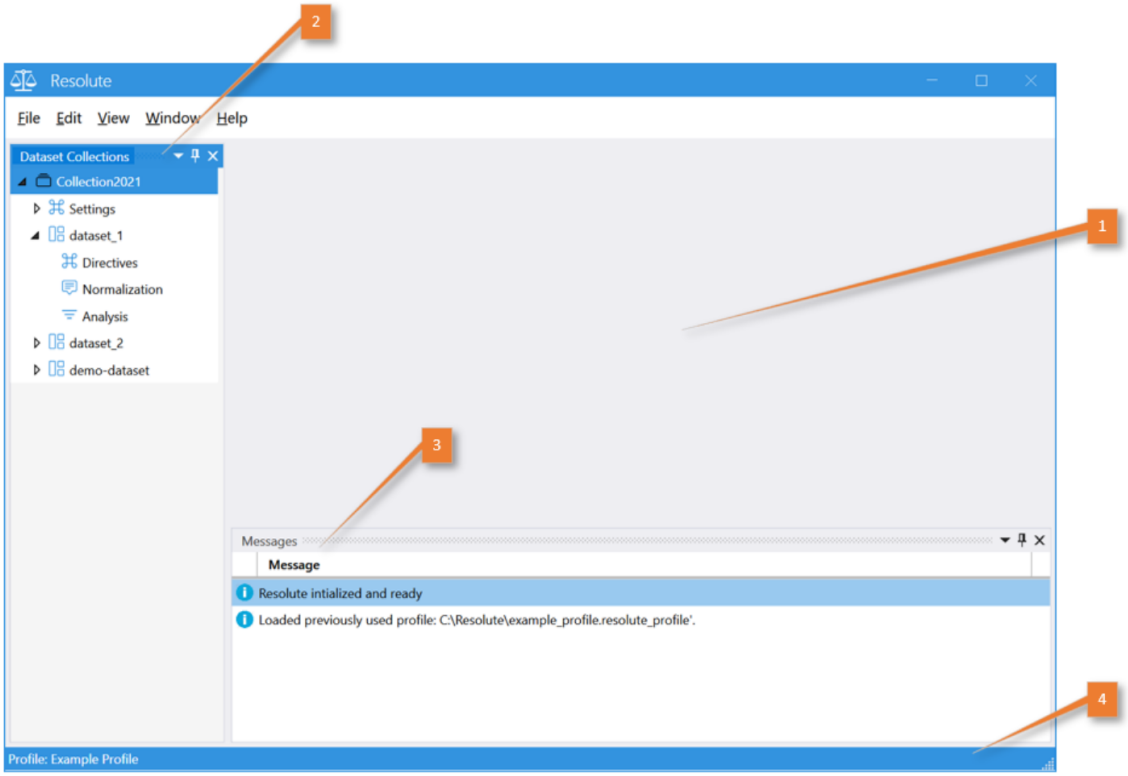

Figure 6-10: Resolute Main Screen

1. The main work area. Different windows will be opened here depending on the options selected.
2. Dataset Collections tool window. Users can navigate, create and manage datasets from here (REQ2).
3. Messages tool window. System messages are displayed here. Some messages include detail buttons that will open another window showing details of an operation, and or extra error information if available. *Figure 6-11: Displaying Message Details* illustrates an error message with extra details (1), and the corresponding window that was opened after pressing the 'Details' button (2). (NFR1)
4. Status bar. This area contains information about the current state of the application. For example, our session is based on the 'Example Profile' profile.

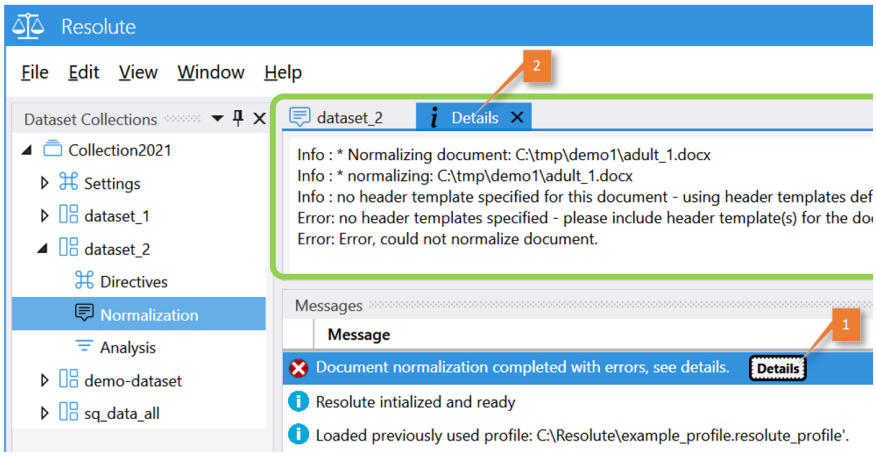

Figure 6-11: Displaying Message Details

All windows and tool windows in Resolute are detachable and can be docked in any predefined sections of the view (NFR1). For example:

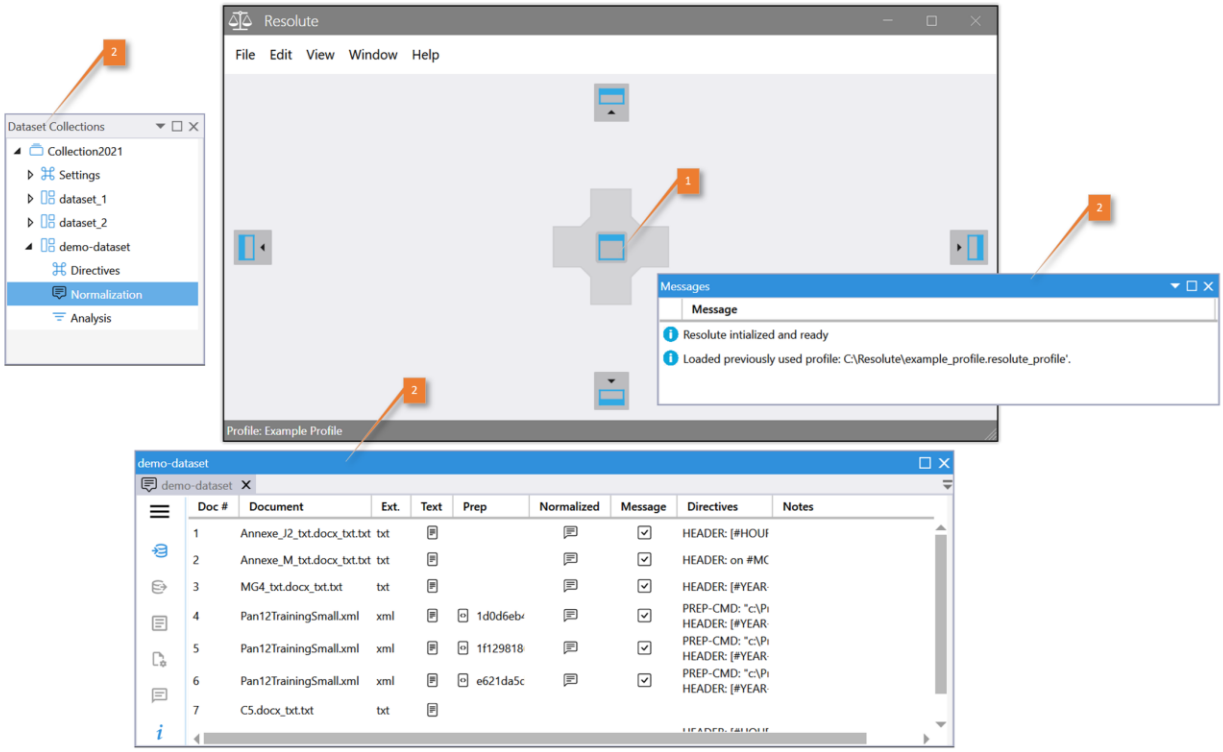

Figure 6-12: Example of detached windows and docking points

Point #1 are the areas where the windows can be docked. Point #2 are detached windows.

6.7.2 Directives

Directives hold normalization template and pre-processing specifications. They also allow the user to control the behavior of different functions throughout Resolute (NFR1).

There are 3 different locations where a directive can be placed, and where it is placed defines its' scope as follows:

- Dataset Collection level: directives specified here apply to all datasets and documents contained within the dataset collection.
- Dataset level: directives specified here apply to all documents contained within the dataset.
- Document level: directives specified here apply to a document.

The following diagram illustrates where directives can be specified:

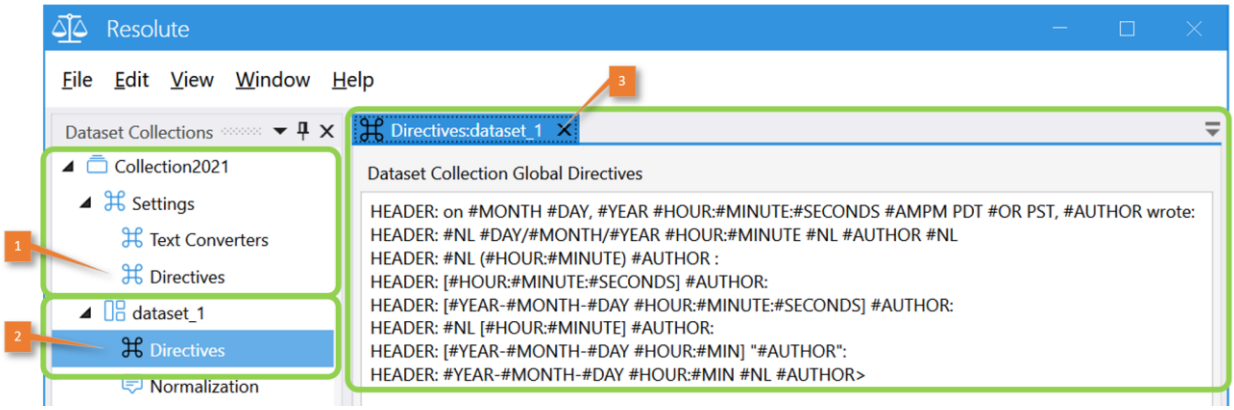

Figure 6-13: Directives at the Dataset(2) and Datsset Collection(1) level

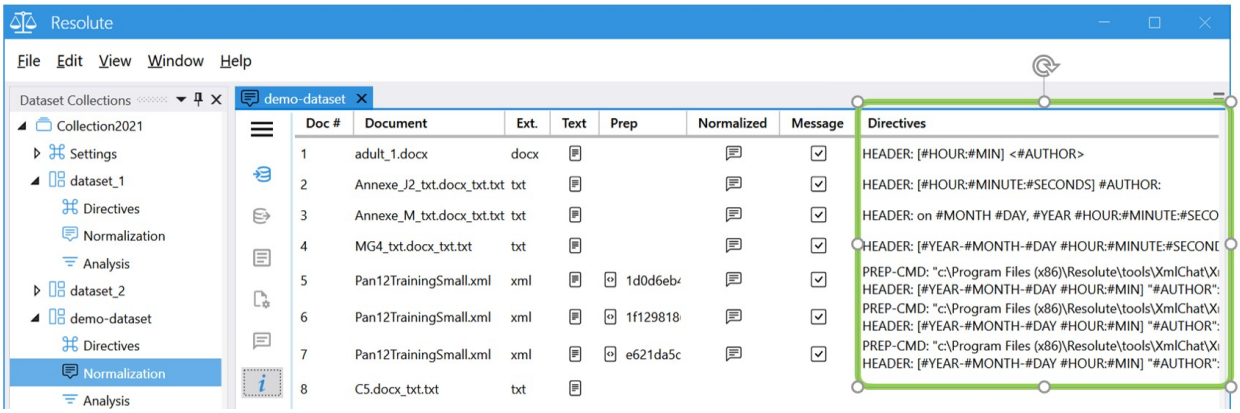

Figure 6-14: Directives at the document level

HEADER Directive

The header directive allows for the specification of normalization templates.

The syntax of the header directive is:

HEADER: *<template-specification>*

Where <*template-specification*> can be any of the following:

#NL	New line expected
#YEAR	Year part of a date
#MONTH	Month part of the date The month can be a number (1-12) or the word in English or French (e.g. Sep, Sept, September, Septembre)
#DAY	The day of the month (1-31)
#HOUR	The hour part of the time
#MIN #MINUTE	The minute part of the time

#SEC #SECONDS	The seconds part of the time
#AMPM	If the time part contains AM or PM
#WEEK	Day of the week in English or French in short or long format. e.g.: Mon, Monday, Lun, Lundi
#AUTHOR	The author of the chat. Author names are usually delimited by some sort of symbol (':'), if they are not then the normalizer assumes the author name is less than 5 words.
<any other character(s)>	Any other character other than the special tokens defined above will be expected in the chat header as is. For example '*HEADER: hello #AUTHOR:*' will expect to find the word 'hello ' followed by the authors' name and ending with a colon.

Example:

[2021-08-31 02:17] Mike: Hello there [2021-08-31 02:17] Julie: hey [2021-08-31 02:18] Mike: How are you doing today/night? [2021-08-31 02:18] Julie: i went to the beach [2021-08-31 02:18] Julie: and you?
HEADER: [#YEAR-#MONTH-#DAY #HOUR:#MIN] #AUTHOR:

In addition to the special template tokens #YEAR, #MONTH, etc. the other characters **[- :]** in the template including spaces are necessary parts of the header. The normalizer expects to find those characters at their respective positions in the chat header.

Preprocessor Directives

The preprocessor directives are used to alter/remove chat structure information from a chat prior to normalization. The following list briefly summarizes the syntax of the preprocessor directives:

- **PREP-CMD:** <*preprocess-tool-executable*> <*input-file-argument*>**%input_file** <*output-folder-argument*>**%output_folder** <*other-arguments…*>
- **PREP-REPLACE:** <*search-text*> [#WITH <*replace-text*>]
- **PREP-REPLACEX:** <*regular-expression*> [#WITH <*replace-text*>]

Note that the preprocessing discussed here is not the same as the preprocessing required for classification (see *5.5 Preprocessing*).

PREP-CMD stands for 'preprocessor command' and it instructs Resolute to invoke an external program or script to preprocess a chat. The function of the external program is to remove any information in the chat not related to the chat header or text. When a preprocess directive is encountered Resolute exports the text chat and passes the exported filename as an input

parameter to the program. An output filename is sent to the program as a parameter as well. Resolute expects to find the resultant output preprocessed file once the chat has been preprocessed. The output file is imported back into Resolute, which will become the file that will be normalized.

The PREP-REPLACE and PREP-REPLACEX directives perform the same function as PREP-CMD but are supported internally in Resolute. PREP-REPLACE allows for simple text search and replace and PREP-REPLACEX allows for more complex search and replace using regular expressions. Both directives allow for cases were a search and replace could convert a structured chat into an unstructured chat. For example, given the following chat taken from the SQ dataset:

```
Session Start: Wed Jul 13 19:51:27 2011

Session Ident: [X]

01[19:51] <Y> sa va X


01[18:51] <Y> sa va X



[18:56] <X> Oui, toi

01[18:56] <Y> oui tres bien
```

Preprocessing with the following PREP-REPLACE directives:

```
PREP-REPLACEX: Session Start: .*|\n$
PREP-REPLACEX: Session Ident: .*|\n$
PREP-REPLACEX: Session Close: .*|\n$
PREP-REPLACEX: Session Xse: .*|\n$
PREP-REPLACE: 01[ #WITH [
```

Gives the following pre-processed chat without the need to call or create an external preprocessing program using PREP-CMD:

```
[19:51] <Y> sa va X


[18:51] <Y> sa va X



[18:56] <X> Oui, toi

[18:56] <Y> oui tres bien
```

6.7.3 Managing Datasets

A Dataset is an object in Resolute where chat documents live. It is where functions like anonymization, normalization, SPI, and analysis happen on chat documents. The functions defined in this section map to requirements: REQ1-REQ4, NFR1.

A Dataset Collection, is a container for holding datasets and folders. Folders are essentially an organizational tool. They simply contain other folders and datasets.

In a typical setup, you could expect few dataset collections. Each dataset collection in turn can contain copious numbers of folders and datasets.

Resolute provides a separate admin console program to create or delete Dataset Collections. This is because a dataset collection is actually a MongoDB database and it requires some basic knowledge of MongoDB to create. The admin application is best used by DBAs and it is not directly accessible by everyday users of Resolute.

Resolute requires a user to define a profile which contains information about which dataset collections a user wants to work with. After a profile is defined or loaded Resolute displays the corresponding Data Collection tool window, with the users' defined dataset collections, for example:

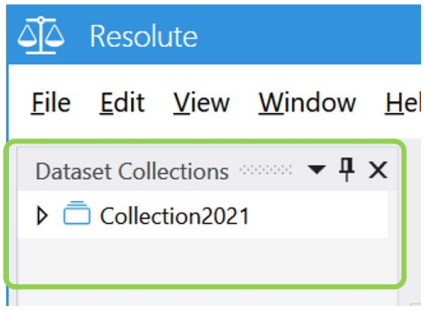

Figure 6-15: Dataset Collection tool window

Datasets or folders can be managed by right-clicking on the root dataset collection, dataset or on a folder and selecting from a context menu which action to perform. For example:

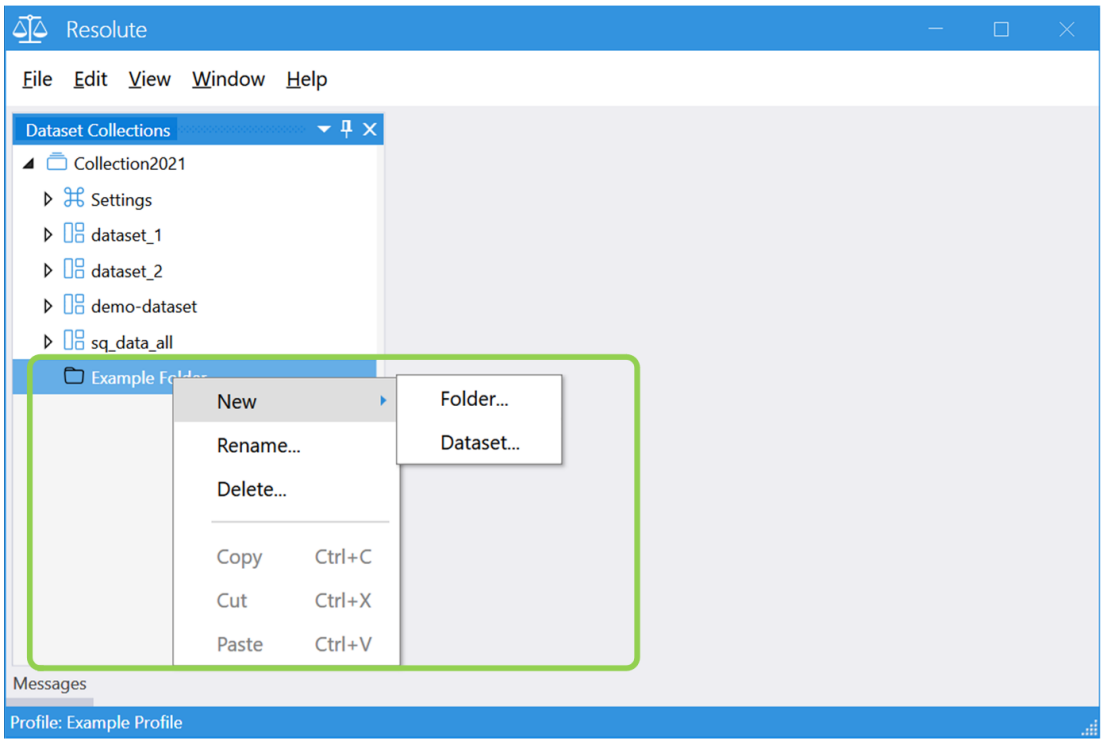

Figure 6-16: Dataset Collection management context-menu

The context menu is setup to disable the functions which are not applicable to the item selected in the dataset collection tree. For example, if a dataset is selected then the New|Folder and New|Dataset options would be disabled. A user can also copy, cut and paste either a dataset or a folder to another folder.

Working with chat documents requires a dataset to have been created. Opening up the subitems of a dataset from the dataset collection tool window shows the dataset workflow entities, for example:

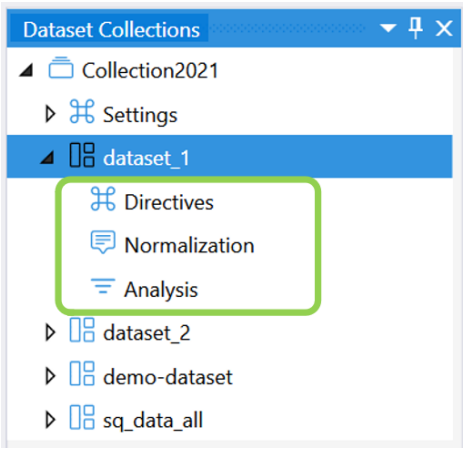

Figure 6-17: Dataset workflow entities

The ‘Directives’ item, allows a user to define any directives which apply only to the chat documents contained in this dataset. See section *6.7.2 Directives* for more information on directives. Items ‘Normalization’ and ‘Analysis’ are detailed below.

Double clicking on an item will bring up the corresponding window in the workspace area. For example, clicking on ‘Normalization’ will bring up the main normalization screen:

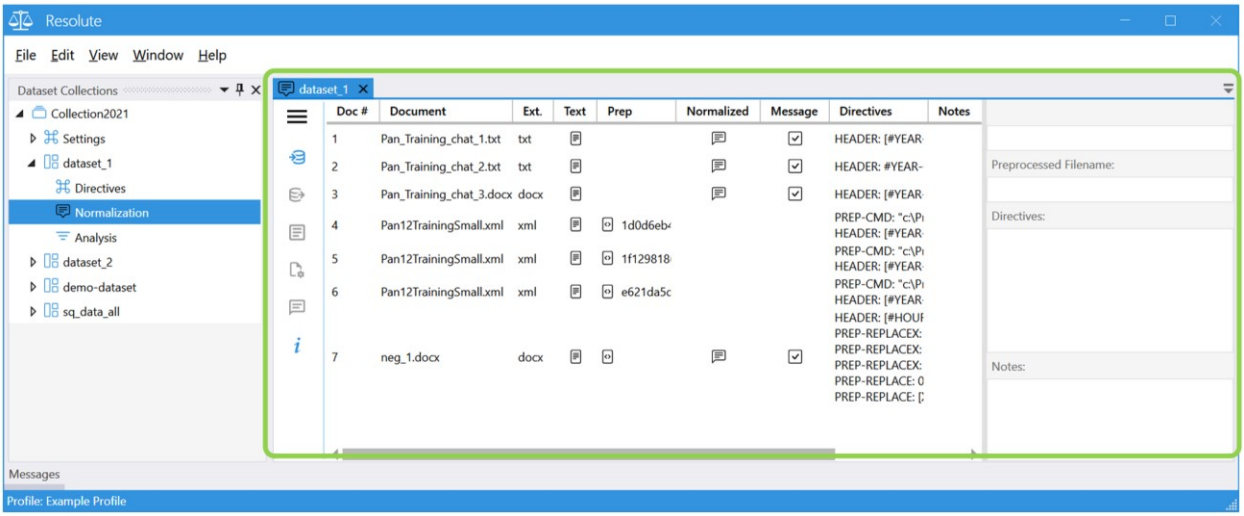

Figure 6-18: Normalization screen

6.7.4 Anonymization

Anonymization is the process of obscuring named entities such as author names, phone numbers, addresses, locations, and internet identifiers, etc. The functions defined in this section map to requirements: REQ5, NFR1.

Resolute offers anonymization with the idea of offering the SQ a tool that could assist them in generating enough French data to be used to train and test an SPI AI agent.

The anonymization feature in Resolute supports basic named entity recognition as follows:

- Author names identified during normalization
- URLs and IP addresses
- Postal codes

In addition, all numbers are changed to either #..#L for numbers under a certain configurable threshold, and #..#H for values above. Each # symbol generated replaces a digit. The threshold value between low and high is configured to the age considered a minor. In Quebec that value is 18. Since our entity recognition logic is limited, replacing all numbers obfuscates several numerical entities such as phone numbers and addresses.

Grouping different chats from the same authors improves SPI detection since it gives the AI agent more data to work with (refer to *Chapter 5: Sexual Predator Identification* for more

information). Therefore, obfuscating the author ids could not be random, since it is important that the same author in different chats keep the same generated name. In addition, the obfuscated names must not be reversable, i.e., not possible to figure out the original authors name from the generated name.

To address this issue Resolute uses the following logic to generate a seemingly random name that will generate the same identifier for the same author but that cannot be reverse engineered:

- Authors' name is encrypted using SHA-256 which is a cryptographic one-way hash function.
- A CRC-16 checksum value is calculated from the encrypted value.

The resultant CRC-16 value is used as the authors' obfuscated name. The extra step of using the CRC-16 value as the name was used to shorten the names to a reasonable length. This shortening of the name to a 16-bit value does mean that there are only 65,535 possible author names, and by applying the pigeon hole principal this implies that some different author names will map to the same generated name. This of course can be mitigated by increasing the generated identifier to a CRC-32 value, but at this time we felt that the CRC-16 value was good enough for our purposes.

Resolute also allows non-normalized text chats to be anonymized. The quality of the anonymization is compromised in this case as the author names are unknown at that point. In these cases, a warning message is presented to the user of this fact and that further manual anonymization is recommended.

The quality of anonymization in Resolute is by no means sufficient, for example it will not obfuscate street names or cities, or misspellings of the author names. The goal of anonymization in Resolute is to provide help, and make the process more efficient. We make it clear to the user that anonymized chats need to be manually verified before they are released.

Anonymization is executed from the normalization screen by selecting the documents, right-clicking and selecting 'Anonymize' from the context-menu. The following is an example:

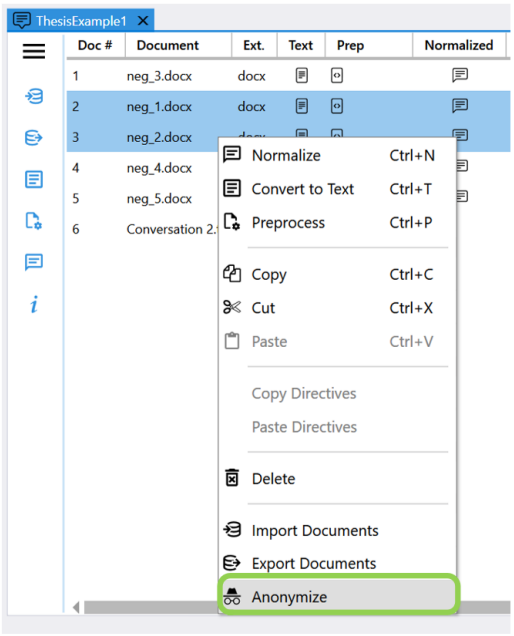

Figure 6-19: Anonymize documents

Resolute will then present the user with a dialog to choose which folder to save the anonymized chats.

6.7.5 Normalization

Importing, conversion to text, preprocessing and normalization are all functions that can be performed from the dataset normalization screen. The functions defined in this section map to requirements: REQ3, NFR1.

The first step requires that a user import the original chat documents. These documents should be in their unaltered native file format. Once imported double clicking on the document entry will bring up the original document for viewing. For example:

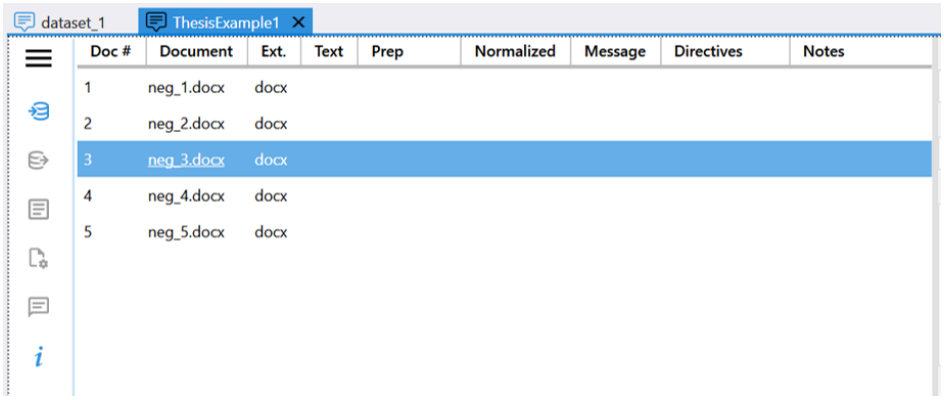

Figure 6-20: Importing documents

Double clicking on document ‘neg_3.docx’ will bring up the document in Microsoft Word. Resolute automatically exports the document to a temporary folder and filename, so editing the document will have no effect on the original. The application that is used to view the document depends on the type of file which is indicated by the filename extension.

The next step is to convert the original text documents from their native file format into text. This is done by selecting which documents to convert and then selecting the ‘Convert to Text’ function. The resulting screen shows a text icon to indicate which documents have been converted, for example:

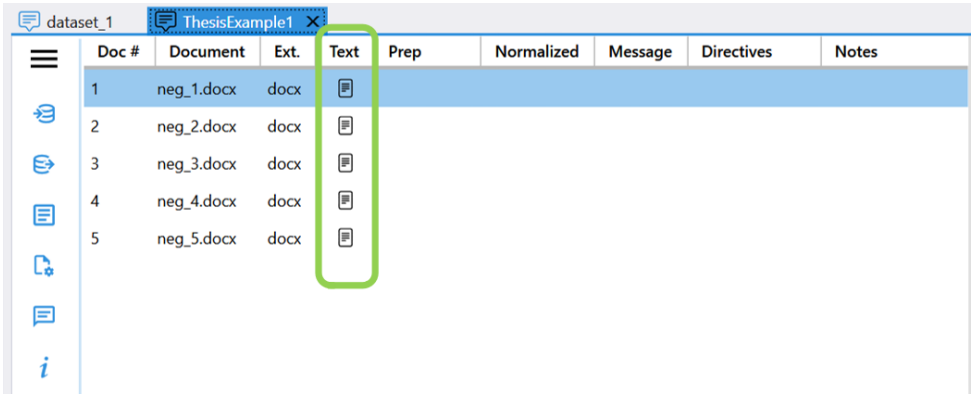

Figure 6-21: Converting to text

Clicking on the corresponding icon will bring up the converted text document in another tabbed window in Resolute, for example:

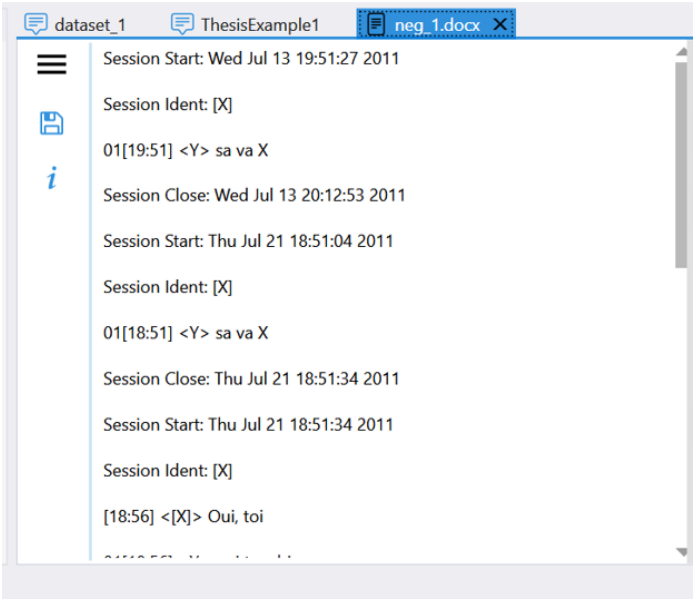

Figure 6-22: Editing a converted to text document

Resolute allows editing of the converted text document. This is useful for removing structure data from chats without having to provide preprocessing directives. This is only practical for a

small number of chats. If there are many chats that require editing then preprocessing them would be more efficient.

The next step after converting all the documents to text is to normalize them. Structured chats however require an extra preprocessing step at this point. Our example in Figure 6-22 shows a structured chat that can be preprocessed into an unstructured chat using only the built-in Resolute PREP-REPLACE directives. Setting up the replace directives at the dataset level and preprocessing the documents results in preprocessed icons being displayed for all documents successfully preprocessed. Note that the message panel will provide details of the preprocess operation. This is useful if there are errors encountered during preprocessing. For example:

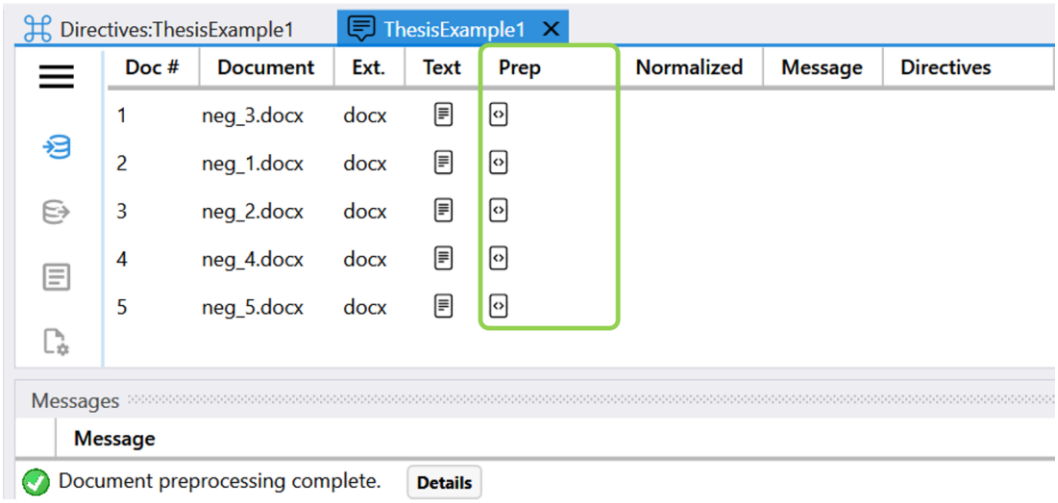

Figure 6-23: Preprocessing screen

Clicking on the preprocess icon opens up a tabbed window with the preprocessed output. The resultant document can be edited if desired, for example:

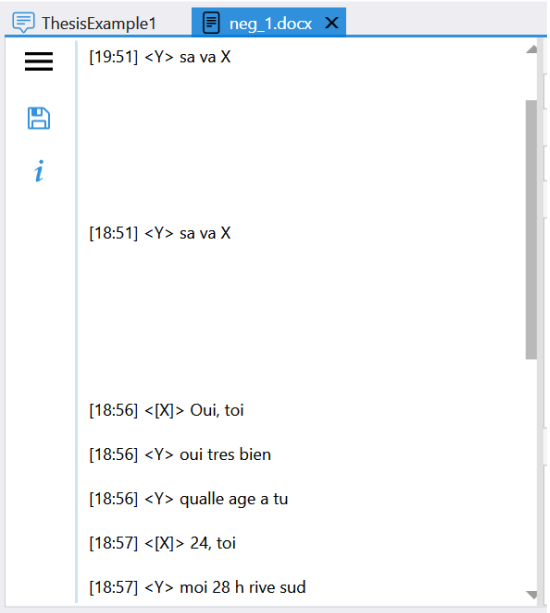

Figure 6-24: Preprocessed output editor

The next step is to normalize the documents. Before normalization can be performed the normalization template directives need to be specified. The chats in our example are of the same format (i.e., class), as a result we can specify the template directive at the dataset level and invoke the normalize function. An extra document was imported for the example below, in which the template was not specified. Upon completion a normalization icon is displayed for all the chats that have been normalized properly, i.e., the Normalized column in Figure 6-25;

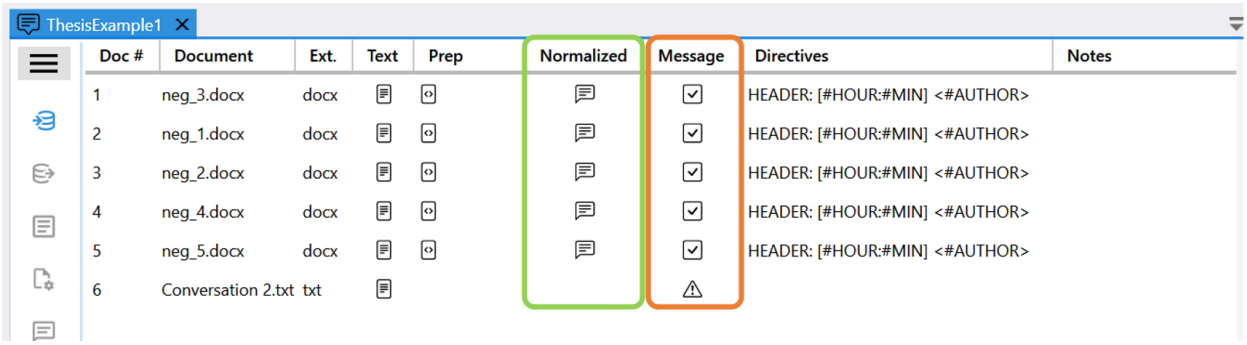

Figure 6-25: Normalized chats

Clicking on the normalized icon will bring up the normalized document in an editor, where the normalized document can be viewed or edited. For example:

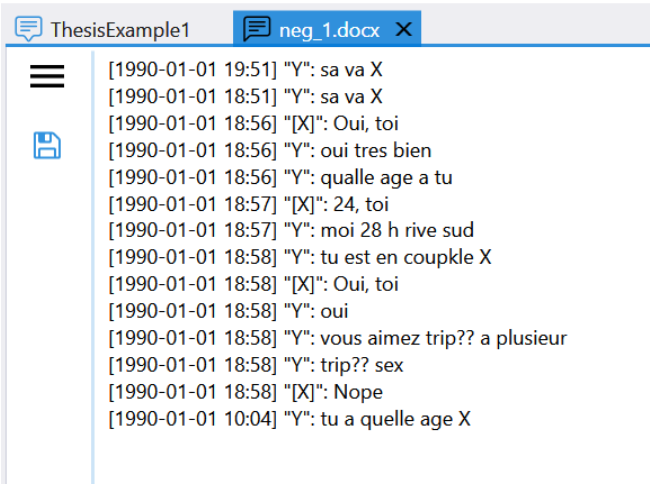

Figure 6-26: Normalized chat text

The icons in the Message column in Figure 6-25 indicate if the normalization operation was successful (check mark) or contained errors or warnings (warning icon). Clicking on either will bring up the details of the normalization operation for the corresponding document, e.g.:

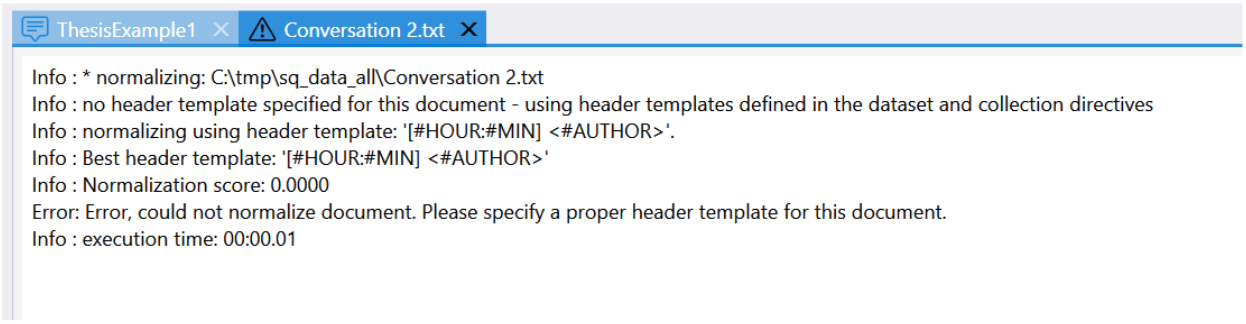

Figure 6-27: Normalization errors

Notice that in the Directives column in Figure 6-25, Resolute has populated that column with the template it deemed the best. That is where directives can be specified at the document level. Resolute populates that field so that the next time the document is normalized, Resolute will not have to go through all the templates specified at the dataset and dataset collection levels, which under certain conditions (e.g., many templates specified) can be a lengthy operation.

Once normalized chats can be analyzed as described in the next section.

6.7.6 Chat Analysis

The chat analysis tools can be accessed from the dataset collection tool window. The functions defined in this section map to requirements: REQ6-REQ9, NFR1. Selecting a dataset and clicking on the Analysis selection brings up the chat analysis window, for example:

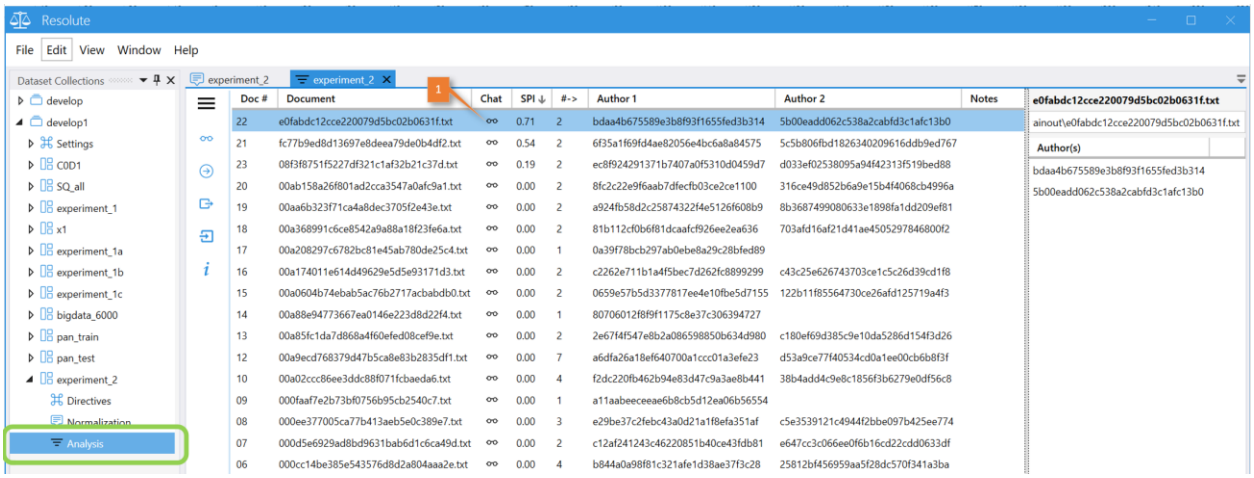

Figure 6-28: Dataset analysis window

The analysis window shows all the documents imported into the dataset from the normalization window. Documents that were normalized display some information about the corresponding chat, such as the number of authors in the chat, and the two authors with the most dialog in the chat. The 'chat' column displays a ' ∞ ' icon, which can be pressed to open up the chat in a viewer. Refer to the section 'Chat Viewer' below for more information. Documents that are not normalized do not display the chat viewer icon and offer no information on the authors.

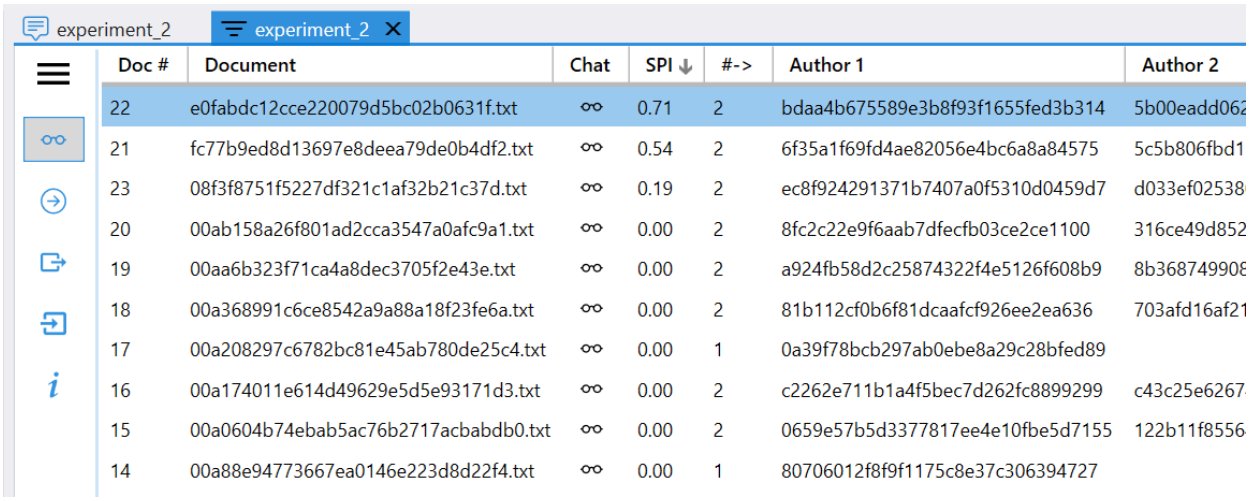

Figure 6-29: Example analysis of dataset

The SPI column displays the probability of predation of a minor that the AI agent assigned to the chat. The example chat in Figure 6-29 highlighted in blue shows a probability of 71% that it is a predatory chat. The top 3 chats are in fact predatory chats taken from the PAN2012 dataset.

Chat Viewer

The chat viewer displays a chat in a presentable manner to the user (REQ9). It also provides basic analysis tools such as color coding of authors and author filtering.

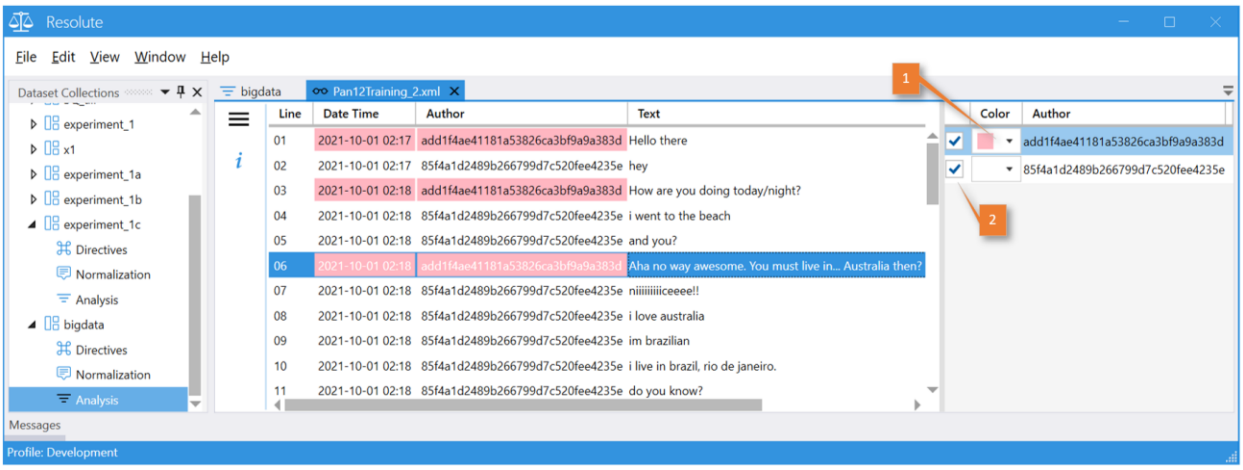

Figure 6-30: Chat Viewer

Figure 6-30 shows a chat in the viewer. The authors can be color coded (point #1) to make it easier to keep track of a predator or victim as the user is reading. Authors can also be filtered out of the chat (point #2) so that the reader can focus on the text for a particular author.

6.8 Experiments and Results

The following experiments consist of recording the time it takes to perform certain functions with a certain number of documents, i.e., the ability to handle large numbers and volumes of data in a performant manner.

All experiments were executed on a 4-core Intel i7-6700 @ 3.4Ghz, with 16G ram, running Windows 10.

The MongoDB database (i.e., dataset collection) that was used for the experiments contained over 1.5 million pre-imported documents before any of the experiments began (approx. 180G of data).

Dataset with 1,000 Documents	mm:ss
Import Documents	0:04
Convert to Text	0:09
Normalize	0:10
Initial Load	0:01
Open Document	0:01
SPI Analysis	0:12

Dataset with 5,000 Documents	mm:ss
Import Documents	0:09
Convert to Text	0:49
Normalize	0:49
Initial Load	0:02
Open Document	0:01
SPI Analysis	0:13

It is not recommended to create datasets greater than 5,000 documents:

- It becomes difficult to work with individual documents, for example scrolling the document list in a window becomes hyper-sensitive.
- Processing time for each operation increases non-linearly. This is because the Windows operating system imposes heavy performance penalties when accessing individual files in a folder with many files. For example, given a dataset with 10,000 documents, the conversion to text operation requires Resolute to export the original document to a temporary folder, the conversion program creates a converted text document bringing the number of files in the folder to 20,000, then Resolute needs to import each text file.

The following test cases show the time for datasets containing more than 5,000 documents.

Dataset with 10,000 Documents	mm:ss
Import Documents	1:13
Convert to Text	1:42
Normalize	1:17
Initial Load	0:03
Open Document	0:01
SPI Analysis	0:24

The PAN-2012 Training Dataset with 66,927 Documents	mm:ss
Import Documents	17:00
Convert to Text	20:00
Normalize	19:00
Initial Load	0:03
Open Document	0:01
SPI Analysis	4:53

The PAN-2012 Test Dataset with 155,128 Documents	mm:ss
Import Documents	74:00
Convert to Text	88:00
Normalize	95:00
Initial Load	0:04
Open Document	0:01
SPI Analysis	31:22

Given a corpus of more than 5,000 documents it is recommended that the documents be split across multiple datasets.

6.9 Future Work

Resolute could benefit from the following enhancements:

- Scan and generate normalization templates automatically. Refer to section *4.9 Future Work* for more information on how this can be accomplished.
- Build functionality that would enable Resolute to poll a directory for incoming documents. Whenever a new document is placed in the directory Resolute would automatically import, normalize and perform SPI analysis on the document. It would raise an alert and notify a person responsible when it discovers a predatory chat. This would enable the SQ to process in-coming chats in real-time. The SQ could get advance

notice of a possible meetup between predator and victim and protect the victim before the crime happens.

- The SPI scoring requires the user to perform some manual steps. Currently normalized documents need to be exported, then used as input to the SPI AI agent which runs outside of Resolute, and the scoring results imported back into resolute. Future work can integrate this directly into Resolute.
- There are a number of auxiliary functions that are not performant on large datasets. For instance, copy/paste of directives in the normalization window, or moving large datasets from one folder to another.
- There are a few long running functions that block the application. The application should never stop responding to user input. These functions need to be enhanced to not block a user from performing other functions while a long running operation is in progress.

Resolute allows multiple users to work on the same dataset collection; however, there is always the risk that two or more users might decide to work on the same dataset (CON2). When that happens, users risk overwriting each other's work. There are multiple ways of handling this, including building an observer that would be aware of any updates done to the dataset and updating any screens/windows working on that dataset in real-time. A more basic solution might be to lock the dataset so that only one user can work on it at a time.

The ability to normalize documents opens up doors for other types of AI agents. For instance, it would be useful to be able to identify authors who use one or more aliases, i.e., authorship identification. The PAN-2012 competition not only included SPI, but it also held an Author Identification competition [32]. With enough data and a good authorship identification AI agent, it would be possible to build and display complex social networks more accurately. This ability would greatly help law enforcement.

6.10 Concluding Remarks

We have shown that in support of hypothesis 2 (see *1.3 Research Objectives*) a chat management system can be defined and built that enables users to manage, process, analyze and identity raw chats for sexual predation more efficiently than it takes to perform the task manually.

Resolute attempts to bridge the gap between an AI agent that works well, and one that works well and is practical to use.

We wanted to provide the SQ with the ability to identify and prosecute suspected sexual predators quicker and with less effort than was done in the past (NFR1). A SPI AI agent alone cannot do that without help in managing and preparing the data first.

We showed that facilitating document normalization is possible, and that it provides significant benefits to the user.

The decision to re-write Resolute v1 was not an easy one to make. We had to abandon a lot of work, however; we felt that a non-performant application that is not productive to use will not serve its intended purpose. In the end we believe we made the correct decision, and v1 served as an excellent prototype for v2. Resolute v2 is a useable performant application.

Chapter 7: Conclusion and Future Research

Social media provides a powerful platform for individuals to communicate globally. This capability has many benefits, but it can also be used by malevolent individuals, i.e., predators. Anonymity exacerbates the problem. The motivation of our work is to help law enforcement protect our children from this potentially hostile environment, without excluding them from utilizing its benefits.

An AI agent by itself cannot provide the practical functionality needed by law enforcement to do an effective job at SPI. The sheer quantity of messy/noisy data make it a necessity to provide document management and normalization features. Our contribution deals with all these issues, and presents a practical effective solution to SPI.

The task of identifying conversations and authors engaged in sexual predation of minors is only the beginning. People adapt, and as one technology becomes effective at catching predators, they find other ways to locate and entrap their victims. Law enforcement is doing their best to protect us, but they have limited resources and need help. The work done in this thesis is our contribution to them. It is a worthwhile effort and one we hope other researchers will carry forward.

We have outlined in chapters 3, 4 and 5, some of the future work we think might provide the most value going forward. The following sections summarize what we think are the ones that can provide the most benefits:

7.1 Authorship Identification and Social Network Analysis

Social network analysis (SNA) is a powerful way of being able to visualize the social interactions of an individual. Predators are not only using social platforms in search of victims; they often use them to participate in support networks with other predators. They use these support networks to exchange information on how to entrap victims and how to avoid detection. Having a clear picture of a predators' social network can provide law enforcement very valuable information on protecting victims and identifying other predators.

There are some issues with SNA that make it less effective than it could otherwise be. People often use different user names on different platforms. This makes it tricky to obtain a full view of a predators' social network. A possible solution to this is an AI agent that can perform accurate identification of authorship. That is, a system that can map the likelihood that authors using different names are in fact the same author.

Enhancing Resolute to support SNA with accurate authorship identification would provide great value.

7.2 Meetup Detection in Chats

It would be useful for law enforcement to be able to detect when a predator and victim are planning to meet, before they meet. With this information law enforcement can prevent the crime before it happens, and they could build a solid case against the predator if they catch him/her red handed.

An AI agent that is capable of detecting in real-time a predator and victim meetup before they actually meet would provide great value to law enforcement. Research on this task has been done by *M. Vogt, et al.* [2] and *Pastor Lopez-Monroy et al.* [3] (see *2.2.1 Overview of the Top Approaches to SPI*). The general idea is for an AI agent that is capable of incorporating named entity recognition with SPI with the intention of a meetup.

Resolute can be enhanced to poll a directory for new files containing recently obtained chats. The chats would be picked up and processed in real-time. Automatic normalization of the chats would be applied, and the documents filed under designated datasets in Resolute. Resolute can set the groundwork for such an AI agent to be of practical use.

7.3 Automatic Template Creation for Normalization

Normalization of a chat requires that a corresponding template describing the chat header be supplied. Currently this needs to be done manually for each chat, and runs the risk of being time consuming and tedious. Resolute has mitigated that risk by supporting directives in which one template can be applied to a multitude of chats. However, it is still not the optimal solution.

It is possible to add functionality that can build the chat templates automatically. There are a few challenges to overcome (see section 4.9) but the benefits would be great. It would also be required if an AI agent that can detect meetups as described in the previous section is to be able to process chats in real-time.

7.4 Multilingual SPI

Using machine learning to give computers the ability to understand language requires large datasets. As part of the PAN2012 SPI competition the organizers created a dataset of English chats composed of 2,883,371 lines of labeled text. In their paper '*Overview of the International Sexual Predator Identification Competition at PAN-2012*' [14] they describe the challenges they faced putting the corpus together. Despite the challenges, they were able to find existing publicly available repositories of chat logs to base the normal conversations on.

Our experience trying to build a dataset of French based chats presented us with multiple challenges. The greatest challenge as it turned out was not the predator/victim chats (provided by the SQ), it was the large number of normal chats we needed to maintain the proportion of predator to normal chats as discussed in the '*Overview of the International Sexual Predator Identification Competition at PAN-2012*' paper. There are surprisingly few publicly available places where the chats can be downloaded. Not being able to find French based chats, we

attempted to scan several English repositories for French based chats. Surprisingly, a small proportion of the chats were in French, but not enough to build a proper dataset.

The idea of multilingual SPI is to use the existing English based PAN2012 dataset and leverage it to detect SPI in other languages. Can the machine translation capabilities of the Transformer be used in this task? There are online resources that enable one to build a decent machine-based language translation agent. These resources when used with a Transformer allow translation to and from many different languages. How can those resources be leveraged? These questions need exploration, and if an appropriate solution is discovered, the practical implications are great.

References

[1] "Online child sexual exploitation and abuse in Canada, 2014 to 2020," [Online]. Available: https://www150.statcan.gc.ca/n1/daily-quotidien/220512/dq220512a-eng.htm.

[2] Matthias Vogt, Ulf Leser, Alan Akbik, "Early Detection of Sexual Predators in Chats," *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing,* vol. 1, pp. 4985-4999, 2021.

[3] Adrian Pastor López-Monroy, Fabio A. González, Manuel Montes, Hugo Jair Escalante, Thamar Solorio, "Early text classification using multi-resolution concept representations," *Proceedings of the 2018 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers),* pp. 1216-1225, 2018.

[4] Mudasir Ahmad Wani, Nancy Agarwal, Patrick Bours, *Sexual-predator Detection System based on Social Behavior Biometric (SSB) Features,* 5th International Conference on AI in Computational Linguistics, 2021.

[5] D. Liu, "Identifying Cyber Predators by Using Sentiment Analysis and Recurrent Neural Networks," 2018.

[6] Fauzi, M.A., Bours, P., *Ensemble method for sexual predators identification in online chats,* 2020 8th International Workshop on Biometrics and Forensics (IWBF), 2020.

[7] E. Villatoro-Tello et al., "A two-step approach for effective detection of misbehaving users in chats," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[8] P. R. Borj, K. Raja and P. Bours, "On Preprocessing the Data for Improving Sexual Predator Detection," *2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA,* pp. 1-6, 2020.

[9] H. K. Patrick Bours, "Detection of Cyber Grooming in Online Conversation," *IEEE International Workshop on Information Forensics and Security (WIFS),* pp. 1-6, 2019.

[10] M. Ebrahimi, "Automatic Identification of Online Predators in Chat Logs by Anomaly Detection and Deep Learning," 2016.

[11] J. Parapar et al., "A learning-based approach for the identification of sexual predators in chat logs," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[12] C. Morris and G. Hirst, "Identifying sexual predators by svm classification with lexical and behavioral features," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[13] G. Eriksson and J. Karlgren, "Features for modelling characteristics of conversations," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[14] Crestani, Giacomo Inches and Fabio, "Overview of the International Sexual Predator Identification Competition at PAN-2012," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[15] "PAN is a series of scientific events and shared tasks on digital text forensics and stylometry," [Online]. Available: https://pan.webis.de/.

[16] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 2019.

[17] Mikolov, Tomas; et al., *Efficient Estimation of Word Representations in Vector Space,* arXiv:1301.3781.

[18] Jeffrey Pennington and Richard Socher and Christopher D. Manning, "GloVe: Global Vectors for Word Representation," 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/.

[19] F. Chollet, Deep Learning with Python (2nd ed), Manning Publications Co., 2021.

[20] Qiang Wang et al., *Learning Deep Transformer Models for Machine Translation,* arXiv:1906.01787.

[21] Ashish Vaswani et al., *Attention is all you need,* arXiv:1706.03762, 2017.

[22] Wikipedia, "GPT-3," [Online]. Available: https://en.wikipedia.org/wiki/GPT-3.

[23] "PAN - Sexual Predator Identification 2012," [Online]. Available: https://pan.webis.de/clef12/pan12-web/sexual-predator-identification.html.

[24] F. Chollet, "Understanding Word Embeddings," in *Deep Learning with Python (2nd ed)*, Manning Publications Co., 2021, p. 329.

[25] Y. Bengio et al., "A Neural Probabilistic Language Model," *Journal of Machine Learning Research,* vol. 3, p. 1137–1155, 2003.

[26] "Pandoc a universal document converter," [Online]. Available: https://pandoc.org/.

[27] "Apache Tika - a content analysis toolkit," [Online]. Available: https://tika.apache.org/.

[28] J. D. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Un-derstanding,* 2018.

[29] M. Popescu and C. Grozea, "Kernel methods and string kernels for authorship analysis," *CLEF (Online Working Notes/Labs/Workshop),* vol. 1178, 2012.

[30] M. Stonis, "The MVVM pattern," in *Enterprise Application Patterns using .NET MAUI*, Microsoft Developer Division, .NET, and Visual Studio product teams, 2022, pp. 9-20.

[31] M. Fowler, "Chapter 14. Web Presentation Patterns," in *Patterns of Enterprise Application Architecture*, Pearson Education, Inc., 2003.

[32] "PAN at CLEF 2012 Authorship Attribution," [Online]. Available: https://pan.webis.de/shared-tasks.html#authorship-attribution.

[33] "CLEF2012 Working Notes," 2012. [Online]. Available: http://ceur-ws.org/Vol-1178/.

[34] J. Sekeres, O. Ormandjieva, C. Suen and J. Hamel, "Advanced Data Preprocessing for Detecting Cybercrime in Text-based Online Interactions," in *Pattern Recognition and Artificial Intelligence - ICPRAI 2020*, Springer, 2020, pp. 416-424.