# Automatic Identification of Online Predators in Chat Logs by Anomaly Detection and Deep Learning

Mohammadreza Ebrahimi

A Thesis
In the Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science) at
Concordia University
Montreal, Quebec, Canada

April 2016

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Mohammadreza Ebrahimi

Entitled: Automatic Identification of Online Predators in Chat Logs by Anomaly Detection and Deep Learning

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:

Dr. Constantinos Constantinides Chair

Dr. Rene Witte Examiner

Dr. Leila Kosseim Examiner

Dr. Olga Ormandjieva Supervisor

Dr. Ching Y. Suen Co-supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____
Dean of Faculty

Date _____

# Abstract

Automatic Identification of Online Predators in Chat Logs by Anomaly Detection and Deep Learning

Mohammadreza Ebrahimi

Providing a safe environment for juveniles and children in online social networks is considered as a major factor in improving public safety. Due to the prevalence of the online conversations, mitigating the undesirable effects of juvenile abuse in cyberspace has become inevitable. Using automatic ways to address this kind of crime is challenging and demands efficient and scalable data mining techniques. The problem can be casted as a combination of textual preprocessing in data/text mining and binary classification in machine learning. This thesis proposes two machine learning approaches to deal with the following two issues in the domain of online predator identification: 1) The first problem is gathering a comprehensive set of negative training samples which is unrealistic due to the nature of the problem. This problem is addressed by applying an existing method for semi-supervised anomaly detection that allows the training process based on only one class label. The method was tested on two datasets; 2) The second issue is improving the performance of current binary classification methods in terms of classification accuracy and $F_1$-score. In this regard, we have customized a deep learning approach called Convolutional Neural Network to be used in this domain. Using this approach, we show that the classification performance ($F_1$-score) is improved by almost 1.7% compared to the classification method (Support Vector Machine). Two different datasets were used in the empirical experiments: PAN-2012 and SQ (Sûreté du Québec). The former is a large public dataset that has been used extensively in the literature and the latter is a small dataset collected from the Sûreté du Québec.

**Keywords:** Data Mining; Text Classification; Online Predator Identification; Chat Logs; Anomaly Detection; Deep Learning; Convolutional Neural Network; Vector Space Model; Naive Bayes; Neural Network; Support Vector Machines

# Acknowledgments

This research has been supported by the Natural Sciences and Engineering Research Council of Canada.
I would like to thank Dr. Ching Y. Suen and Dr. Olga Ormandjieva for their illuminating advice and compassionate guides during my academic journey.
I would like to thank sergeants Daniel Karneyeff-Bisson and Maxime Grenier in Sûreté du Québec for making the dataset available to this research and having illuminating discussions on the topic.

*To My Lovely Family,*
*Mohammad, Robab, and Siavash*

# Contributions of Authors

I hereby certify that all the materials in this thesis are entirely my own work and directly authored by me except about two pages of the "literature review" (sub sections 2.3.3 and 2.3.4) which are originally written by other authors in our co-authored in-press book chapter titled *"Automated Identification of Child Abuse in Chat Rooms by Using Data Mining"* in book "Data Mining Trends and Applications in Criminal Science and Investigations", IGI-Global publications, 2015.
In addition, all the artworks and tables that have not been referenced in their captions have been created by me.

Dated: March 1, 2016
Author: Mohammadreza Ebrahimi

# Table of Contents

# List of Figures

# List of Tables

# Terms and Abbreviations

**Age Disparity:** The significant difference between the age of a predator adult and that of a minor (i.e., victim)

**Conversation:** A chat session that encompasses the messages exchanged between participants.

**Clique:** In the context of graph theory, a clique is a set of vertices whose corresponding subgraph is complete (i.e., fully connected). In the context of mining criminal networks, this definition can be simplified to be less mathematical. Specifically, in the context of chat logs, a clique might be defined as a set of persons who participate in a minimum number of chat sessions.

**Luring Communication Theory:** A communication theory that models the behavior of predators for approaching, entrapping, and establishing a predatory relationship with a minor.

**Maximum Entropy Principle:** This principle states that the best probability distribution for a statistical model is the one that has the maximum entropy.

**Minor:** A Person under the age of 18 who is considered as the potential victim of predatory attack in cyber space.

**Vector Space Model:** The algebraic representation of documents based on their terms and the frequency of occurrence of each term.

**Anomaly:** Samples that do not conform to the underlying distribution or regular pattern of data.

**OPI:** Online Predator Identification

**SVM:** Support Vector Machine

**KNN:** K-Nearest Neighbor

**ConvNets/CNNs:** Convolutional Neural Networks

**NNs/ANNs:** Neural Networks/Artificial Neural Networks (original feed-forward Multi Layer Perceptron (MLP))

**DBNs**: Deep Belief Networks

# CHAPTER 1

## INTRODUCTION

The ease of access and anonymity of Internet users facilitate child exploitation and cyber sexual abuse. Due to the prevalence of the online conversations, mitigating the undesirable effects of juvenile abuse in cyber space has become critical. This has been a major concern in developed countries with a high rate of Internet access in which children are basically the most vulnerable Internet stakeholders. Providing a safe environment for juveniles and children in online social networks is considered one of the major factors in improving public safety. According to Kierkegaard (2008), sexual solicitations of 89% of youth are made in chat rooms. This highlights the vital need for mining large volumes of anonymous chat logs in order to address this kind of social crime.

Automated Online Predator Identification (OPI) is a proactive means to counteract the undesirable effects caused by the aforementioned crimes. Recently in the literature, this has also been known as Sexual Predator Identification (SPI) or Sexual Predator Detection (SPD). Although practical OPI involves dealing with textual data and images, textual data are considerably more convenient to be used for automation purposes rather than the imagery data. Accordingly, dealing with textual data is the main focus of this thesis, wherever the OPI is mentioned in general.

This chapter serves as a basis of problem understanding for the rest of chapters of the thesis. It highlights the importance of Online Predator Identification (OPI) as an effective action toward improving public safety in society. This is also known as Sexual Predator Identification (SPI) or Sexual Predator Detection (SPD). We will address the problem as OPI in the rest of this writing. This section also discusses the relationship between machine learning and OPI which is the focus of next chapters.

## 1.1. Domain Concepts

This section contains the essential information about legal and psychological aspects of online predator identification.

### 1.1.1. Legal Aspects

Although legislative and regulatory provisions regarding online child sexual abuse aim to combat and mitigate the impact of this threat, they may vary in different countries or even different jurisdictions in the same country. According to Kierkegaard (2008), "*while virtual child porn using avatars is generally considered illegal in the European Union, it might not necessarily be treated as such in the United States"* (p.44). Similarly, "*images that are illegal to view in the USA may not be illegal to view in Germany"* (p. 41). The same situation exists about the concept of age disparity between adults and minors. There have been countless writings on the legal aspects of child sexual abuse in online environments that go beyond the scope of this chapter.

### 1.1.2. Psychological Aspects

The most effective and also simple psychological aspects of predatorhood might be those defined by Morris in his master of science thesis (Morris, 2013). The author defined predatorhood as having two major components: *age disparity* and *inappropriate intimacy*. The former relates to the psychological immaturity of the victim compared to that of predator (adult) which may differ in various countries by law. The latter corresponds to the attempt of adult to establish an intimate conversation that usually involves implicit or explicit sexual comments.

One of the most practical psychological theories which is widely used in online predator identification is known as luring communication theory (Olson, Daggs, Ellevold, & Rogers, 2007). The theory comprises three main phases needed for committing a predatory act:

1. Gaining access to the victim
2. Entrapping and grooming until the victim accepts sexual advances
3. Initiating and maintaining the abusive relationship

On the Internet, the most common way for gaining access is through online conversations in chat rooms.

The second stage can be distinguished by observation of the predator's attempt to desensitize the child to the inappropriate intimacy.

Finally, the third step involves explicit sexual exploitation of the minor. At this point, a reliable OPI system can flag the conversation for the attention of law enforcement in order to prevent the predator from approaching the victim.

## 1.2. OPI Problem Definition

During the past decade, automated Online Predator Identification (OPI) has become tractable by using text mining algorithms. These algorithms are capable of identifying likely predators for the attention of law enforcement. Using automatic ways to address this kind of crime is challenging and demands efficient and scalable data mining techniques which are able to handle large volumes of chat logs. There are two major OPI problems in which text mining plays an important role:

1. Detecting predators
2. Visualizing and analyzing predator criminal networks

The solution to the first problem, which is the main focus of this thesis, can be casted as a combination of textual preprocessing in data/text mining and pattern classification in machine learning and also criminal psychology (Figure 1). The solution to the second OPI problem is provided by extracting the underlying relationships and using graph mining techniques to analyze the resultant social networks.

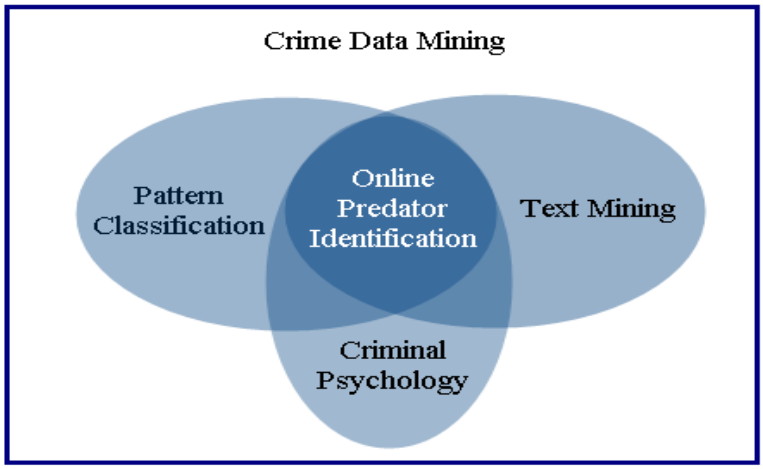


*Figure 1. Relationship between OPI, Text mining, Pattern Classification and Criminal Psychology*

As illustrated in Figure 1, text mining and pattern classification techniques form the algorithmic foundations of OPI. A comprehensive but concise survey of text mining which incorporates the gist of text mining algorithms is provided in (Aggarwal, 2015). Pattern classification (also known as pattern recognition) encompasses the classification algorithms used in predator identification and goes hand in hand with machine learning techniques (Duda, Hart, & Stork, 2012).
Figure 2 illustrates the main data mining techniques used in OPI along with their relationships.

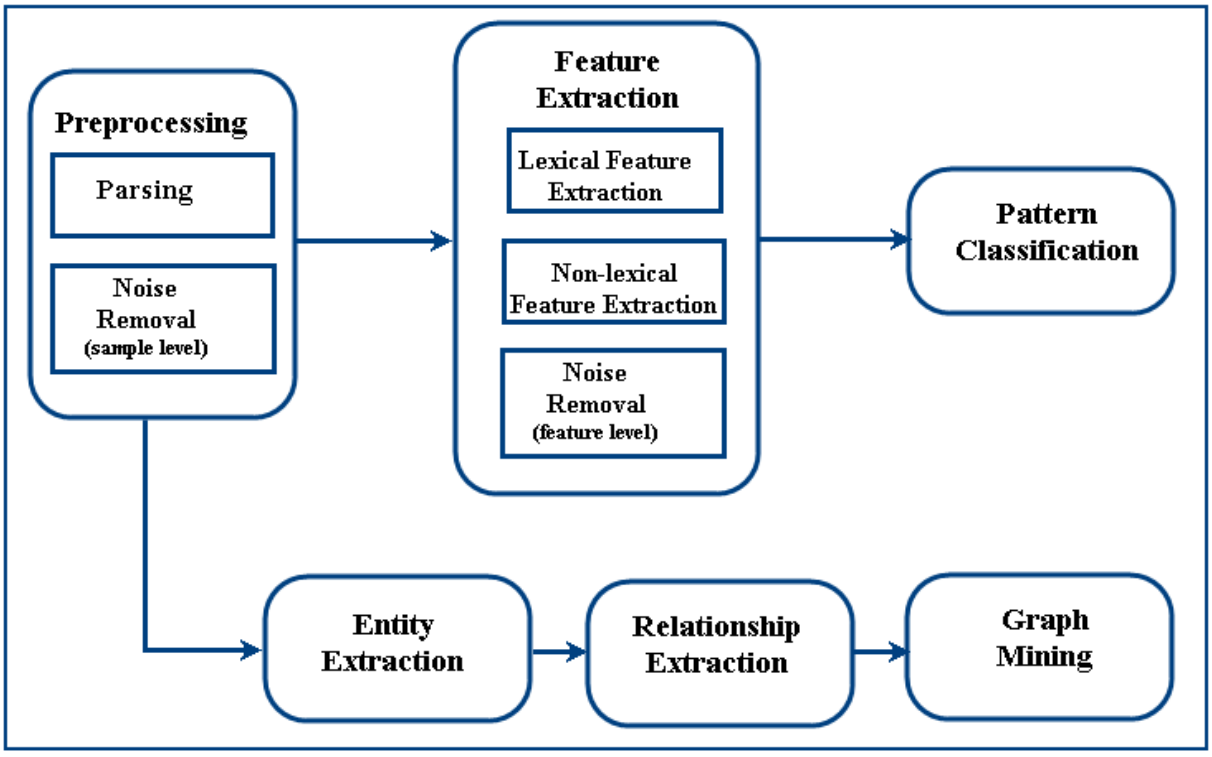


*Figure 2. Major data mining techniques used in OPI and their interdependencies*

The first module is responsible for performing preprocessing tasks on the input text. The constituent of this module depends on the format of the data. Since in our domain we are mostly dealing with XML files, using an XML parser is necessary. After parsing, some documents may be chosen to be eliminated from further processing.
In the feature extraction module, prominent lexical features are extracted. Additionally, some behavioral and structural features are extracted based on the statistical and linguistic characteristics of the conversation. Afterwards, the low-quality features might be removed. The quality of a feature is usually defined to be proportional to the amount of contribution of the feature on the output (see Section 2-3). The *pattern classification* module performs the classification task and produces the final output (see Section 2-4).
There has been a dedicated competition for Sexual Predator Identification in PAN-2012 as part of the CLEF 2012 competition[1] that expedited the movement of applying data mining techniques on chat logs in order to identify the likely predators. Several competitors from all over the world applied their data mining techniques on a relatively large volume of chat logs. The competition encompasses the following two tasks (Inches & Crestani, 2012):

1. Distinguishing the predators and victims.
2. Specifying predatory messages in predatory conversation.

Accomplishing the first task is of greater help to law enforcement in terms of narrowing down their search space significantly. According to Villatoro-Tello et al. (2012), this task can be performed in two consecutive steps:

1. Identifying the predatory conversations among all conversations.
2. Distinguishing the sexual predator and the victim among participants of predatory conversations.

This thesis proposes an abstract taxonomy that encompasses different classification techniques that are used in the OPI field. There are three main granularity levels of analysis in dealing with online predator identification. These levels are shown in Figure 3.

[1] http://pan.webis.de/

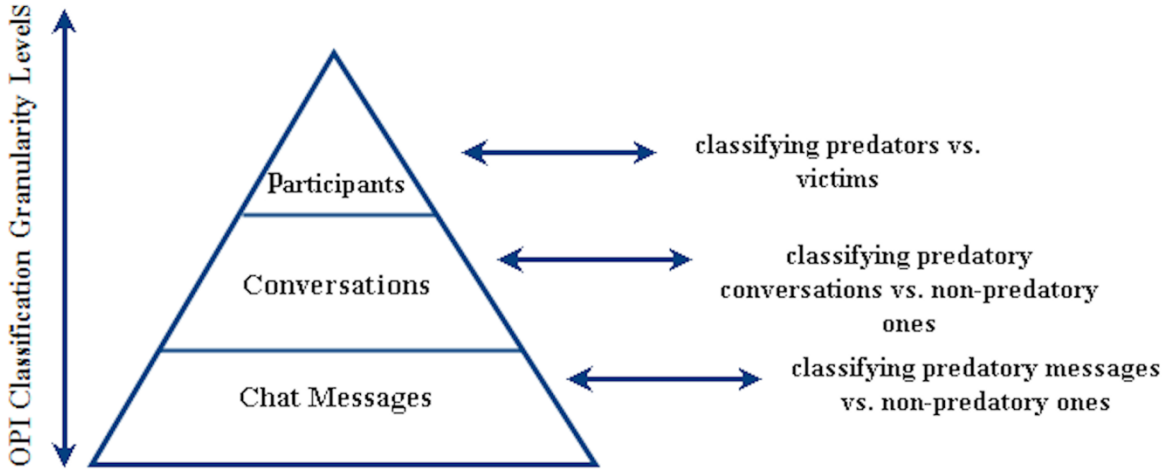


*Figure 3. Classification Granularity Levels and their corresponding classification problem in OPI (Ebrahimi, Suen et al., 2016)*

As seen in Figure 3, the most fine-grained level corresponds to direct analysis of *messages* (also known as *interventions*) exchanged by participants in their conversation. This kind of analysis corresponds to the task of distinguishing the sexual predators in PAN-2012. The next level of granularity relates to the task of identifying predatory online conversations. Finally, distinguishing the predators among all of the participants in the entire corpus can be considered as the highest level of abstraction that might be the ultimate goal. All of these analyses are accomplished by utilizing the proper machine learning classification techniques. For a more cohesive and detailed introduction of these techniques, please refer to (Keyvanpour, Ebrahimi, et al. 2015) or the second chapter of this thesis.

## 1.3. Objective and Contribution

This thesis aims to investigate and address two major problems in the domain of OPI and adapt machine learning techniques to address these problems. The objectives can be categorized as follows:

***Objective 1: Eliminating the problem of gathering negative training instances while keeping the performance acceptable***

In practice, finding enough negative (non-predatory) instances is tedious and sometimes unrealistic. This thesis proposes a semi-supervised anomaly detection approach that only utilizes one of the class labels in the training process rather than both labels.

***Objective 2: Improving the classification performance in terms of $F_1$-score compared to traditional machine learning algorithms (SVM and ANN)***

Binary classification methods that have been used in the OPI domain have a relatively lower classification performance than that of the deep learning method that is proposed in this thesis.

## 1.4. Research Questions and Hypotheses

Aligned with the two main problems mentioned in the preceding section, the followings are main research questions:

***1)*** *Considering the OPI domain, how efficient will an anomaly detection model that only uses one of the class labels perform, as compared to binary classification models that use both positive and negative class labels?*

***Hypothesis 1:*** Predatory/Non-predatory conversations can be represented as anomalous conversations that do not conform to the underlying data distribution. The problem can be casted to a *one-class* classification problem. The performance would be comparable to that of binary classification which uses two class labels.

***2)*** *Is a deep learning architecture able to increase the classification performance and outperform the current state-of-the-art performance obtained in this domain?*

***Hypothesis 2:*** Learning deep architectures for classification of chat logs can outperform the state-of-the-art methods in terms of $F_1$-score.

## 1.5. Research Methodology

First, a comprehensive study was conducted on the literature and the extension points and problems are identified. Accordingly, this thesis stated two main hypotheses that are mainly related to alleviate current problems in the domain. Then, datasets are obtained and pre-processed. Afterwards, several empirical sets of experiments are designed to support or to reject each hypothesis. In parallel, a software prototype is designed and implemented in Java to support hypothesis 1. Finally, the results of the experiments are articulated and the related challenges are discussed. Additionally, a speculation of future research direction is presented based on the observations.

## 1.6. Structure of this Dissertation

To answer the research questions and evaluating the stated hypotheses this dissertation has been organized as follows:

Chapter 2 contains a literature review covering preprocessing, feature extraction and classification techniques used in the OPI domain. It contains the background information on anomaly detection and deep learning required to understand the content of the remaining chapters.

Chapter 3 is dedicated to the usage of anomaly detection in our problem at hand. It describes the required adaptations and customizations; it also contains the description of the corresponding experiments and the analysis of the results. The goal of this chapter is to support *hypothesis 1* stated above.

Chapter 4 covers the experiments and results of applying and customizing deep learning techniques on the domain of OPI. The goal of this chapter is to support *hypothesis 2* stated above. Chapter 5 covers software engineering aspects of the prototype that was designed and

implemented as a proof of concept for supporting hypothesis 1. Chapter 6 has been devoted to draw conclusions, revisiting the objectives, and corresponding hypotheses, and the research schedule. Finally, Appendices A and B illustrate some samples of XML processes used in Chapter 3 and also several samples of bash scripts used in Chapter 4.

# CHAPTER 2

# BACKGROUND AND LITERATURE REVIEW

This chapter covers the related work regarding the whole contribution of this thesis. It starts with describing general usage of machine learning in OPI and then moves along to the background review of anomaly detection techniques. Finally, a literature review of deep learning is discussed. The chapter describes the usage of preprocessing, feature extraction, and classification in various aspects of an OPI problem. Different methods of preprocessing that have been found to be useful for working on chat logs are discussed. These methods include the most popular data formats, noise removal procedure and dimensionality reduction. Then, feature extraction and the standard ways of feature enrichment in this application domain are described.

The chapter also covers the usage of sentiment features as a complimentary set of features that can improve the performance of classification. In addition, a brief basic tutorial about the classification algorithms that are used in the domain of automated predator identification is presented. These algorithms cover a wide range of classification algorithms, such as entropy-based classification, Naïve Bayes, Support Vector Machine, and Neural Networks. We have tried to refer the keen reader to the related resources about the fully-detailed theories behind these algorithms.

Finally, in the last part of the chapter we describe Social Network Analysis (SNA) as another area related to this field of study. We provide a high-level introduction to the usage of SNA and its relationship to the online predator identification.

## 2.1. Online Predator Identification

As mentioned in the previous section, OPI has its root in text mining and pattern classification. With the rapid increase of available textual data in different domains including news, social media, and web pages, text mining has drawn the attention of researchers during the last decade. There has been a variety of algorithms and approaches including text clustering and classification, text summarization, topic modeling, and opinion mining. Opinion mining or sentiment analysis is an important discipline in Natural Language Processing (NLP) that extracts people's opinion, attitudes and emotions toward entities, other people, events, and their attributes. The reader may refer to (Aggarwal, 2015; Irfan et al., 2015; Liu & Zhang, 2012) for comprehensive explanations of these algorithms and approaches. Here, we narrow down the focus of the chapter to the usage of these techniques in OPI and the related background.

One of the very first successful attempts for applying data mining to OPI problem was accomplished by Pendar (2007) who used a weighted K-NN classifier to distinguish predators from underage victims. In addition, the first empirical system with the capability of determining predatory messages in chat logs is ChatCoder1 (Kontostathis, 2009). Afterwards, ChatCoder2 (see Section 2) was developed on top of the previous version to improve its performance (Mcghee et al., 2011). The system used a rule based approach in conjunction with decision trees and instance-based learning methods (K-NN).

Michalopoulos and Mavridis (2011), and Escalante et al. (2013) have utilized Luring Communication Theory (described in Section 1-1-2), to combine psychological aspects of predation phenomenon with computer science and machine learning.

Recently, the PAN-2012 conference has acted as a boost for applying machine learning techniques to this area. Several machine learning algorithms have been used to address the OPI problem in this competition. These algorithms cover a wide range of classification algorithms such as Entropy-based Classification (Eriksson & Karlgren, 2012), K-Nearest Neighbor (Kang et al., 2012), Support Vector Machine (Morris, 2013) and Neural Networks (Villatoro-Tello et al., 2012). The team with the best performance in terms of $F_1$-score (Villatoro-Tello et al., 2012), used a two-step binary classification approach called SCI (Suspicious Conversation Identification) and VFP (Victim From Predator Disclosure) using SVM and Neural Networks.

Escalante et al. (2013) proposed a novel method using chained-classifiers based on adapting a psychological hypothesis that underscores three stages employed by predators to approach the victim. Although this method could not outperform the approach used by Villatoro-Tello et al., it revealed that adopting psycho-linguistic hypotheses could improve the accuracy.

Due to the inadequacy of bag-of-words models in reflecting deep semantic notions hidden in the conversations, Bogdanova et al. (2012b) tried to enrich the features by introducing sentiments and emotions to the original feature set. In another research (Bogdanova, Rosso, & Solorio, 2014), the authors improved their feature set by adding more high level features such as neuroticism and psychological aspects. We will cover these studies in more detail through the chapter.

The use of deeper linguistic features in this field were attempted by Forsyth & Martell (2007) in their research for creating an annotated chat corpus with both lexical and semantic tags to facilitate the application of data mining in this domain. As another linguistic analysis example, Bogdanova et al. (2012a) have worked on identifying fixated discourse on chat logs. Fixated discourse signifies on the strong intention of the predator to keep the focus of the conversation on sexual topics. Finally, a holistic approach has been presented by Cano et al. (2014) based on leveraging lexical features, sentiment features, content and psycho-linguistic features, and discourse patterns. They have used semantic frames, which incorporate the general aspects of a discourse, and added them as additional features to the original bag-of-word model.

In the remainder of the chapter, we discuss the methods mentioned above in greater detail and highlight their strengths and weaknesses. Table 1 shows a mapping between data mining techniques and their applications in predator identification problem as well as the reference to corresponding works that have been done in the domain.

*Table 1. Mapping of applications in OPI and corresponding data mining techniques*

| Application Area in OPI | Data Mining Technique(s) | Previous works |
|---|---|---|
| Predator Detection | Binary Classification | • Approaches in PAN-2012<br>• (Morris, 2013)<br>• (Cano et al., 2014)<br>• (Pendar, 2007) |
| | Latent Semantic Indexing | • (Kontostathis et al, 2013) |
| | Rule-based Approach | • (Mcghee et al., 2011) |
| | Anomaly Detection | • (Ebrahimi et al., 2016) |
| Criminal Network Analysis | Graph Mining | • (Iqbal et al., 2012) |
| Criminal Network Visualization | | |

### 2.1.1. Criminal Network Analysis and Visualization

Another aspect of OPI deals with pedophile covert network analysis and visualization. Iqbal et al. (2012) have used the concept of criminal clique mining on chat logs to reveal the hidden relationship among criminals. Figure 4 depicts the main components of their framework.

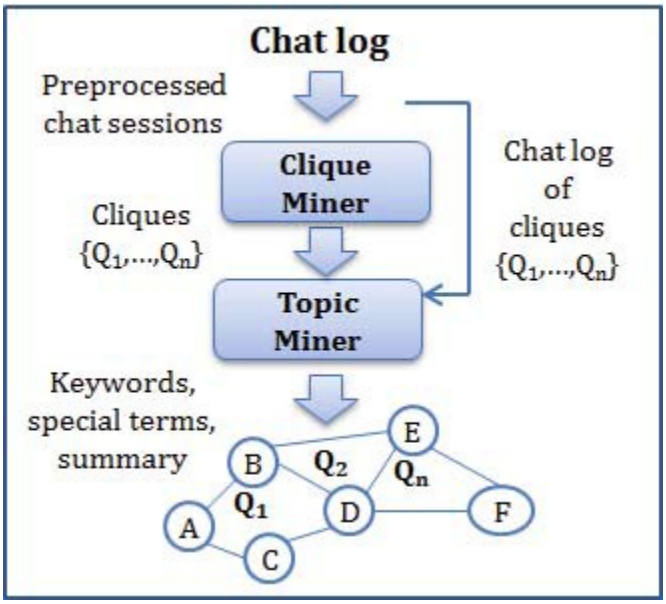


*Figure 4. Overview of the framework for mining criminal networks in chat logs (Iqbal et al., 2012)*

To analyze a criminal social network we need to explore the communication structure and the patterns by which network communications evolve. Based on the work of Klerks (2003), criminal network investigation approaches can be categorized into three types: 1) manual approach, 2) graphic-based approach, and 3) social network analysis. Since the focus of this chapter is on the identification aspects rather than network visualization and analysis, we focus on the predator detection through the remaining sections.

### 2.1.2. Successful Sample Tools

In order to introduce successful examples of automated online predator identification we briefly describe two software tools in which data mining techniques have been applied to this domain in a practical environment.

- **ChatCoder 2:** This tool was implemented at the Mathematics and Computer Science Department of Ursinus College in Pennsylvania in 2011 (Mcghee et al., 2011). The software uses a rule-based approach to classify the messages in forums into several categories including 'exchange of personal information', 'grooming', 'approach', and 'none of the categories'. These categories have been chosen based on the different phases defined in communication theory described in Section 1.2.2. This software system could provide a 68% accuracy on a public dataset available by a non-profit organization called perverted justice. One of the success factors of the software is that the system outperformed the human labeling in some of the categories.
- **Child Exploitation Tracking System (CETS):** In 2003, Microsoft initiated a new tool called Child Exploitation Tracking System (CETS) through close collaboration with the Toronto Police Service to create an infrastructure for sharing the relevant documents and evidences among different investigators. According to the Royal Canadian Mounted Police (RCMP), the tool is still being extensively used in Canada as a cross-jurisdiction information sharing system between child exploitation investigators (Toews, 2013). CETS can be construed as a reliable tool that is being effectively utilized for combating child exploitation.

## 2.2. Preprocessing methods for chat logs

Many data mining processes require a domain specific data preprocessing task which is often a tedious and time consuming task. Preprocessing of chat logs includes different subtasks ranging from parsing the raw textual log files to removing noise and reducing dimensionality.

### 2.2.1. Chat Log's Data Format

Since analyzing textual data is less expensive and more efficient than analyzing other media such as image and video, currently almost all of the approaches for OPI use solely textual data. Log's format strongly depends on the software tool that is used for logging the conversations on the chat server. Although it can be any log format depending on the logging tool, usually the raw chat log data is gathered in a semi-structured textual format such as XML or JSON file. Various items can be stored in these logs among which the following three elements are essential for OPI analysis:

- **Authors:** participants who are usually identified by a unique identifier
- **Message text:** the textual transferred message
- **Time stamps:** the date and time corresponding to each exchanged message

Figure 5 depicts a sample template for a chat log in XML format.

```
<Log date=May 05,2015>
    <message id=1>
            <participant id= Q12445678D></participant>
            <text> hi </text>
            <time>May 05,2015-16:45:20</time>
    </message>
    <message id=2>
            <participant id= Q12445678D></participant>
            <text> r u there? </text>
            <time>May 05,2015-16:45:26</time>
    </message>
    <message id=3>
            <participant id= F11445E211></participant>
            <text> hi </text>
            <time>May 05,2015-16:45:20</time>
    </message>
</Log>
```

*Figure 5. A simple chat log in XML format with essential items for OPI*

### 2.2.2. Noise Removal

Noise removal procedure is done for the purpose of improving the performance of learning (in terms of precision, recall, and accuracy) and also reducing the training time. In OPI problems, noise removal procedure falls into one of the following categories (or a combination of both):

**Removing noisy conversations:** This category of noise removal procedure includes identifying and eliminating useless samples that do not affect the learning process. This includes removing the following items:

- *Non-textual samples:* Real-world chat logs may contain conversations with only non-textual data or a very tiny amount of textual information. These samples can be safely ignored.
- *Conversations which include only one participant:* This kind of conversations usually exists in a chat log corpus due to the fact that a participant may be unsuccessful in having a conversation with another participant.
- *Extremely short messages* (e.g., those which only contain a short greeting between two or more participants)

**Removing noisy features:** This category includes removal procedures for eliminating noise from features obtained during the feature extraction procedure (see Section 4).

Feature extraction and its corresponding methods in OPI will be discussed in the next section. For now, features can be considered as the set of important terms in a conversation.

Noisy features may include the followings:

- *Terms which are not in the proper encoding:* This happens especially when there are multiple languages involved in the training corpus or there are other encodings than Unicode Transformation Format (UTF).
- *Small images or emoticons transferred among a whole bunch of textual conversations are construed as noisy features depending on the approach:* It should be noted that emoticons are considered as a valuable source of information especially for extracting sentiment features. In such cases, emoticons should not be treated as noise.
- *Unintentional misspelled words throughout the conversation:* It is worth mentioning that *intentional* misspelled words often play an important role in this application domain. As Villatoro-Tello et al. (2012) state: *For example in the grooming phase the perpetrator may amend the relationship by an emphasized "soryyyyyyyyy" when the child felt threatening by any obtrusive language* (p. 4).On the other hand, differentiation of intentional from unintentional spelling errors is not an easy task. Therefore, some researchers avoid the entire spell checking in the hope of gaining quality improvement.

### 2.2.3. Feature Selection and Dimensionality Reduction

Let $\boldsymbol{D}$ be a dataset with set of *n* terms denoted by *N*, and also let $f$ be the function which maps the conversations into $l \in \{p, np\}$ in which *P* and *NP* represent the predatory instances and non-predatory instances respectively. The feature selection can be defined as the process of finding $N^* \subseteq N$ so that the performance of classifier $f$ is maximized. The performance of the classifier is typically defined by accuracy, precision, recall and $F_1$-score. Forman describes a holistic introduction of feature selection techniques used in text classification (Forman, 2003).

Two common feature selection techniques are widely used in the OPI domain. One of them falls into the category of supervised feature selection techniques and the other one is an unsupervised technique:

- ***Unsupervised feature selection:***

*Document Frequency Thresholding:* Let *d(t)* be the number of documents in which term *t* occurs. Subset $N^*$ contains *t* if and only if $d(t) \geq l$, in which $l \in \mathbb{N}$ is an arbitrary threshold. As Yang and Pedersen (1997) state, document frequency thresholding is the simplest technique which scales well to large corpora.

- ***Supervised feature selection:***

*Information Gain:* Supervised feature selection techniques measure each feature based on its contribution to the identification of the correct category (i.e., predatory or non-predatory). Information gain is a common supervised technique used for feature selection in text classification in the domain of OPI. The notation used by Forman (2003) for calculating information gain in a binary classification problem suits well in OPI. According to this notation, information gain for a specific term (feature) is defined as:

$$IG(t) = e(pos, neg) - [P(t)e(tp, fp) + P'(t)e(fn, tn)] \quad (1)$$

where:

$$e(x,y) = -\frac{x}{x+y}log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y}log_2\left(\frac{y}{x+y}\right) \quad (2)$$

In addition, *pos* and *neg* represent the number of predatory and non-predatory cases, respectively. Also *tp* represents the number of predatory cases containing the term, *fp* is the number of non-predatory cases containing the term, *fn* is the number of predatory cases not containing the term and *tn* represents the number of negative cases not containing the term. *P(t)* is calculated as follows:

$$p(t) = \frac{no.of cases\ containing\ term\ t}{total\ no.of cases};\ p'(t) = 1 - p(t) \quad (3)$$

There is a large variety of dimensionality reduction techniques used in data mining, but there is a text-specific dimensionality reduction technique called stemming, which is widely used in text mining. The purpose of doing this preprocessing step is to reduce, for instance, the terms '*work*', '*works'*, '*worker'* and *'working'* into one dimension as *'work'*. This process usually has a desirable effect on the performance of text categorization, both in terms of quality and time efficiency.

However, unfortunately this technique may not provide the same desirable effect for the OPI problem domain due to the fact that it will distort the information pertaining to the writing style of predators in chat logs (Villatoro-Tello et al., 2012). Accordingly, the best results have been reported by other researchers in PAN-2012, while stemming has been avoided.

## 2.3. Feature Extraction

Table 2 categorizes the features that have been used in mining OPI problems along with corresponding previous works that have utilized these features.

*Table 2. Categorization of features used in OPI problem*

| **Feature Category** | **Description** | **Previous works** |
|---|---|---|
| Lexical Features | Bag-of-word representations including:<br>• Unigrams<br>• Bigrams<br>• Trigrams | • (Villatoro-Tello et al., 2012)<br>• (Morris, 2013)<br>• (Pendar, 2007) |
| Behavioral Features | • The number of times this author initiates a conversation<br>• The number of times the author asks a question<br>• Response Time<br>• Conversation Dominance<br>• Number of turn-takings | • (Morris, 2013) |
| Psychological and Linguistic Features | • Fixated Discourse (see below)<br>• Writing Style (see below)<br>• Emoticons (see below)<br>• Tendency to change conversation to sexual discourse<br>• Awareness of doing an illegal and non-moral action that may cause prosecution<br>• Mimicking children language | • (Bogdanova et al., 2012a)<br>• (Mcghee et al., 2011)<br>• (Hogenboom et al., 2013) |
| Sentiment-oriented Features | • Fear, Anger, Anticipation, Joy, Sadness, Disgust, Surprise, etc. | • (Bogdanova et al., 2012b) |

In the rest of this section, the above feature extraction categories are described in greater detail.

### 2.3.1. Lexical Features

Lexical features are word-related features that are directly extracted from the sentence. These words are used as candidate features in classification algorithms in order to determine the category to which chat logs belong.

One of the simplest approaches used in the OPI problem to extract features is the *bag-of-words approach* that treats a chat log as a bag of words. Each word that exists in a chat log is considered as a candidate feature and then these features are weighted in terms of their frequency of occurrence. Typically, the initial candidate feature set is built by extracting *n*-grams (unigrams, bigrams, and trigrams) from the training data. Usually words that have weak lexical meaning are known as stop words and are filtered out during the pre-processing in standard text categorization and Information Retrieval (IR) studies. However, because of the fact that chat logs have an informal writing style, the general list of stop words may not be adequate. That is why some researchers create their own list of stop words. As an example, the stop word list used by Pendar (2007) contained specifically 79 most frequent

words in the corpus. After filtering out the entire stop words successfully, the *n*-grams and their corresponding frequencies for each chat log (or for each chat participant) are extracted.
Standard bag-of-words model has been shown to be robust in a wide variety of text classification problems. *Term Frequency-Inverse Document Frequency (TFIDF)* is the most common weighting approach that is extensively used for weighting the candidate features before performing any feature selection procedure. In this weighting scheme, the most important words tend to have higher weights. This is implicitly achieved by multiplying the frequency of term *t* by a magnitude that is inversely proportional to the occurrence of term *t* in the whole corpus.
Generally speaking, building unigrams and bigrams (pairs of consecutive words) produce better results than higher *n*-grams. Depending on the characteristics and also the size of the training corpus, the use of bigrams may increase the performance at the expense of increasing the size of the feature-space. However, for a training problem it may turn out that unigram model produces a better performance.
Regardless of the classification performance, the size of the feature set for bigram representation is typically much larger than that of unigram model. The resultant feature set can also be enriched by adding *domain-specific* features. As an instance, in the work of Morris (2013), special tokens such as \SMILEY, \MALE.name, \FEMALE.name, \NUM and \PHONE.name were added to the lexical features in order to enrich the initial feature set. However, it was mentioned that these enhancements seemed to add an insignificant improvement.
Considering the inadequacy of bag-of-words models in reflecting deep semantic notions hidden in the conversations, one can also enrich the feature set with behavioral features which would be explained in detail below.

### 2.3.2. Behavioral Features

In this section, we list the high-level behavioral features and their applicability in the detection of online predators. Behavioral features are characterized as features that capture the typical actions of a user within a conversation. Morris (2013) has classified behavioral features as: *'Initiative'*, *'Attentiveness'* and *'Conversation dominance'* for which the details are given accordingly:

- **Initiative:** This can be measured by *number of initiations* (i.e., number of times a specified participant starts the conversation), *initiation rate* (i.e., the ratio of number of initiations to the whole number of conversations), questions and question rate in order to understand the author's tendency during the conversation.
- **Attentiveness:** This feature corresponds to the mean, median, and max response times for each author.
- **Conversation dominance:** A set of features such as 'Message Ratio', 'Word Count Ratio' that reflect the degree to which the focal author dominates the conversation.

In order to successfully distinguish predators from victims, the above mentioned features are critical for *'symmetry-breaking'* (Morris, 2013). That is, given the fact that two authors in a chat conversation use very similar languages, behavioral features are one of the significant identifiers or non-lexical aspects of the conversation which are able to differentiate predators from victims.

### 2.3.3. Psychological and Linguistic Features

Psycho-linguistic features form another important aspect of feature extraction in OPI domain. *'Fixated discourse'* is one of the most prominent psychological features used in OPI. Bogdanova et al. (2012a) defined fixated discourse as the unwillingness of the predator to change the topic. For instance, predators often ignore questions or interruptions of pseudo-victims and have the tendency to go back to a sex-related conversation. According to Bogdanova et al. (2012a), chat logs might include implicit and explicit sexual content as predators gradually alter the direction of conversation to sex by starting with some ordinary compliments. On the other hand, predators often are aware of the fact that what they do on chat rooms is not moral and they try to transfer the responsibility to the victim and often behave as children by copying the children's linguistic style (Bogdanova et al., 2014).
The following analysis of chat logs identifies the important characteristics of predators' psychological and linguistic features (Bogdanova et al., 2014)

- *"Implicit/explicit content: Typically, pedophiles shift gradually to the sexual conversation, starting with ordinary compliments and then they shift the conversation to make it overtly related to sex. They do not hide their intentions.*
- *Fixated discourse: Pedophiles are reluctant to step aside from the sexual conversation. In other words, pedophiles try to come back to the sex-related conversation when the victim steps outside of the topic.*
- *Offenders often understand that what they are doing is not moral*.
- *They transfer responsibility to the victim*.
- *Offenders often behave as children, copying their linguistic style. Colloquialisms appear often in their messages."*

Mcghee et al. (2011) have proposed the following linguistic features which were denoted as 'Writing Style' in Table 2: Total number of words in a line, number of first-person pronouns, second-person pronouns or third-person pronouns, number of personal information nouns *(e.g., age, pic)*, number of relationship nouns *(e.g., boyfriend, date)*, number of family nouns *(e.g., mom, sibling)*, number of communicative desensitization words *(e.g., kiss, bra)*, number of approach verbs *(e.g., meet, see, hotel)*.
Hogenboom et al. (2013) indicate that people were influenced by nonverbal cues and emoticons. These are widely used to express sentiments such as happiness, sadness, joy or anger, therefore emoticons could also be used to reveal the predators' sentiments and their tendencies in order to be dominant in the conversation and also to copy children's' behavior as explained above. The following table shows the typical examples of emoticons and their sentimental interpretations (Hogenboom et al., 2013).

*Table 3.Typical examples of emoticon synsets (Hogenboom et al., 2013)*

| Emoticon synset | Emoticons |
|---|---|
| Happiness | :-D, =D, xD, (^_^) |
| Sadness | :-(, =( |
| Crying | :'(, ='(, (;_;) |
| Boredom | -_-, -.-, (>_<) |
| Love | <3, (L) |
| Embarrassment | :-$, =$, >///< |

### 2.3.4. Sentiment-oriented Features

In addition to the emoticons explained earlier, the sentiment of chat logs can provide significant markers in terms of predator identification and unveil other important semantic dimensions. According to Bogdanova et al., (2012b), in general, predatory conversations contain more positive and less negative words.

The following sentiments and emotional markers were used as features in their experiments:

*Table 4. Sentiment Features (Bogdanova et al., 2014)*

| Feature | Example |
|---|---|
| Positive words | Cute, pretty |
| Negative words | Dangerous, annoying |
| JOY words | Happy, cheer |
| SADNESS words | Bored, sad |
| ANGER words | Annoying, furious |
| SURPRISE words | Astonished, wonder |
| DISGUST words | Yucky, nausea |
| FEAR words | Scared, panic |

## 2.4. Learning Predatory Patterns

First, we formally introduce the notion of binary classification which is used in the OPI problem.

Let dataset D be defined as $\boldsymbol{D} \subset \boldsymbol{X} \times \boldsymbol{Y}$ where $X = \{\chi_1, \chi_2, \dots, \chi_m\}$ is the set of $m$ observation vectors $\chi_i;\ i \in \{1,2,\dots,m\}$, so that $\chi_i = (x_1, x_2, \dots, x_n)^T$ is the corresponding vector of $i^{th}$ observation containing $n$ feature values $x_j; j \in \{1, \dots, n\}$. Also $Y = \{p, np\}$ is the set of two class labels corresponding to predatory and non-predatory instances respectively. The classification task is defined as finding a mapping function $f : X \to Y$ such that $f(\chi)$ is able to predict $y \in Y$ as accurately as possible.

As a typical approach, the data is split into training and testing sets. The classification model learns from the training set and is then applied to the test set to evaluate the performance of classification. A wide variety of learning algorithms for learning the function $f$ have been

proposed and utilized in data mining (Duda et al., 2012). In the following subsections the most common algorithms that have been used for solving OPI problems are introduced.

### 2.4.1. OPI Standard Classification Methods

This section is dedicated to introducing concrete data mining classification algorithms which have been used in OPI problems. First, we describe a standard probabilistic model called Naïve Bayes which has been used for text classification since the last two decades. Then we discuss an intuitive algorithm which is usually used in information retrieval called K-NN. Then we proceed to more advanced algorithms such as Entropy-based classification, Support Vector Machines and Artificial Neural Networks.

### 2.4.2. Naïve Bayes

This model is used extensively as a baseline in text classification studies. This means that researchers accept it as an efficient algorithm and aim to improve its performance through other novel algorithms. Although the reader can refer to Duda et al. (2012) for a thorough explanation of the algorithm, we present a brief description here.

Let $\boldsymbol{D}$ be the dataset defined in the previous section. Assuming that all discrete-valued features are conditionally independent given the class label $y$ (known as '*Naive Bayes assumption*') we can simplify $P(\chi|y)$ that is the conditional probability of observation vector $\chi$ given $y$ as follows:

$$\begin{aligned} P(\chi|y) &= P(x_1, x_2, \dots, x_n|y) \\ &= P(x_1|y)P(x_2|x_1, y) \dots P(x_n|y, x_1, x_2, \dots, x_{n-1}) \\ &\approx P(x_1|y)P(x_2|y) \dots P(x_n|y) \\ &= \prod_i P(x_i|y) \quad \textit{by Naive Bayes assumption} \end{aligned} \tag{4}$$

Having $P(\chi|y)$ calculated as above, one can predict the most likely class label $(y^*)$ by using the Bayes rule as follows:

$$y^* = \underset{y}{argmax}\, P(y)\, P(\chi|y) = \underset{y}{argmax}\, P(y) \prod_i P(x_i|y) \tag{5}$$

Specifically in OPI problems, we deal with labels *p* and *np* and we can rewrite the above as:

$$y^* = \max\left( P(y = p) \prod_i P(x_i|y = p)\,, P(y = np) \prod_i P(x_i|y = np) \right) \tag{6}$$

### 2.4.3. K-Nearest Neighbor

Kang et al. (2012) have used K-Nearest Neighbor (also known as KNN) for online predator identification. They have used a weighted modification of classic KNN model. Although their result is not comparable with other methods, their approach is worth mentioning due to the interesting justification behind it.

The simplest version of the algorithm can be outlined as follows. More sophisticated explanation can be found in Duda et al., (2012).

**Algorithm: KNN(D, $\chi$, k, m)**
Input:
$\boldsymbol{D} \leftarrow Training\ dataset$
$\boldsymbol{\chi} \leftarrow\ query\ vector$
$\boldsymbol{m} \leftarrow distance\ measure\ (e.g. Euclidean\ Distance\ )$
Output:
**y:** predicted class label 'p' or 'np' (i.e., deciding whether the sample $\boldsymbol{\chi}$ is predatory or not)
$\boldsymbol{N}, \boldsymbol{L} \leftarrow \emptyset$
- Find **k** nearest neighbors to $\boldsymbol{\chi}$ based on the distance measure $\boldsymbol{m}$ and add them to neighbors set **N.**
- Add the corresponding class labels of found neighbors in **N** to labels set **L.**
- $\boldsymbol{y} \leftarrow majority(L)$

where the *majority* function simply calculates the majority of class labels among the *k* nearest neighbors.
There are also weighted versions of KNN which assign different weights to the neighbors based on their distance to the query point (i.e., $\boldsymbol{\chi}$). The authors of (Kang et al., 2012) state that choosing a good value for *k* that can provide good results remains a challenge.

### 2.4.4. Maximum Entropy Classification

The conditional independence assumption of Naive Bayes is not realistic at least when dealing with textual documents. Therefore, other statistical models have been proposed to consider the notion of dependent random variables.
A Maximum Entropy Classifier is a *discriminative* approach that tries to build a statistical model of conditional probability distribution $P(y|\chi)$. There is an infinite number of such models, but based on the maximum entropy principle, the best model is the one which maximizes the entropy *H(P)*. Berger et al. (1996) formally state this as follows:

$$H(p) \equiv -\sum_{\chi,y} \tilde{p}(x)p(y|\chi)logp(y|\chi) \tag{7}$$

In order to choose the best model $p^*$ from set C (the set of valid probability distributions), based on the maximum entropy selection, we have to solve the following optimization problem:

$$p^* = \underset{P \in C}{argmax}\, H(P) \tag{8}$$

Note that the above optimization problem is a constrained one. Nevertheless, describing the complete theoretical background of Maximum Entropy Classifier is beyond the scope of this chapter. The keen reader may refer to the seminal paper by Berger's et al. (1996) about maximum entropy classifier for natural language processing.

Eriksson and Karlgren (2012), and Kern et al. (2012) used a Maximum Entropy Classifier in the OPI domain on the PAN-2012 dataset. Their results are comparable with the best performing approach in PAN-2012 and can be considered as a successful approach for addressing OPI problems.

### 2.4.5. Support Vector Machines

Currently, Support Vector Machines (SVMs) have shown the best results among all different classification algorithms which have been used for OPI problems (Villatoro-Tello et al., 2012). SVM was originally introduced by Vapnik, (1995) in his book entitled *'The Nature of Statistical Learning Theory'*. The goal of the SVM algorithm is to give a hyperplane that maximizes the margins between positive and negative instances (predatory and non-predatory samples in our case). The margin is defined as two times the distance from the decision hyperplane. Figure 6 illustrates the idea of SVM.

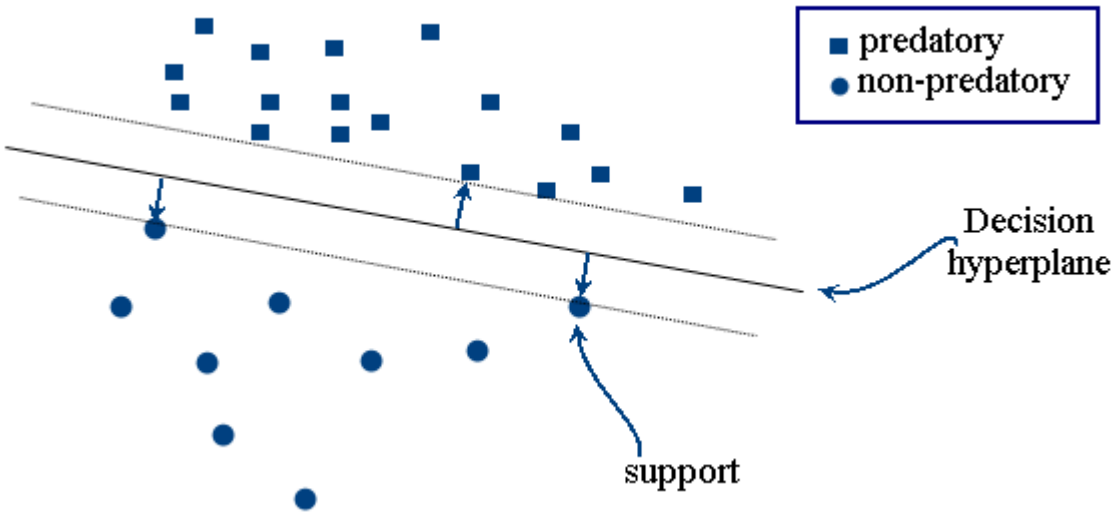


*Figure 6. An example of showing the output of SVM for a binary classification as well as the margin and three support vectors.*

Instances located on the margins are called *'support vectors'*. The main result of training an SVM model is to obtain these support vectors also known as *supports*. These instances are the only ones that are taken into consideration when a new instance needs to be classified by the model. The training procedure consists of solving a constrained optimization problem, which is usually solved by out-of-the-shelf quadratic programming tools. SVM has been used by Morris (2013), and Villatoro-Tello et al. (2012) and as already mentioned, SVMs have obtained the highest performance at PAN-2012. The reader may refer to Bishop (2006) for the mathematical background of support vector machines.

Note that besides the classification algorithm, the preprocessing methods and feature extraction methods described in Section 2.3 have a significant impact on the final performance of the classification.

### 2.4.6. Neural Networks

Even though there are a large variety of neural network algorithms, the one that has been successfully utilized in OPI is the Multi Layer Perceptron (MLP). Villatoro-Tello et al., (2012) tested MLP for identifying predatory conversations in addition to identifying the likely predators and then compared their performance with that of Support Vector Machines. The results showed that the performance of the MLP and the SVM are comparable to each other. More specifically, the neural network outperformed SVM in identifying predators versus victims; while the SVM performed better in identifying the predatory conversations.

Here, we introduce the key concepts of MLPs without a detailed description of algorithm.

Units are considered as building blocks of MLPs. Each unit resembles a neuron that has a specific *'activation function'* that generates the output of a neuron. At a higher level of abstraction, the network contains several layers of units including input, output and one or more hidden units. The units in a layer are not connected to each other, while usually all of the units in a previous layer are connected to the units in the next layer. A real number called *'weights'* is assigned to each connection. The final output $z_j$ of a neuron located in the first hidden layer with M units is calculated as follows (Bishop, 2006):

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) \tag{9}$$

where *i* denotes the index of the features in the input vector and *j* denotes the index of the neurons located in the first layer of the network. Also *h* is the activation function for the neurons in the hidden layer which is usually chosen to be a sigmoidal function. *D* is the number of features in the input sample. Finally, superscript (1) denotes the layer in which neuron is located. This process of linear transformations continues in a cascading manner from previous layers to the last hidden layer and eventually to the output layer, in which the final output of the network is generated.

The learning algorithm of an MLP with a specified structure finds the relatively optimal network weights via local optimization. The learning procedure encompasses a technique called '*backpropagation'* and also an optimization technique called '*gradient descent'*. For a complete description of Neural Networks the reader may refer to Bishop's book (2006). There are several out-of-the-shelf libraries and tools for building and training neural networks.

In terms of accuracy, precision, and recall rate (see Section 2.4.7), Support Vector Machines and Neural Networks (Multi-layer Perceptron) seem to outperform other classification methods for the task of OPI. However, it is worth to apply a simple approach such as Naïve Bayes to obtain a performance baseline so that if there is a problem with parameter tuning of the two mentioned algorithms, it will be revealed at the first stages of the analysis.

### 2.4.7. Performance Measures

Precision and recall rate in addition to their harmonic mean ($F_1$-score) are usually used as performance measures in classification. Let *tp* be the number of predatory conversations identified correctly (i.e., true positive), *tn* be the number of non-predatory conversations identified correctly (i.e., true negative), *fp* be the number of non-predatory conversations identified as predatory by mistake (i.e., false positive), and

finally let $fn$ denote the number of predatory conversations identified as non-predatory by mistake (i.e., false negative). Then the precision is calculated as $tp/(tp + fp)$ and the recall would be $tp/(tp + fn)$.

## 2.5. Anomaly Detection Literature Review

The first successful attempt to use machine learning in OPI problem to distinguish predators from underage victims was done by Pendar by means of a weighted K-NN classifier (Pendar, 2007). To the best of our knowledge, the first empirical system with the capability of determining predatory messages in chat logs is ChatCoder1 (and Chatcoder2) implemented by Kontostathis and her colleagues (Kontostathis, 2009; Mcghee et al., 2011). The system uses a rule based approach in conjunction with decision trees and instance-based learning (KNN). It is worth mentioning that in (Tan, 2005) the author introduced a general approach for using a weighted version of the KNN algorithm to mitigate the problem of learning from imbalanced datasets in text categorization.

The PAN-2012 conference has acted as a boost for applying machine learning techniques to online sexual predator identification. The main strength of this conference is providing the first publicly available standard dataset which was specifically developed for the sexual predator identification task. Researchers tuned their proposed methods against the same training data and reported their performance on the official test data. Several machine learning algorithms have been used to address the OPI problem in this competition. These algorithms cover a wide range of classification algorithms such as maximum entropy-based classification (Eriksson & Karlgren, 2012), KNN (Kang et al., 2012), Support Vector Machine (Morris, 2013) and Neural Networks (Villatoro-Tello et al., 2012). The top team (Villatoro-Tello et al., 2012) has used a two-step binary classification approach called SCI (Suspicious Conversation Identification) and VFP (Victim From Predator Disclosure) using SVM and Neural Networks.

Inspired by the approach used by Villatoro-Tello (2012) we have used the SVM method to compare the performance of our anomaly detection approach. Escalante and his colleagues (Escalante et al., 2013) proposed a new method based on learning a chain of three local classifiers corresponding to three segments of each conversation but the approach could not outperform that of the top performing method in PAN-2012.

A related research has been done on cyber bullying by Kontostathis et al. (2013), which is very close to predator identification. They utilize a different supervised learning algorithm based on Latent Semantic Indexing (LSI) that is called *Essential Dimensions* for identifying cyber bullying. They built their own dataset using 'formspring.me', a question-and-answer popular website.

More recently, Cano et al., (2014) have proposed enriching the traditional bag-of-word approach by adding other feature types including sentiment features, psycho-linguistic features and discourse patterns. Eventually, they have used a binary classification for the actual predator identification task.

Generally, the algorithms used in PAN-2012 can be considered as the state of the art in sexual predator identification. However, in regard to anomaly detection, there is a wide variety of unsupervised, supervised, and semi-supervised models. A comprehensive survey of

anomaly detection has been done in (Chandola et al., 2009). The authors have categorized anomaly detection methods into six major categories: clustering based, classification based, nearest neighbor based (which includes density based methods), statistical, information theoretic and spectral methods.

We use a slightly different taxonomy to show where our method stands with respect to the learning method that is used for anomaly detection. We avoid describing different methods and foundations of anomaly detection since it is beyond the scope of this thesis. Instead, we focus on the specific anomaly detection method (i.e., one-class SVM) that yielded the competitive results in this application domain based on our results. Figure 7 illustrates the taxonomy of the most common anomaly detection techniques as well as the position of semi-supervised techniques. One-class SVM has been highlighted in the figure. For the sake of completeness, the unsupervised SVM-based algorithms are shown as well. The corresponding leaf nodes of the taxonomy will be introduced in the next section.

Recently, several works have addressed the problem of anomaly detection in micro-blogs or short messages especially in Twitter (Anantharam et al., 2012), (Guzman & Poblete, 2013). (Kumaraswamy et al., 2015) et al. have used domain-specific features encoded as first order logic for textual anomaly detection. Anomaly detection methods have not been applied to the SPI problem.

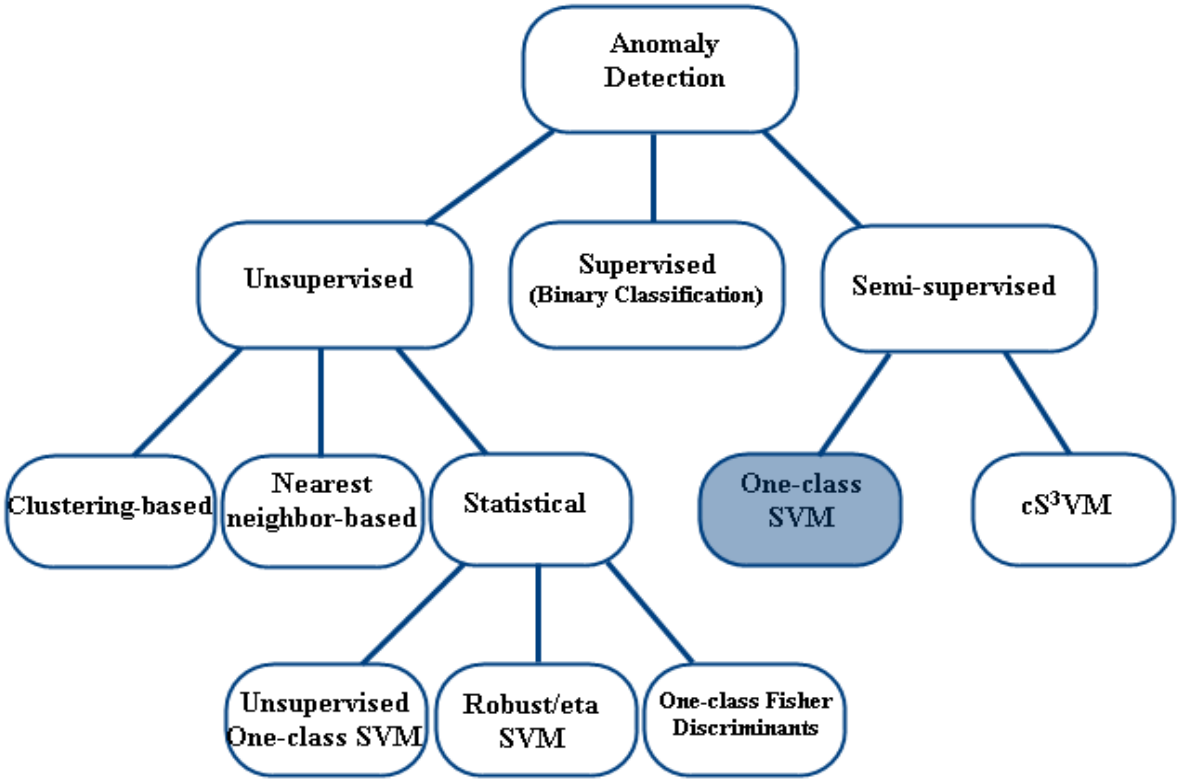


*Figure 7. Position of Semi-supervised and SVM-based techniques in the taxonomy of anomaly detection techniques*

To our knowledge, anomaly detection has not been applied to the OPI domain. As a new area of application, we examined this approach on the PAN-2012 data and we discuss the results of this approach and compare it with other more commonly used two-class classification

methods in the following sections. Chapter 3 discusses the adaptation of the notion of anomaly detection to sexual predator identification.

## 2.6. Deep Learning Literature Review

Deep learning has recently emerged as a hot trend in artificial intelligence and machine learning. Most commonly used deep architectures can be categorized into two major types: Discriminative and Generative models. In a probabilistic setting, assuming that $x$ denotes the observed input variable and $y$ is the unobserved target variable, discriminative approaches directly model $P(y|x)$ and $P(y)$ while generative models focus on $P(x|y)$. Ng & Jordan (2002) describe these two approaches in detail for classification problems. Figure 8 depicts the taxonomy of deep learning architectures and also illustrates the location of Deep Neural Networks as well as the methods used in NLP and textual analysis domain.

Generative models can be divided into three main sub categories: Deep Belief Networks (DBNs), Energy-based models and Neural-Network-based models. DBNs are graphical models composed of several stochastic hidden units. Hinton et al. (2006) proposed the first efficient learning and inference algorithms for DBNs. Lecun et al. (2006) characterized energy-based models by associating a scalar energy value based on an energy function $E(y,x)$- to each combination of variables. Hence, the learning algorithm is defined as finding the appropriate energy function and the inference is equivalent to finding the configuration of variables that minimizes the energy.

Autoencoders are basically traditional neural networks with the same input layer as the output layer with the goal of learning a representation of input data (usually called encoding) which can be expressed in a lower dimension as that of the original input. Deep Autoencoders are usually used as a pre-training phase in many of the deep learning models for feature extraction.

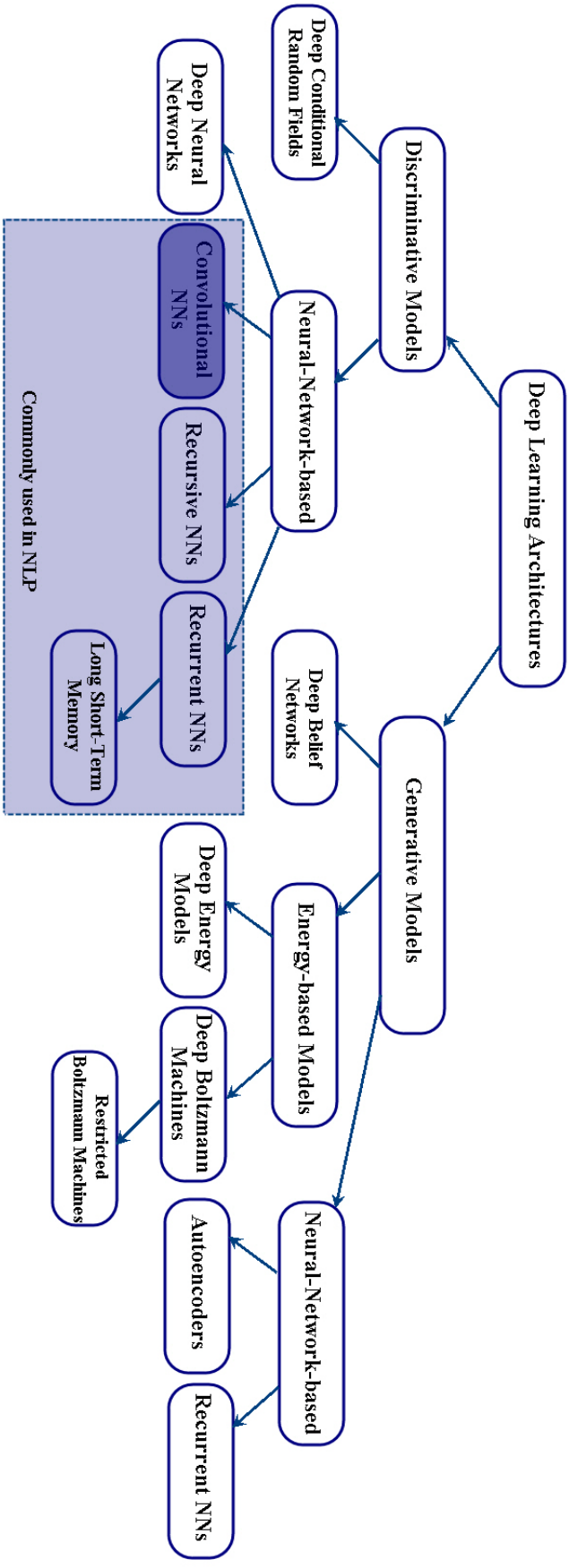


*Figure 8. Taxonomy of deep learning architectures*

Discriminative models contain neural-network-based methods and deep conditional random fields. These models encompass the notion of a traditional neural network, which is characterized by defining affine transformations on a set of random variables and applying a non-linearity afterwards (Bengio, 2009). This means that assuming the input as the first layer $h^0$, the output of layer $h^k$ is defined by the following formula (Bengio, 2009):

$$h^k = \sigma(b^k + W^k h^{k-1}) \tag{10}$$

in which $\sigma$ is a non-linear function, $b^k$ is the bias (offset) for layer k and $W^k$is the weight matrix between layers *k* and *k-1*. Very often, the non-linear function $\sigma$ is *sigmoid*, *tanh* or *rectifier* function. These functions are as follows:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{11}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

$$rec(x) = Max(0, x) \backsimeq \ln(1 + e^{x}) \tag{13}$$

In this thesis, we focus on the architectures that are common in text mining and NLP. These models are highlighted in Figure 8. These models are Recursive Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, and Convolutional Neural Networks. Here we describe the gist of each of these algorithms and we will delve into more details of Convolutional Neural Networks since it is the approach that is used in this work.

Recursive Neural Networks have unique characteristics that make them suitable for structured data such as natural language processing tasks that often can be represented by parse trees (Socher, et al., 2011). In this architecture, the computation units (i.e., neurons) are usually arranged in a tree structure. The learning phase leverages a variant of the backpropagation approach called "backpropagation through structure". Socher et al. (2013) have proposed a variant of this model for accomplishing sentiment analysis and improved the state of the art by 5.4% in positive/negative sentiment classification. Their method is interestingly capable of handling negations (Socher et al., 2013). Figure 9 shows the visualized output of their method which is interestingly capable of handling negations (Socher et al., 2013).

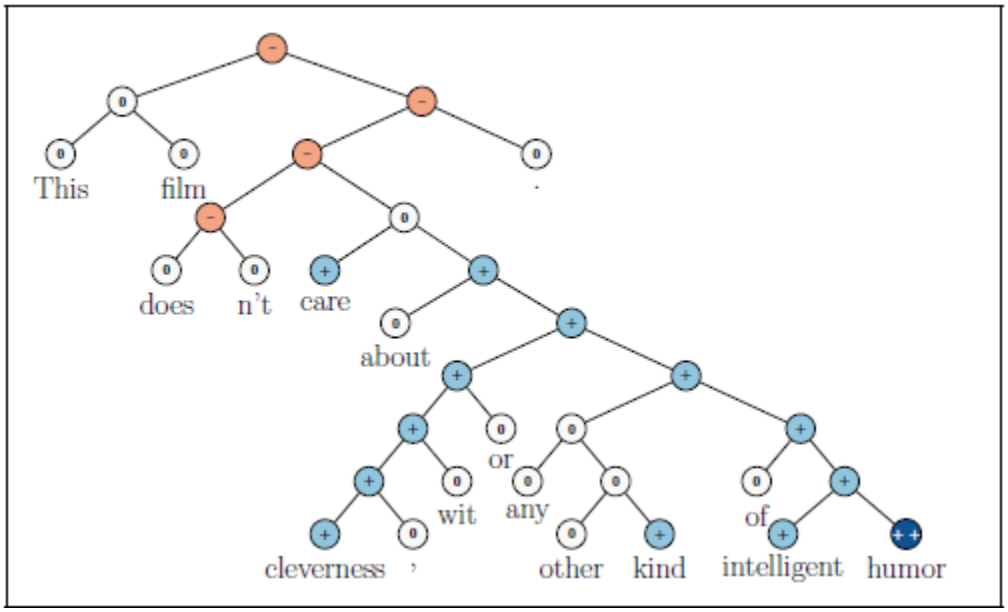


*Figure 9. An example of fine-grained labeling and percolation of sentiments by using Recursive Neural Network (Socher et al., 2013)*

Recurrent Neural Networks are designed for processing variable-sized sequential input data in which the observed input affects the probability of observing subsequent inputs. Unlike the bag-of-words model that considers each word independently, Recurrent Neural Networks can deal with this type of data efficiently (Mikolov, et al., 2010) . That is, the generated output at time *t* depends on the output that has been generated by the network in time stamp *t-1*. In practice, the output is only influenced by a limited number of previous outputs. This behavior mimics the notion of temporal memory in these systems. This is an important property that makes these models significantly more efficient in processing sentences as inputs.
The abstract architecture of this model has been illustrated in Figure 10.

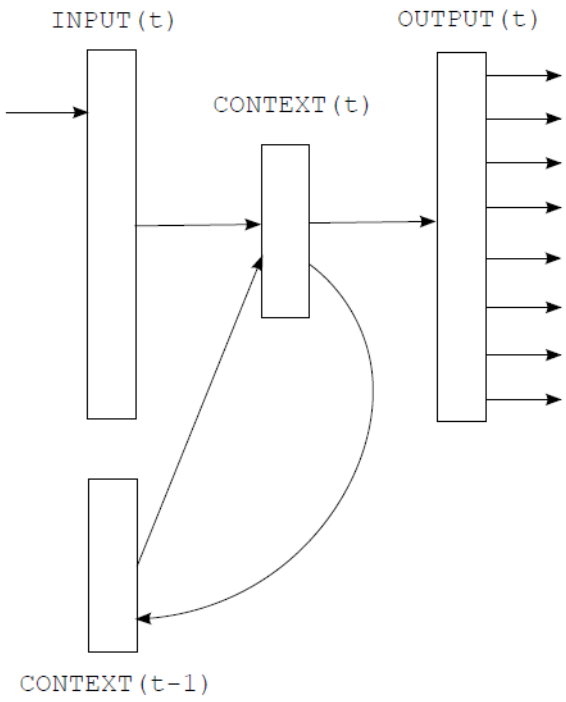


*Figure 10. Abstract architecture of a Recurrent Neural Network (Mikolov et al., 2010)*

It is worth mentioning that, as is shown in the taxonomy of figure 9; Recurrent Neural Networks can be used in both generative and discriminative setting. In discriminative RNNs, the goal is sequence labeling, while in a generative setting, the goal is generating the next likely sequence.

As a good example, sequential data analysis can be applied to opinion mining from textual sentences. Irsoy and Cardie (2014) show that applying recurrent neural network outperforms the state of the art method (a variant of Conditional Random Field). The interesting point in their work is that the performance has been achieved without using a standard hand-curated sentiment lexicon and syntactical analysis required in previous successful approaches (İrsoy & Cardie, 2014). The learning algorithm for Recurrent Neural Network is known as *Backpropagation Through Time* and is basically a variant of the traditional backpropagation that also takes the partial derivatives into account in each of the previous time stamps.

Considering many previous time stamps require calculating the corresponding gradients in all of them. This leads to a well-known challenge in training Deep Recurrent Neural Networks called *vanishing gradients*. That is, if we want to consider relatively long time stamps (in order to capture the dependency in the sequence) the common activation functions such as *sigmoid* or *tanh* go to the saturated zone. Saturation happens when the partial derivative of the activation function is almost zero. Long Short-Term Memories (Hochreiter & Schmidhuber, 1997) is a special type of Recurrent Neural Networks to alleviate this problem. Since the scope of this work does not contain Recurrent Neural Network, we do not delve into more details about this particular successful type of deep models.

Another important notion in using deep learning in natural language is called *word representation* or *word embedding*. It is different from the traditional bag-of-words document representation in the sense that the feature vectors are not simply comprised of the frequency

of each word that exists in the document. Instead, each word has its own vector representation and the final feature set of a document is obtained by the concatenation of these vector representations (a.k.a. embeddings) to form a feature matrix. The first word embedding was introduced by Rumelhart et al. (1986) and it was a technique that represented features (words) based on backpropagation errors. Recently, many word embedding techniques (sometimes called language models) have been proposed in the literature. Among these methods, *skip-gram* method, proposed by Mikolov et al. (2013), turned out to be more efficient compared to the others. The implementation of the method is called Word2Vec and has been excessively used by researchers recently. A similar variant has been also provided for obtaining the representation of sentences and paragraphs in (Le & Mikolov, 2014). Later, Pennington et al. (2014) introduced Global Vectors (GloVe) that outperformed the Word2Vec representation.

We dedicate the remaining part of this section to introduce Convolutional Neural Networks (CNNs). We describe the usage of this model in detail in Chapter 4 when we see it in practice applied to our domain of OPI.

### 2.6.1. Convolutional Neural Networks for Texts

Being inspired by visual systems, Convolutional Neural Networks (CNNs) have been successfully used in various image processing tasks (LeCun, Kavukcuoglu, & Farabet, 2010). CNNs have been recently applied to the text mining and natural language domains and showed promising results that could push forward the performance of the state of the art on several datasets. Zhang and Wallace (2015) has authored a holistic guide on designing CNNs for sentence classification. They depict a general CNN for a binary sentence classification task with only one convolution and one pooling layer as shown in Figure 11. Since this architecture is common in many CNN applications for text classification, we describe it here to help the reader gain a basic understanding. The input is the concatenation of word representations (word embedding) of the words in the sentence in the form of a sentence matrix. Then the whole input is divided into an arbitrary number of regions (see Figure 11). Each region can have its own filters. One can think of filters as a linear transformation of the features. Then each filter is applied on the input sentence matrix to form a variable-sized vector. This is called convolution in the neural network literature because literally applying the filter makes some of the connections in the network off, which leads to having a partially connected network in the corresponding layer.

The inherent problem with the output of the convolution layer is that the output is variable-sized. The standard solution to solve this is having a *pooling* layer right after the convolution layer. This layer aggregates each output vector as a singular value by simply taking the maximum or average among the elements of each feature vector. However, this is not the only reason for having the pooling layer. Pooling also acts as a sub sampling procedure, which enables the CNN to capture more abstract aspects and characteristics of the data. As a result, each layer captures more abstract features compared to the previous layer.

Finally, the output layer is usually a fully-connected layer with a softmax activation function or it can be another multi–layer network that operates on fix-sized feature vectors in order to produce the final classification results (in this case positive/negative).

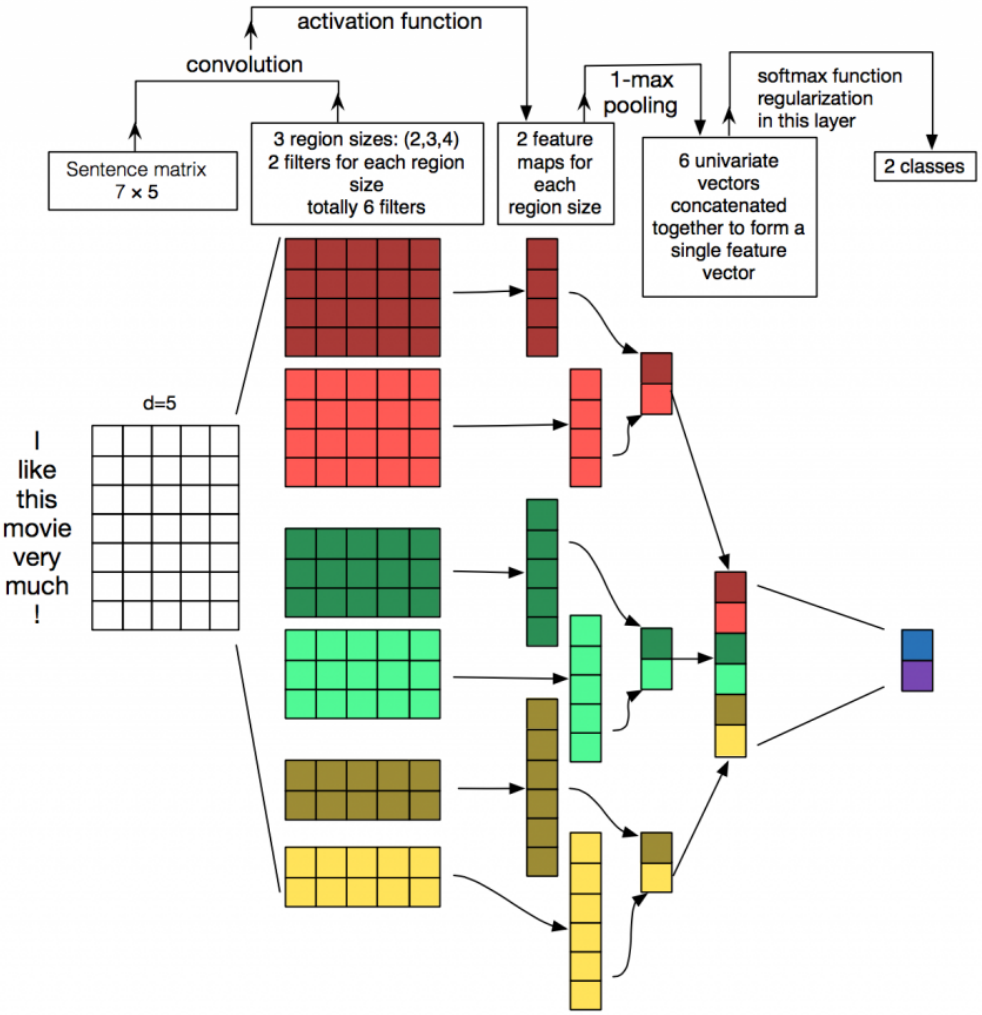


*Figure 11. The general CNN's architecture for sentence classification (Zhang & Wallace, 2015)*

Using CNNs, Collobert et al. (2011) proposed a unified general-purpose language framework that is capable of performing a variety of NLP tasks including part-of-speech tagging, chunking, named entity recognition and semantic role labeling.

This common architecture utilizes a specific word embedding approach in which the first layer encodes words by using their index in a general dictionary and then the network is trained using backpropagation to obtain the feature vectors. Then the output of this layer is fed to the convolution layer. The architecture has two modes of operation (called approaches in the paper): *window approach* and *sentence approach*. In the former approach the number of words that are being fed to the first layer are fixed to a specified number by the user, while in the latter, the number of words are variable depending on the size of each sentence. Figure 12 shows the architecture in sentence approach mode (Collobert et al., 2011).

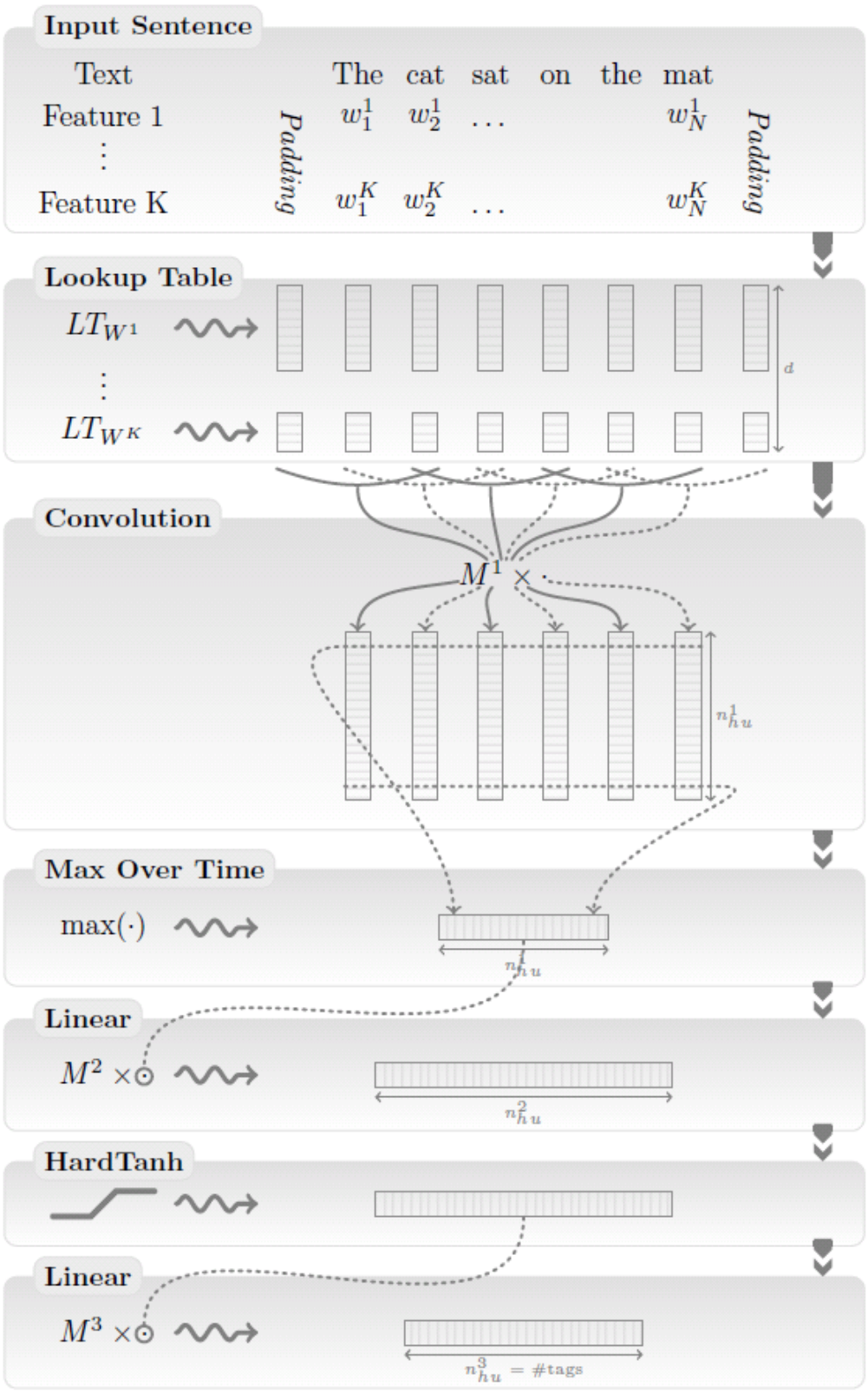


*Figure 12. The sentence approach generic architecture proposed for sentence classification (Collobert et al., 2011)*

Another successful application of CNNs to the test classification has been introduced by Kim (2014). He used the Word2Vec word embedding in order to obtain the feature vectors and then applied the original architecture of the CNN in order to carry out seven NLP tasks mainly related to sentence classification and sentiment analysis. Surprisingly, this model has pushed the state of the art in four of the tasks just by having one convolution layer, one pooling layer and a softmax classifier. Figure 13 shows this model (Kim, 2014).

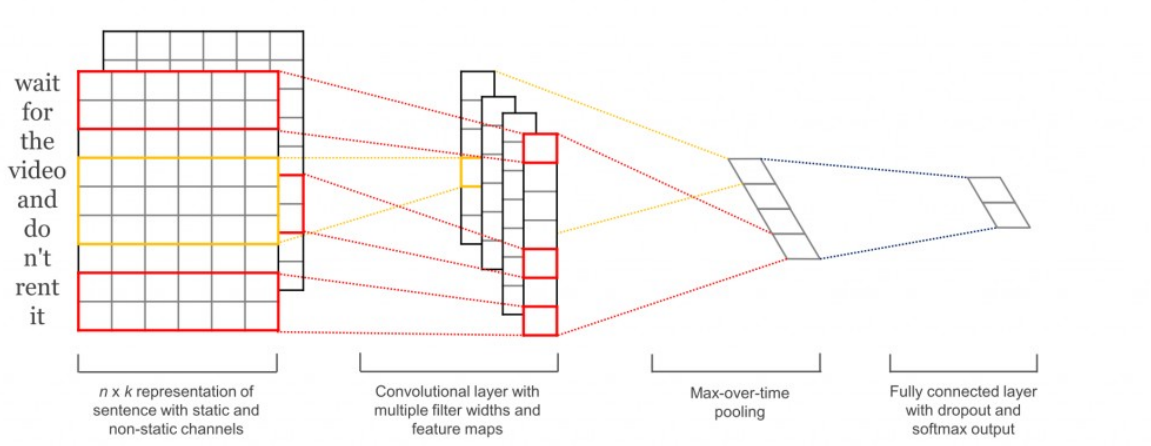


*Figure 13. CNN-based sentence classification model using Word2Vec embedding and max-pooling (Kim, 2014)*

As a practical example related to our application domain, Dwyer[2] has applied the same architecture for filtering malicious chat messages. The system's goal is to filter the chat messages identified as potentially obscene or inappropriate in real time. He obtains the pre-trained word representation by applying Word2Vec and then run a CNN with the same architecture as described in (Kim, 2014). In his work, the main goal was to classify each individual message into eight categories ranging from 'super safe' into 'obscene'. Since the chat messages are relatively short, his model has only 50 units in the convolution layer. The research shows that the model outperforms the traditional neural network with hand-curated features. Figure 14 shows the performance of this approach.

[2] https://www.mitacs.ca/en/projects/word-representation-learning-detecting-malicious-chat-messages

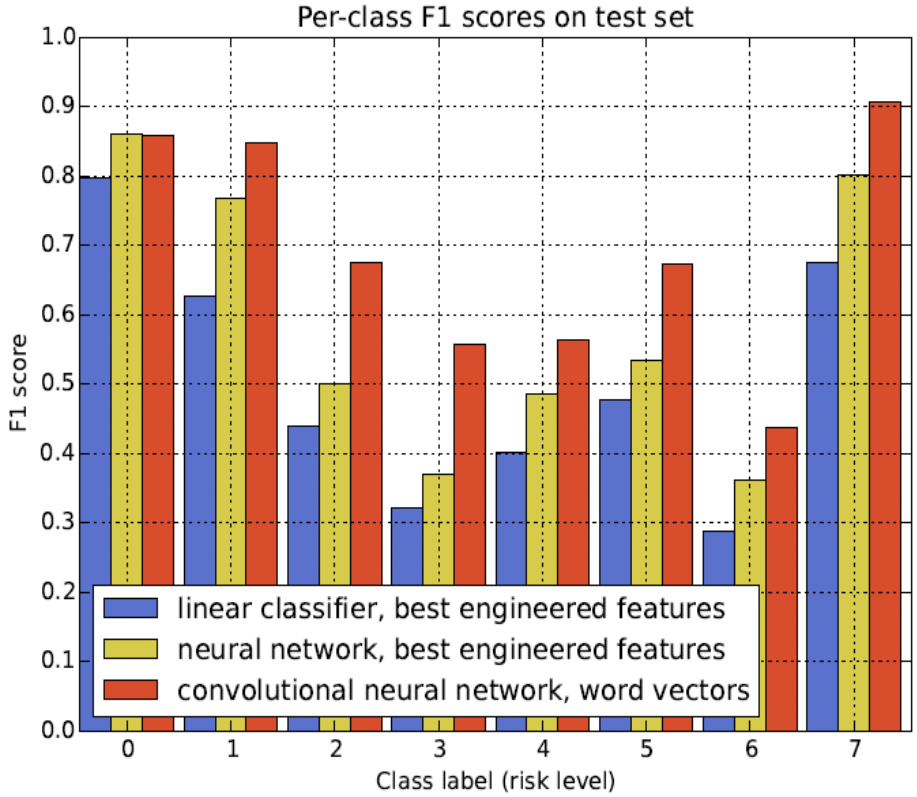


*Figure 14. The performance of CNN compared to traditional classification approaches measured by $F_1$-score (Dwyer, 2015)*

A more complex variant of CNNs called Dynamic CNNs has been introduced for sentence classification by Kalchbrenner et al. (2014). The model enhances the pooling mechanism and is able to form feature graphs for each sentence to capture the semantic relations.
Johnson and Zhang (2015a) proposed a novel supervised approach for using CNNs without any pre-trained embeddings upfront. This is the method that we applied to this domain. They have also introduced a semi-supervised setting (Johnson & Zhang, 2015b).
The specific characteristics of the data in OPI domain make it different from domains such as sentiment analysis and topic classification which usually deal with general web or news documents. As a result, inspired by Johnson's approach (Johnson & Zhang, 2015a), we do not aim to use general pre-trained word vectors that are common in these aforementioned domains. Accordingly, we build the word embeddings internally in the CNN training process.
In this thesis, Johnson's approach (Johnson & Zhang, 2015a) is used to create specific word vectors directly from the input in the training process. We investigate empirically the applicability of this approach on the PAN-2012 dataset in Chapter 4 and discuss the results.

### 2.6.2. Deep Learning Tools and Frameworks

It is worth knowing the characteristics of different existing deep learning tools so as to choose them according to the corresponding application domain. Table 5 summarizes and compares some of the characteristics of deep learning framework/tools as well as their pros and cons. Since we aimed to specifically use convolutional neural networks on relatively

large documents, we chose ConText2.0 as an out of the box library to help us to investigate the performance of different CNN architectures.

*Table 5. Comparison of Deep Learning Frameworks*

| Framework | Language | Algorithm Coverage | Developer | Characteristics |
|---|---|---|---|---|
| Caffe[3] | C++/Cuda | CNNs | Berkeley Vision and Learning Center | • Highly efficient for image processing with ConvNets<br>• Specific to image and machine vision |
| Theano[4]/ PyLearn2 | Python | • Restricted Boltzmann Machines<br>• Stacked Denoising Autoencoders<br>• CNNs | LISA lab at the University of Montreal | • General-purpose<br>• requires symbolic math expressions |
| Torch[5] | Lua Script | • Restricted Boltzmann Machines<br>• Stacked Denoising Autoencoders<br>• CNNs | Facebook and Google | • Matlab-like script and Intuitive in usage<br>• General-purpose<br>• Learning curve for Lua language |
| ConText[6] | C++/Cuda | • Supervised CNN<br>• Semi-supervised CNN | (Johnson & Zhang, 2015a) | • High performance<br>• Specific to document classification<br>• Easy-to-use bash script support<br>• Runs on NVIDIA GPU |
| DL4J[7] | Java and Scala | • Restricted Boltzman Machines<br>• CNNs<br>• Recursive Nets<br>• Recurrent Nets<br>• Deep-belief Nets<br>• Stacked Denoising Autoencoders<br>• Deep Autoencoders | Sky Mind Company | • Faster than python<br>• General-purpose<br>• Transparent parallelism<br>• Work with Hadoop and Spark<br>• Slower development speed compared to scripting languages |
| CNTK[8] | C++/Cuda | • CNNs<br>• Recurrent Nets<br>• Long Short Term Memory Networks (LSTMs) | Microsoft | • Graphical User Interface<br>• Can run on multiple GPUs on multiple machines |
| TensorFlow | C++/python | • Multi-purpose | Google | • Multi-platform (CPU, GPU and Mobile Device)<br>• General purpose |

[3] http://caffe.berkeleyvision.org/
[4] http://deeplearning.net/software/theano/
[5] http://torch.ch/
[6] http://riejohnson.com/cnn_download.html
[7] http://deeplearning4j.org/
[8] https://github.com/Microsoft/CNTK

In this chapter, we described the common text classification algorithms background in addition to the literature review of anomaly detection and deep learning methods. In Chapters 3 and 4, we will describe the details of applying anomaly detection and deep learning to this problem domain respectively.

# CHAPTER 3

# ANOMALY DETECTION FOR OPI

As discussed in Chapter 1, machine learning techniques used in OPI usually require a large volume of high-quality training instances of both predatory and non-predatory conversations. However, collecting non-predatory conversations is not practical in real-world applications, since this category contains a large variety of conversations with many topics including politics, sports, science, technology and etc. Usually law enforcement agencies have a considerable amount of predatory or suspicious conversations that have been gathered during several years. While this can be leveraged in building a training set, they do not have the non-predatory data samples at hand.

We utilized a new semi-supervised approach to mitigate this problem by leveraging an anomaly detection technique called one-class SVM, which does not require non-predatory samples for training. We compared the performance of this approach against other state of the art methods that use both positive and negative instances.

We observed that, although anomaly detection approach utilizes only one class label for training (which is a very desirable property in practice), its performance is comparable to that of binary SVM classification. In addition, this approach outperforms the classic two-class Naïve Bayes algorithm which we used as our baseline in terms of both classification accuracy and precision.

We conducted the experiments on two datasets: 1) The large publicly available dataset in PAN-2012 (Inches & Crestani, 2012) and 2) a small practical dataset collected from an archive of real conversations gathered by the Sûreté du Québec, the police department responsible for combating online predator identification in the province of Québec.

## 3.1. Hypotheses statement

***Hypothesis 1 (revisited):*** Predatory/Non-predatory conversations can be represented as anomalous conversations that do not conform with the underlying data distribution. The problem can be casted to a *one-class* classification problem. The performance would be comparable to that of binary classification which uses two class labels.

***Hypothesis 1-2 (subsidiary):*** Anomaly detection algorithms are sensitive to noise and outliers. Therefore, they need to incorporate a noise removal process.

## 3.2. Our Contribution Revisited

According to some of the researchers who participated in PAN-2012, there has been an important weakness in the dataset of this competition: The non-predatory and non-sexual samples were exclusively gathered from publicly available Internet Relay Chat (IRC) logs, which mainly contain chats about computer and web technologies; therefore cannot represent "general conversations" (Morris, 2013). The samples in general conversation category (which are non-predatory) should however include a variety of topics such as sport, music, games, computer, etc. In practice, it is not an easy task to assemble such a training dataset. As a

result, the current top-ranked algorithms at PAN-2012 may have learned how to distinguish computer-related chats vs. sexual-related chats instead of identifying actual predatory chats in online cyber space. Accordingly, one can expect that their performance would decrease in real-world applications. In other words, we believe that although the top-ranked algorithms at PAN-2012 had a significant $F_1$-score on the test dataset (87% for the top team), since they rely on general samples that are not able to represent the non-predatory data properly, their performance is expected to decrease significantly in practical environments such as law enforcement. Therefore, in this work we propose a novel way to handle this problem by eliminating the need for having both class labels in the train dataset. Due to the absence of one of the class labels in the training process, our method will be more practical at the expense of having a lower, but still acceptable, $F_1$-score. Furthermore, in order to guarantee the efficiency of our approach we aim to beat the baseline (Naïve Bayes algorithm) in terms of $F_1$-score. Note that each chat conversation represents a document in our recognition process; hence, in the following sections of this thesis we use the terms "document" and "conversations" interchangeably.

## 3.3. Problem Definition

Let dataset $D$ be defined as $D \subset X \times Y$ where $X = \{\chi_1, \chi_2, \dots, \chi_n\}$ is the set of $n$ conversations $\chi_i; i \in \{1,2, \dots, n\}$, so that $\chi_i = (x_1, x_2, \dots, x_m)^T$ is the corresponding $m$-dimensional feature vector for the $i^{th}$ conversation containing $m$ feature values $x_j; j \in \{1, \dots, m\}$. Also, let $Y = \{p, np\}$ represent the set of two class labels corresponding to predatory and non-predatory instances, respectively. In a probabilistic setting, it is assumed that each conversation $\chi_i$ is roughly drawn from probability distribution $P(\chi)$. The anomaly detection task is defined as finding a probability distribution $P$ such that $P(\chi)$ is near one for the majority of samples considered as normal and contrarily close to zero for the majority of anomalous samples. One can choose $l \in \mathbb{R}$ as the threshold for recognizing a conversation as a predatory one when $P(\chi)<l$. The notion of anomaly is a domain-specific concept that is defined based on the properties of the problem domain. This means that an anomalous sample in a specific domain might be considered as normal in another area of application.

Figure 15 shows the probabilistic view of anomaly detection in the OPI problem for only two features.

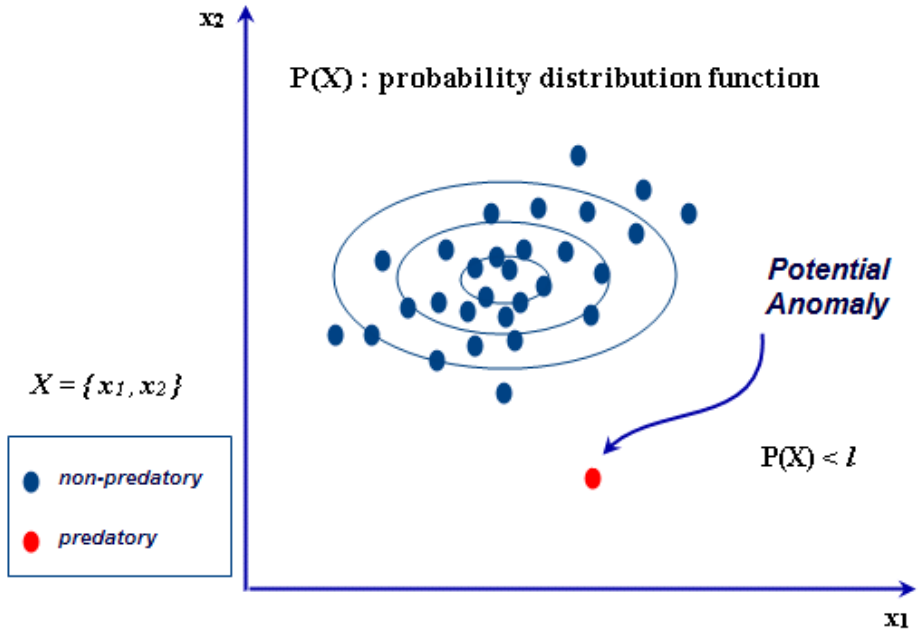


*Figure 15. Probabilistic view of anomaly detection in SPI setting (while predatory samples are considered anomalous)*

Anomaly detection, also known as novelty or outlier detection, is often referred to as finding instances which do not conform to the underlying pattern of normal data (Chandola et al., 2009). The following two research questions arise in regard to the application of semi-supervised methods to sexual predator identification:

***1) Why not use unsupervised anomaly detection?***

This can be shown that supervised and semi-supervised anomaly detection methods outperform unsupervised methods in terms of performance (Görnitz et al., 2013). We focused on semi-supervised techniques due to their superior predictive power compared to that of unsupervised methods. Although according to (Görnitz et al., 2013), the predictive power of semi-supervised methods comes at the expense of a weak identification of actual novel samples; in the domain of sexual predator identification, this weakness does not have a drastic impact due to the lack of such novelties that we may deal with in another domain, such as network intrusion detection.

***2) Why not use supervised anomaly detection?***

As already mentioned in Section 3.2, in our application domain, providing non-predatory samples is not practical. So we utilize a semi-supervised anomaly detection method that is capable of learning from only one class label in contrast to the binary (i.e., two-class) classification methods.

Moreover, one of the circumstances that justifies using an anomaly detection approach is when the data is naturally imbalanced. Because predatory samples are rare compared to non-predatory ones, we usually deal with datasets containing several hundred predatory conversations among several hundred thousands of non-predatory conversations.

It is worth mentioning that one can apply a reverse notion of anomaly in a manner that considers predatory conversations as normal ones and non-predatory conversations as anomalous.

## 3.4. One-class SVM

One-class SVM has been introduced by Scholkopf as a novelty detection technique and has been widely used in the area of anomaly detection (Scholkopf et al., 2000). The algorithm is a variation of ν-SVM (Schölkopf et al., 2000) which uses parameter $\boldsymbol{\nu} \in [\mathbf{0}, \mathbf{1}]$ to control the fraction of support vectors as well as fraction of outliers (anomalies). It is worth mentioning that in the standard SVM choosing the best regularization parameter $C \in [0, \infty)$ is a real challenge. ν-SVM tries to address this problem by introducing the parameter $\boldsymbol{\nu}$ that indirectly affects the regularization. The main idea of One-class SVM is to provide an algorithm that returns a function $\boldsymbol{f}$ with output +1 in a small region capturing most of the data points, and -1 elsewhere. The constrained optimization problem is defined as indicated in Equation 14 (Schölkopf et al., 2001).

$$\begin{gathered} \min_{w \in F, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho \\ with regard to: \quad w \\ subject to: \ \left(w.\phi(\chi_i)\right) \geq \rho - \xi_i, \\ \xi_i \geq 0 \end{gathered} \tag{14}$$

In Equation 14, $n$ is the number of conversations in the dataset, $\rho$ parameterizes a hyperplane in the feature space, $w$ is the weight vector, $\xi$ is the slack variable which penalizes the objective function and $\phi$ is the internal mapping function used in the kernel. Note that (.) in this notation represents the inner product.

The optimization problem can be solved by using the following Lagrangian in which $\alpha_i, \beta_i \geq 0$ :

$$L(w, \xi, \rho, \alpha, \beta) = \frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho - \sum_i \alpha_i \left(\left(w.\phi(\chi_i)\right) - \rho + \xi_i\right) - \sum_i \beta_i \xi_i \tag{15}$$

Finally, the decision function will be obtained as follows:

$$f(\chi) = sgn\left(\left(w.\phi(\chi)\right) - \rho\right) \tag{16}$$

Besides the original method described above, there is another variant of semi-supervised SVM-based technique for anomaly detection called cS$^3$VM (Chapelle et al., 2006). This method is based on the cluster assumption (i.e., there is a one-to-one mapping between clusters and classes.) Since the optimization problem in this setting is non-convex, the authors leverage a method to convert the non-convex optimization problem to a convex one by using a method called smoothing in an iterative manner.

Based on the given taxonomy, there are also several unsupervised methods for anomaly detection. One-class SVM can naturally be used in an unsupervised setting as well (Amer et al., 2013). Moreover, there are two unsupervised variations of SVM which have been recently introduced by Amer et al. (2013) called robust-svm and eta-svm. Since these versions are completely unsupervised, they sacrifice the performance (i.e., accuracy, precision, and recall) too much, so we chose to use the original method in this study. Using one-class SVM has led to acceptable results in the area of anomaly detection, but it has not been utilized in such a problem yet. In the following section, we describe the dataset as well as the results of applying this method on the OPI problem.

## 3.5. Experiments

This section describes the process that we have conducted to address the OPI problem including the dataset, preprocessing, feature extraction, and pattern classification. The proposed process that we have conducted on the dataset is illustrated in Figure 16.

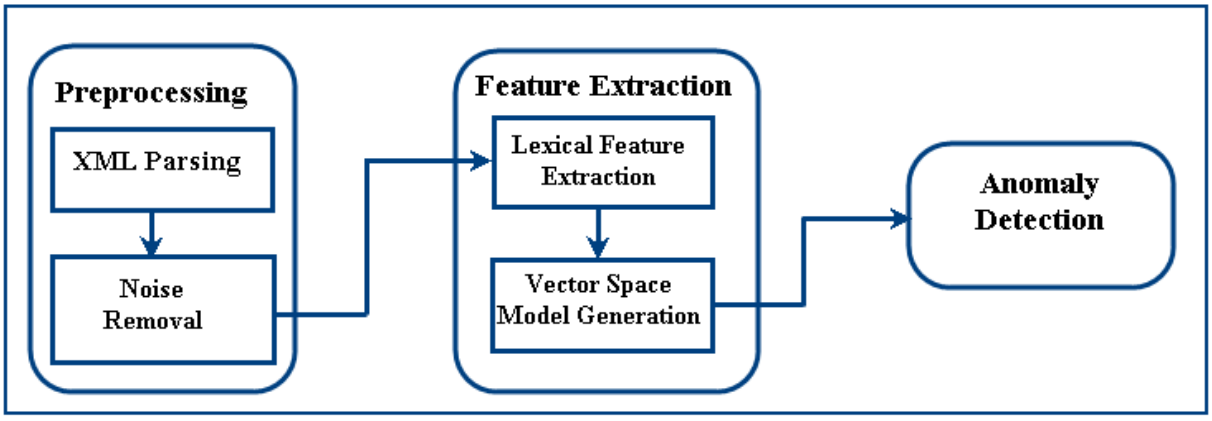


*Figure 16. Proposed Modular Process for Predator Identification*

### 3.5.1. Dataset

We used the training and testing dataset of PAN-2012[9], which today is the largest publicly available dataset according to our knowledge. This dataset is highly imbalanced. It contains 66,927 conversations in the training set and 155,128 conversations in the test set. There are 2,016 and 3,737 predatory conversations in training and testing set, respectively. These predatory conversations are related to 142 (out of 97,695 unique users) and 254 (out of 218,716 unique users) predators respectively. The total number of exchanged messages in the training corpus is 903,607. Table 6 summarizes the characteristics of the PAN dataset.

[9] http://pan.webis.de/

*Table 6. No. of conversations in the PAN Dataset*

| | **Training set** | **Testing set** | **Total** |
|---|---|---|---|
| Predatory | 2,016 (≈3%) | 3,737 (≈2%) | 5,753 |
| Non-Predatory | 64,911 (≈97%) | 151,391 (≈98%) | 216,302 |
| Total | 66,927 (≈100%) | 155,128 (≈100%) | 222,055 |

The SQ dataset is a small dataset which has been gathered from real chat logs of the Sûreté du Québec. Since this dataset has been obtained from a practical environment, unlike the PAN dataset, it contains many positive instances while the number of negative instances is low. All of the conversations are in French and we applied a French lexicon provided in RapidMiner library[10] for doing stop-word removal on this dataset. The following table summarizes the statistics of the SQ dataset.

*Table 7. Characteristics of the SQ Dataset*

| | | |
|---|---|---|
| SQ Dataset | No. of conversations | 82 |
| | No. of predatory conversations | 76 (≈93%) |
| | No. of non-predatory conversations | 6 (≈7%) |

In the PAN dataset, both the training set and test set are in XML format. The data schema has been shown in the Figure 17.

[10] https://rapidminer.com/

```
<conversations>
    <conversation id=1>
        <message id=1>
            <author id=1>AuthorName</author>
            <text>MessageText</text>
        </message>
        <message id=2>
            <author id=2>AuthorName</author>
            <text>MessageText</text>
        </message>
        ...
    </conversation>
    <conversation id=2>
        ...
    </conversation>
    ...
</conversations>
```

*Figure 17. Data Schema of Conversations in PAN-2012's Dataset*

In the SQ dataset the logs are stored in individual Microsoft word files. We extracted the corresponding text out of these files using the Apache Tika[11] software in our prototype software.

### 3.5.2. Experimental Settings

In our experimental setting we chose Naïve Bayes as a common binary text classifier as our baseline. Also, we tried to simulate the results of the top team at PAN-2012 for identifying predatory conversations based on Support Vector Machines. In addition, we performed two main categories of experiments: 1) training the model with non-predatory samples, and 2) training the model with predatory samples.

Table 8 shows the experiments we have conducted. We will refer to each experiment by its shortened name and describe the corresponding results Section 3.5.5.

[11] https://tika.apache.org/

*Table 8. Different Experiments Conducted in this setting*

| Experiment No. | Experiment Short Name | Experiment Description |
|---|---|---|
| 1 | *Train-NP-B* | Train one-class SVM on non-predatory conversations and bigram features |
| 2 | *Train-P-B* | Train one-class SVM on predatory conversations and bigram features |
| 3 | *Test-P-B* | Test one-class SVM on predatory conversations and bigram features |
| 4 | *Train-P-B-NR* | Train one-class SVM on predatory conversations with bigram features after noise removal |
| 5 | *Test-P-B-NR* | Test one-class SVM on predatory conversations with bigram features after noise removal |

The training model have been evaluated via k-fold cross validation with *k=10* and micro-averaging the results for each fold.

In order to evaluate the performance of the algorithms four common performance criteria have been used: accuracy, precision, recall and F-measure. Normally, the last measure is calculated as the harmonic mean of precision and recall and called $F_1$-score, unless one wants to weigh either precision or recall more than the other one. The general formula for the F-score is as follows (Chinchor, 1992):

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad (0 \leq \beta \leq \infty) \tag{17}$$

At the PAN-2012 international competition, both β=1 and β=0.5 were used as the main performance measures. The latter was used to put more emphasis on precision and raised controversies. Accordingly, in order to consider precision as important as recall, we use β=1 and calculated the widely-acceptable $F_1$-score as our main performance measure. We used RapidMiner™ (Mierswa, Wurst, et al., 2006) as an open-source powerful tool for our preprocessing and also LibSVM (Chang & Lin, 2011) for C-SVM and One-class SVM. The designed pre-processing steps are available on Github at the following address as an XML file that can be imported in Rapidminer:

https://github.com/mohammadrezaebrahimi/pre-process-PAN.git.

The process includes XML parsing, feature extraction, noise removal and feature selection tasks which are described in the remaining of this section.

### 3.5.3. Preprocessing and Feature Extraction

We parse the XML data and extract the raw textual document for each conversation. As most of the approaches in this domain, we leverage the bag-of-words model for feature extraction in our experiments and generated both unigram and bigram representation of the data to examine the performance of training on these two different features. Typically, there are three

options for data representation in text classification: 1) binary representation in which the occurrence of the specified term is encoded as 1 or 0 otherwise, 2) Term Frequency (number of occurrences), and 3) A normalized TFIDF weighting scheme such as the one that has been used in RapidMiner's[12] text processing plug-in (see Appendix C for the source code). We used the same waiting scheme using RapidMiner framework. This weighting scheme is described as follows:

Let *t* and *c* denote the *term* and the *conversation* in which the term has appeared respectively. Also let *N* be the total number of conversations and *df(t)* be the number of documents in which the term *t* has occurred. Finally, *tf(t,c)* is the term frequency of term *t* in conversation *c*. We used the normalized TFIDF weighting scheme in RapidMiner (see Appendix C for the source code). The weighted word vector $W = [w_1, w_2, \ldots, w_n]$ consists of elements $w_i$; $i \in \{1,2,\ldots,n\}$ and *n* is the number of the documents. The inverse document frequency of term *t* is indicated by *IDF(t)* and is calculated by Equations 18.

$$IDF(t) = \log\left(\frac{N}{1 + df(t)}\right) \tag{18}$$

Denoting the number of terms in the document by *n(t),* we obtain the non-normalized TFIDF-weighted value of feature *t* for $i^{th}$ document (Equation 19).

$$w_i^{non-normalized} = \frac{tf(t,c)}{n(t)} \times IDF(t) \tag{19}$$

Finally, the values are normalized by the L2-norm and the normalized word vector, $W^{normalized}$, is given by Equation 20.

$$W^{normalized} = \frac{1}{sqrt(\sum_{i=1}^{n} {w_i}^2)} \times W^{non-normalized} \tag{20}$$

The unigram or bigram features were obtained by regular *tokenization* and *stop-word removal* in RapidMiner™. The resultant unigram and bigram vector space models for training dataset contain 45,450 and 280,378 features, respectively before doing feature selection. This is also important to note that the number of features in a text classification problem depends on the different factors, such as the number of documents, the average length of the documents, and the language characteristics. Having too many features in the bag-of-words model can lead to a problem known as curse of dimensionality in which the feature space is so sparse that the classification model is not able to learn any useful pattern. As an example, in our work, we observed this happened with trigram features and as a result, the performance dropped significantly for the huge set of trigram features. As a practical example, Villatoro-Tello (2012) used 117,015 features in one of their successful models.

As a side note, unigram and bigram features are the most common representation techniques among bag-of-words approaches used in this domain. While Pendar (2007) has used trigram features some other researchers such as (Popescu & Grozea, 2012) have used Kernel-based

[12] https://rapidminer.com/

features at the character level, instead of the word level. However, their method's performance is not as successful as the bag-of-words method.

In addition, it is wise not to use *stemming* while we are dealing with informal conversational documents, which usually have informal writing styles. Because performing noise removal (at the term level) as well as stemming will distort the stylistic patterns the authors use in their conversations. According to (Villatoro-Tello et al., 2012), the predator may try to maintain the connection by writing "soryyyyyyyyy" in case the child feels bad about the inappropriate intimacy. As a result, we did not use any stemming for dimensionality reduction in our pre-processings.

Figure 18 depicts an illustrative example showing the preprocessing procedure for labeling conversations as predatory or non-predatory[13]:

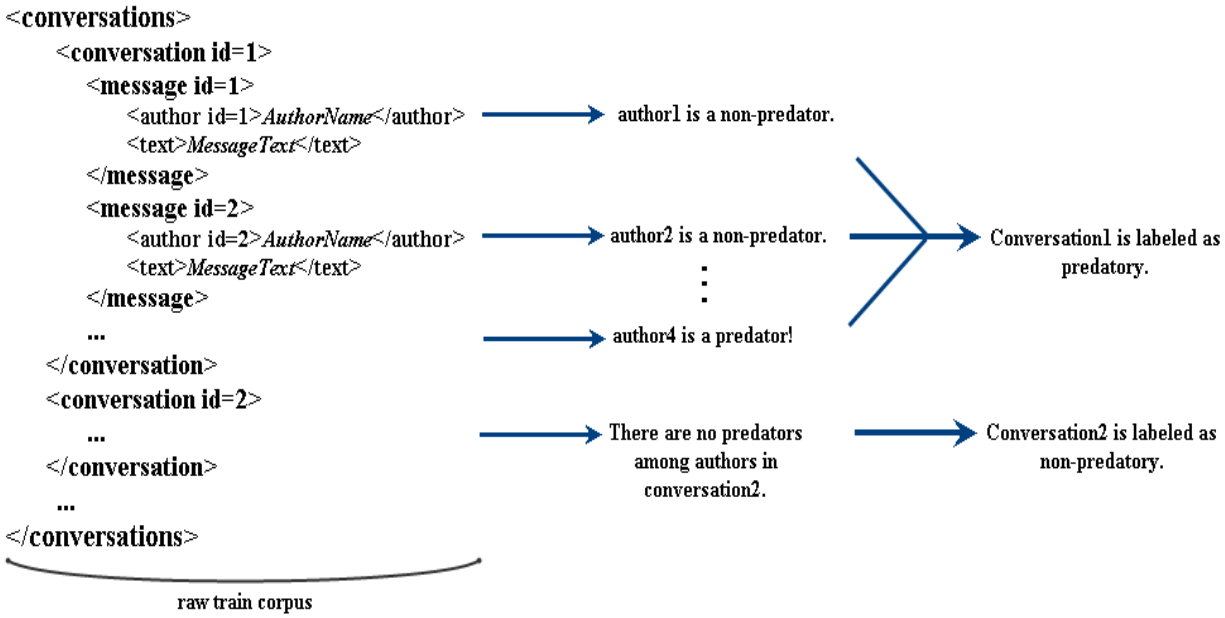


*Figure 18. Labeling Conversations in Training Data*

## 3.5.4. Feature Selection

In order to choose the most salient features we fed the primary features obtained from the previous phase into a supervised feature selection based on Information Gain. That is the amount of reduction in the entropy that might be obtained by leveraging feature *t*. The information gain on a dataset *D* for a candidate feature *t* is calculated based on the Equation 21.

$$IG(D|t) = H(D) - H(D|t) \tag{21}$$

[13] Note that although we have labeled both predatory and non-predatory conversations in the training dataset, we use only one of these two classes in model training unlike binary classifiers that leverage both of the class labels.

In which *H()* represents information entropy. We conducted several feature selection experiments to identify the best bigram and unigram feature set. However, it turned out that the bigram feature set leads to a better result for this dataset. The feasibility of each feature set was based on the performance of the classification on the training set using that feature set. We calculated the information gain for each of the features in the dataset and then sorted them in increasing order of their corresponding information gain. Then the top k-percent of the ordered set was selected each time to make five feature sets. Table 9 shows the feature sets in this experiment.

*Table 9. Different feature sets and their corresponding top-k selected features on the PAN dataset*

| No. | Top K-Percentage | Number of features |
|---|---|---|
| 1 | 60% | 168,227 |
| 2 | 70% | 196,265 |
| 3 | 80% | 224,302 |
| 4 | 90% | 252,340 |
| 5 | 100% | 280,378 |

Then we performed one-class SVM classification algorithm on each of the above five feature sets and measured the performance by four criteria: Accuracy, Precision, Recall, and $F_1$-measure. Figure 19 shows the performance for the feature sets. As it can be seen, the feature set containing 224,302 features has the best performance. We used this feature set for building the classification model.

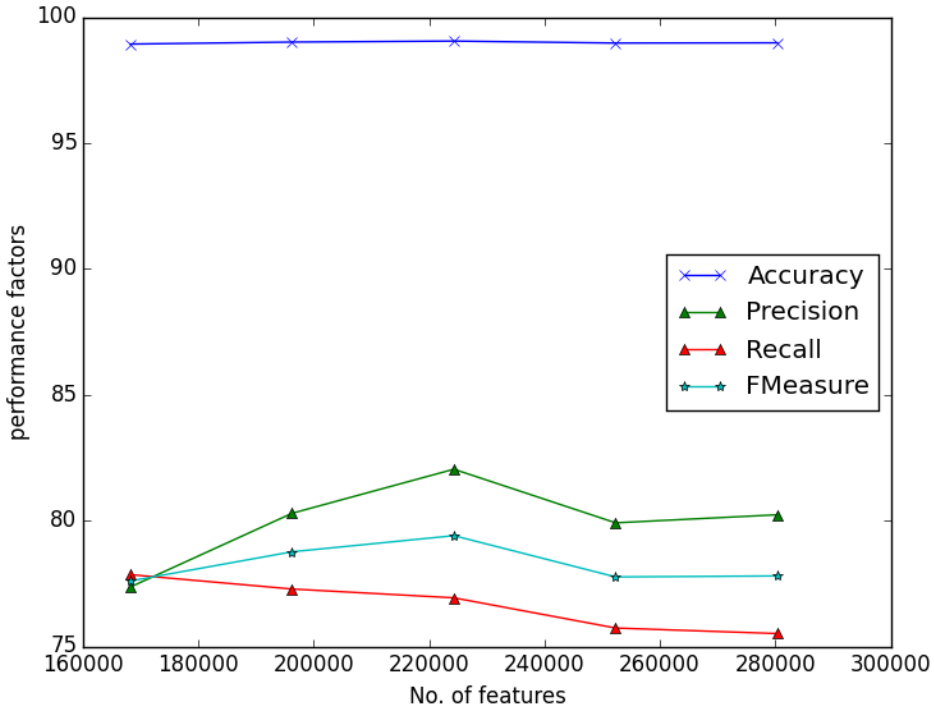


*Figure 19. Changes of performance criteria versus number of features in PAN Dataset*

### 3.5.5. Pattern Classification Results

We conducted two sets of experiments divided by the PAN and SQ datasets and evaluated the results separately.

***PAN Dataset***

In this part, we describe the achieved results and compare them with the baseline and SVM as highly standard binary classification method, which was used by the top team of PAN-2012. The training has been done via 10-fold cross validation and then the resultant model has been applied on the standard test set described in Section 4-1. First, we assess whether the one-class SVM should be trained on non-predatory or predatory conversations. In the first case, we trained the model on negative samples by filtering out the predatory samples. In this case, one-class SVM learns the distribution of non-predatory conversations. Tables 10 and 11 show the results for training the model on non-predatory and predatory conversations respectively. For a discussion on parameter optimization, please refer to the Section 3.5.6.

*Table 10. Results of training on Non-predatory samples (Experiment Train-NP-B)*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 16.63 | 100 | 28.52 |
| SVM *(regularization parameter C =10)* | 99.24 | 84.82 | 91.46 |
| One-Class SVM *(lower bound parameter nu =0.13)* | 4.35 | 32.09 | 7.66 |

*Table 11. Results of training on predatory samples (Experiment Train-P-B)*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 16.63 | 100 | 28.52 |
| SVM *(regularization parameter C =10)* | 99.24 | 84.82 | 91.46 |
| One-Class SVM *(lower bound parameter nu =0.2)* | **65.14** | **70.73** | **67.82** |

From comparing Tables 10 and 11, it can be inferred that training the model on predatory conversations yields better results. But when we apply the model on the test set, the results are not so promising (Table 12). Particularly the precision rate is low.

*Table 12. Results of testing on predatory samples (Experiment Test-P-B)*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 10.75 | 91.83 | 19.26 |
| SVM *(regularization parameter C =10)* | 75.46 | 50.43 | 60.45 |
| One-Class SVM *(lower bound parameter nu =0.2)* | **5.83** | 79.03 | 11.25 |

We believe that this behavior is due to the fact that the anomaly detection algorithms are more sensitive to noise than binary classification algorithms. As a result, we conducted a new series of experiments after doing a naïve noise removal procedure to examine the effect of noise removal on performance improvement. For our noise removal procedure, we simply omitted the conversation with just one participant. Tables 13 and 14 show the results after performing noise removal on the train and test data respectively. As we expected, even though the performance of all of the algorithms has increased after removing useless data, the noise removal procedure affects the performance of one-class SVM more significantly compared to that of other methods. Accordingly, the $F_1$-measure rises from 11% to 75%. This confirms our hypothesis about the sensitivity of one-class SVM to the noise.

*Table 13. Results of training on predatory samples after noise removal (Experiment Train-P-B-NR)*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 13.13 | 100 | 23.21 |
| SVM *(regularization parameter C =10)* | 99.92 | 95.68 | 97.75 |
| One-Class SVM *(lower bound parameter nu =0.2)* | **80.23** | **75.51** | **77.80** |

*Table 14. Results of testing on predatory samples after noise removal (Experiment Test-P-B-NR)*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 10.72 | 91.92 | 19.20 |
| SVM *(regularization parameter C =10)* | 78.13 | 50.06 | 61.02 |
| One-Class SVM *(lower bound parameter nu =0.2)* | **70.73** | **44.47** | **54.61** |

As it can be observed, one-class SVM outperforms the baseline and its performance is comparable to binary SVM. This behavior can indicate that Naïve Bayes is not suitable for handling this imbalanced dataset while SVM and One-class SVM are able to handle this characteristic of the dataset in a more decent way. Figure 20 summarizes the above results on the PAN dataset at a glance.

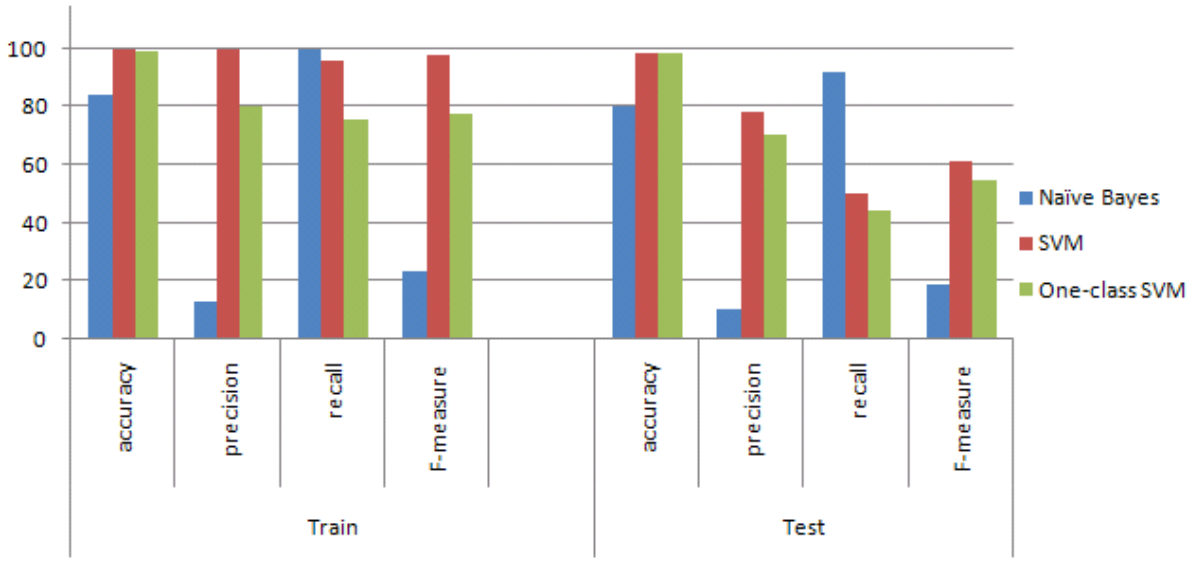


*Figure 20. Comparison of the anomaly detection approach with Naïve Bayes and SVM*

To summarize, we observed that when we added a noise removal module into the process, One-Class SVM out-performs the baseline (Naïve Bayes) and its performance is comparable with two-class SVM in this application domain.

We can also draw the following two subsidiary conclusions: Firstly, Naïve Bayes is superior with a high percentage of recall (100% on train set and 91% on test set), which implies that in terms of lower leakage rate (i.e., false negative), the baseline defeats other approaches. Secondly, SVM outperforms other methods with the highest percentage of precision (78.13%). In other words SVM has the lowest false alarm rate (i.e., false positive) among the applied methods.

***SQ Dataset***

We faced some limitations in having access to the French conversations due to the concern of preserving people's privacy. As a result, we could only access a small number of samples. We admit that since SQ dataset is too small, the results of experiments on this dataset cannot be valid for drawing any meaningful conclusion. However, these results can be used as a proof of concept or a simple test scenario for checking the validity of our hypothesis (see Chapter 1).

We trained and evaluated the system through 2-fold cross validation. It is important to note that in this set of experiments, unlike the previous setting, we considered the predatory instances as anomalies and the non-predatory instances as normal.

*Table 15. Results of training and evaluating through 2-fold cross validation on SQ dataset*

| Learning Algorithm | Precision (%) | Recall (%) | $F_1$-measure (%) |
|---|---|---|---|
| Naïve Bayes | 92.59 | 98.68 | 95.54 |
| SVM *(regularization parameter C =1)* | 92.68 | 100 | 96.20 |
| One-Class SVM *(lower bound parameter nu =0.1)* | **95.00** | **100** | **97.44** |

As we can see, the semi-supervised approach performs better than the other algorithms on SQ dataset. This can be attributed to the fact that in small datasets one-class SVM is able to capture the minimum enclosing hyperplane around the smaller set of either positive or negative instances.

## 3.5.6. Parameter Optimization Remarks

As we discussed earlier, one-class SVM needs the parameter $\boldsymbol{\nu} \in [\mathbf{0}, \mathbf{1}]$to be tuned. Although the parameter is bounded, it turns out that this parameter optimization is a challenging task for which there is no exact formal solution. In order to estimate a good value for this parameter, we used the exhaustive grid search which simply tries the entire set of combinations of parameters in a classification problem and chooses the best parameter setting based on the performance criterion (i.e., $F_1$-score). In this case, we considered $\boldsymbol{\nu}$ as the main parameters for tuning. Using a linear discretization, we chose 15 discrete points out of the interval of parameter $\boldsymbol{\nu}$ in a linear manner into 15 points: [0.66, 0.13,0.2, 0.26, …, 1]. Based on the performance evaluation, $\nu = 0.13$ in experiment setting *Train-NP-B* and $\nu = 0.2$ in experiment setting *Train-P-B-NR* revealed the best performance results. We used the same approach for estimating the value of regularization parameter in SVM binary classification. Although this approach does not necessarily lead us to the global optimum, it is a typical parameter setting approach that is quite common in practical pattern recognition tasks.

## 3.6. Concluding Remarks

We carried out a novel successful application of anomaly detection for online predator identification, which is of more use in practice compared with the current binary classification approaches that require non-predatory samples to be learned. Although as a semi-supervised technique we only used the predatory samples to train our model, as the results show, not only our approach outperformed the baseline learning algorithm (Naïve Bayes), also it is even comparable to the common binary classification algorithms on PAN dataset and outperforms the binary classification on a small dataset such as SQ.

In order to increase the performance of our model, we plan to combine the Naïve Bayes algorithm with the current model through designing an ensemble of heterogeneous classifiers. This way, we aim to also obtain the benefit of high recall rate of Naïve Bayes algorithm. Also, we plan to apply other mentioned semi-supervised anomaly detection approaches on the dataset in order to compare the performance of the method with them.

# CHAPTER 4

# DEEP LEARNING FOR OPI

Recently, deep learning has revived as a hot trend among researchers in the field of artificial intelligence and machine learning. It includes a new paradigm of learning which can mimic the behavior of human brain or visual system in an efficient way (Bengio, 2009; Schmidhuber, 2015). This new paradigm is called deep learning because there is a hierarchy of numerous layers in the main model and each layer encodes a level of abstraction in the training data. Using these models has been proven to be more efficient than the simple data mining and machine learning models. Accordingly, we apply a special architecture of Convolutional Neural Networks (a type of deep learning methods) on this application domain.

We propose an architecture based on Convolutional Neural Networks (CNNs) and apply it on PAN-2012 dataset. Based on the conducted experiments, this method provides better performance (almost 1.7% in $F_1$-measure) than the current traditional machine learning algorithms that have been applied to this domain. Furthermore, since the learning algorithm runs on general-purpose graphic cards, this approach is quite scalable. As a result, the time required for training and testing the model is comparable to that of other machine learning approaches. This chapter can be useful as a practitioners' guide in the area of applying CNN on OPI domain.

## 4.1. Hypothesis Statement

By using the appropriate deep architecture, the classifier would outperform the current algorithms based on the $F_1$-measure performance criterion. In other words, applying the proper Convolutional Neural Network (CNN) will achieve a higher performance than the standard SVM and classic Neural Network approaches.

## 4.2. Our Contribution Revisited

We propose a CNN architecture in order to improve the performance of the classification based on a widely accepted indicator for supervised learning algorithms, $F_1$-measure. Our CNN architecture convincingly outperforms Support Vector Machines and also beats the traditional neural networks). Our work is different from Dwyer's work (Dwyer, 2015) in two aspects: 1) the goal of our model is not just to identify whether a chat message is *obscene* or safe. Instead, we consider the whole conversation as a sample. 2) We do not use any pre-trained language model such as Word2Vec. Inspired by Johnson's approach (Johnson & Zhang, 2015a), we do not aim to use general pre-trained word vectors that are common in domains such as sentiment analysis and text categorization. In other words, because Word2Vec model has been trained on general web documents by Google, the model may be too general to be used in a specific domain such as OPI. Accordingly, we build the word embeddings internally in the CNN training process. To our best knowledge, this is the first time that this approach is used in OPI at the level of chat conversations.

## 4.3. Problem Definition

Let $D \subset X \times Y$ be the dataset that contains the conversations, where $X = \{\chi_1, \chi_2, \dots, \chi_n\}$ is the set of conversations so that $\chi_i = (x_1, x_2, \dots, x_m)^T$ is an $m$-dimension feature vector for $i^{th}$ conversation. Also, let $Y = \{p, np\}$ be the set of class labels in binary classification problem in which non-predatory and predatory conversations are denoted by *np* and *p* respectively. The goal is to assign the right label from *Y* to each conversation.

## 4.4. Solution: Applying CNNs

Convolutional Neural Networks can be used as a binary classifier in order to accomplish the above-mentioned task. In general, the input vector **x** is segmented into *m* region vectors $\boldsymbol{r_0(x)}$, $\boldsymbol{r_1(x)}, \dots, \boldsymbol{r_m(x)}$. There is at least one convolution layer followed by a pooling layer in a CNN. The computation units in the convolution layer are not fully connected to the input elements (unlike in original Neural Networks). This happens because the convolution layer operates on different regions of the input. The pooling layer is a sub-sampling layer that provides a higher-level abstraction of feature in each convolution layer. The most common pooling methods are max-pooling and average pooling. According to (Zhang & Wallace, 2015) max-pooling usually outperforms average-pooling for text classification.

The learning algorithm uses backpropagation in order to calculate the gradients and tries to minimize the loss function, which is usually square, logistic, or cross entropy loss. The square loss is one of the most commonly used losses and is defined as:

$$L\big(y, f(\mathrm{x})\big) = C(y - f(\mathrm{x}))^2 \tag{22}$$

in which x is the input vector, *y* denotes the actual class label assigned to the input vector, and $f(\mathrm{x})$ is the classifier output, and C is a constant normally set to 0.5 or 1. Holding the same notation, the logistic loss is defined as:

$$L\big(y, f(\mathrm{x})\big) = \log\big(1 + e^{-yf(\mathrm{x})}\big) \tag{23}$$

Finally, the Cross entropy loss is defined as:

$$L\big(y, f(\mathrm{x})\big) = -y\, ln\big(f(\mathrm{x})\big) - (1 - y)\, ln(1 - f(\mathrm{x})) \tag{24}$$

We discuss a suitable architecture of CNN which can be used in the OPI domain successfully.

### 4.4.1. Proposed CNN Architecture

In choosing the appropriate architecture for identifying the predatory conversations, the designer deals with three major decisions that we address here:

***Recurrent Neural Networks vs. CNNs***

As already mentioned in Section 2, Recurrent Neural Networks can be trained efficiently on a short window of words or on short sentences. However, when the input sequence consists of multiple sentences or even multiple paragraphs (as in OPI domain), the training of Recurrent

Neural Networks becomes intractable. Both RNNs and CNNs can be utilized for the identification of online predators depending on the ultimate goal of the analysis. However, in our use case (identification of predatory conversations), we deal with relatively long documents as our input sequence (tens of sentences in average). As a result, Recurrent Neural Networks cannot be used due to aforementioned problem, while CNNs do not suffer from this issue.

***Pre-trained word embeddings vs. internal word embeddings***

Word embeddings such as Word2Vec are pre-trained language models trained on general web documents. That can explain why using Word2Vec leads to good results in general domains, such as sentiment analysis and topic classification. However, they may not be efficient enough in domain specific usages such as OPI. That can be the reason for building the embedding internally in CNN training process and decided to not use the pre-trained word vectors. As a result, we feed the feature vectors directly to the convolution layer to learn them internally. This approach has been proposed in (Johnson & Zhang, 2015a). However, it is worth mentioning that recently, a novel embedding approach has been proposed in (Le & Mikolov, 2014) called Paragraph2Vec which results in high-quality embedding for larger chunks of text such as paragraph or even a document.

***Bag-of-words feature encoding vs. one-hot feature encoding***

If the designer decides not to use word embeddings, as in our case, s/he should consider another way of feature representation to feed the input text into the convolution layer. Basically, there are two main approaches:

*1) Bag-of-words variants*: Assuming that the number of features (words) in the corpus is denoted by *n*, a simple way is the binary representation of each region in which the presence or absence of a feature is represented by *1* and *0* respectively. As an example, let **D** be a short document containing the sequence *"r u there?"*. Then we can define 3 overlapping regions of size 2 in a way that $\boldsymbol{R_1}= [10...0100]^T$ represents 'r u', $\boldsymbol{R_2}= [00...0110]^T$ represents 'u there', and finally $\boldsymbol{R_3}= [0...0011]^T$ represents 'there ?'. As observed, in each region vector there are two *1*s and all the remaining n-2 features are 0. Of course, one can also use the normalized frequency of words instead of only 0 and 1. One of the main drawbacks of this approach is the fact that it does not count the order of words at all.

*2) Sequential concatenation of one-hot vectors:* This encoding method was introduced by Johnson and Zhang (2015a) and (2015b) in order to take the words order into account: In this approach, the region sequences are concatenated to form the feature vector representing a document. Considering the document **D** again, the three overlapping regions *'r u'*, *'u there'* and *'there ?'* would be represented by $\boldsymbol{R_1}=[10...0|010...0]^T$, $\boldsymbol{R_2}=[010...0|0...10]^T$, and $\boldsymbol{R_3}=[0...10|0...01]^T$ respectively. As can be seen, each vector has two parts separated by a pipe (for the sake of visualization). Each part corresponds to one token in the region and contains only a single *1* for that token in the whole vocabulary. For example, the first part of $\boldsymbol{R_1}$ represents token 'r'. The drawback of this approach is that it makes the feature space extremely sparse, but according to (Johnson & Zhang, 2015a) if the implementation leverages the sparse matrix vector calculations as in (Johnson, 2016), it can lead to better classification results in some cases.

However, after trying both approaches, we observed that in our case the best results were obtained by utilizing the bag-of-words approach (c.f. Section 4.5.4).

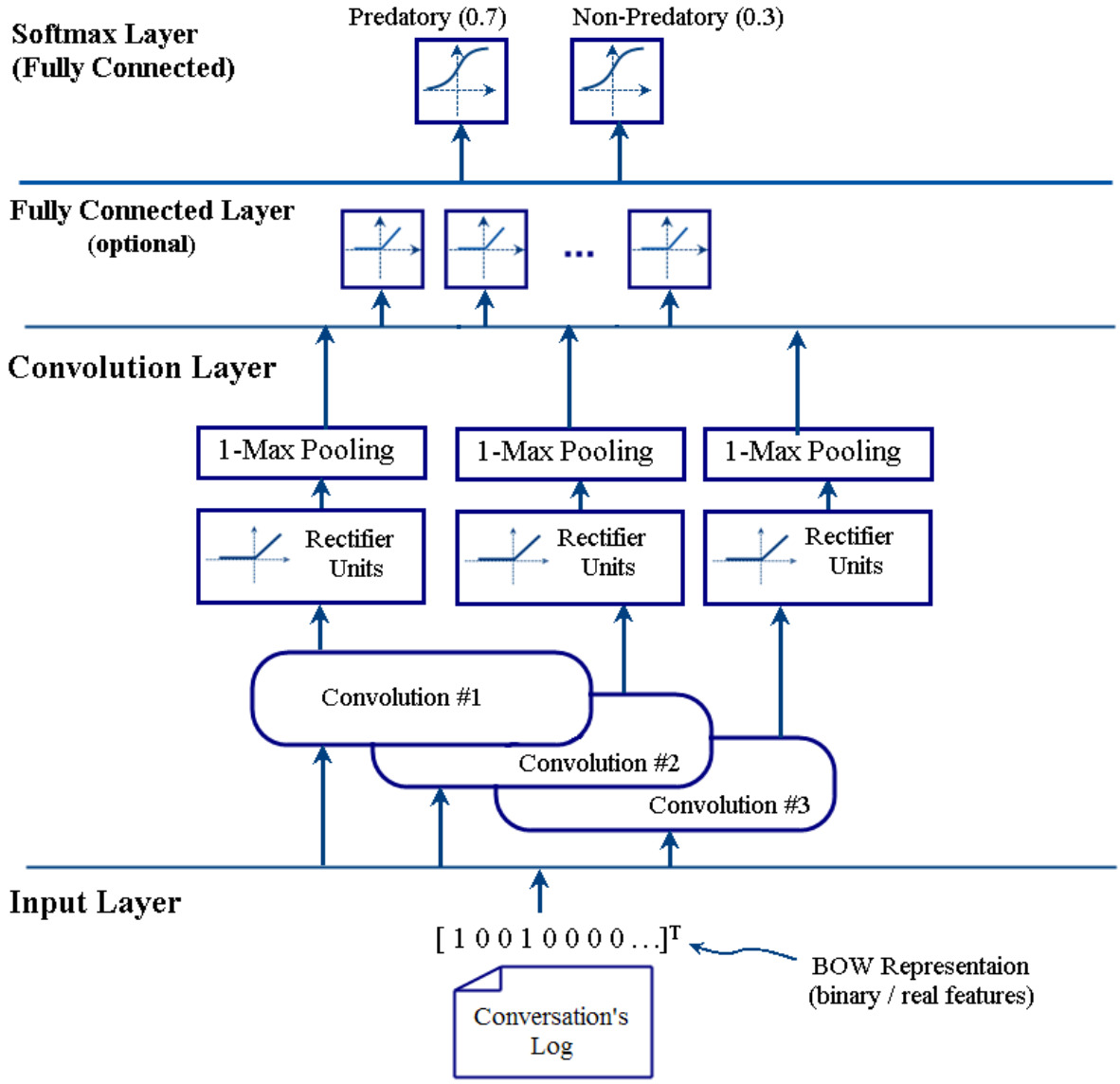


*Figure 21. The proposed CNN Architecture used for OPI*

After deciding about the topology of the CNN we can proceed to setting the hyper parameters. We describe the hyper parameters' settings and the resultant outcome in Section 4.5.

## 4.5. Experiments

This section covers the conducted experiments as well as their corresponding settings and the characteristics of the dataset.

### 4.5.1. Environmental Settings

Due to the parallel nature of the neural-network-based learning models, they are mostly efficient when the parallelism is implemented at the hardware level by using graphical processing units. Hence, we run the processes on Calcul Québec, a high-performance computing cluster in Canada (Calcul Québec, 2016) which has several NVIDIA Tesla K80

GPUs. We ran all the processes on one K80 GPU with 24 GB of memory and 2496 processor cores. In fact, the architectures discussed in the next section cannot be run on a CPU.

### 4.5.2. Dataset

We conducted the experiments on the PAN-2012 dataset (Inches & Crestani, 2012), which was described in chapter 3. The predatory instances in this dataset have been gathered by a non-profit organization called *Perverted Justice* (http://www.perverted-justice.com). These conversations occurred between experts who posed as juveniles and convicted prolific online predators. The dataset has been used extensively in the literature (Kontostathiset al. 2010; Mcghee et al., 2011; Pendar, 2007). The extended dataset has been used in PAN-2012 international competition as a reference dataset for the task of recognizing the predatoriness of either messages or users. See Table 6 for the characteristics of this dataset.

The dataset is formatted in two XML files, one for a training set and another one for a testing set. Since we are doing the analysis at the conversation level, the XML files need to be parsed in order to extract the conversations. Each conversation contains the messages of each participant as shown in Figure 22.

```
<conversation id="8ff4c51529c81dabb0978206cb6bf06a">
   …
<message line="4">
<author>f4113d73c0b80c35c5e085e01f736ab4</author>
<time>12:34</time>
<text>u didn't talk 2 me yesterday</text>
</message>
<message line="5">
<author>47243a4a2c68f2f00899670d455a21fa</author>
<time>12:34</time>
<text>I wasn't on</text>
</message>
<message line="6">
<author>47243a4a2c68f2f00899670d455a21fa</author>
<time>12:34</time>
<text>Sorwy</text>
</message>
<message line="7">
<author>47243a4a2c68f2f00899670d455a21fa</author>
<time>12:35</time>
<text>I got urmsgthoe..</text>
</message>
<message line="8">
<author>f4113d73c0b80c35c5e085e01f736ab4</author>
<time>12:36</time>
<text>what r u doing?</text>
</message>
<message line="9">
<author>47243a4a2c68f2f00899670d455a21fa</author>
<time>12:37</time>
<text>Workn..</text>
</message>
…
</conversation>
```

*Figure 22. A sample snippet of a conversation*

The dataset has been originally labeled with predators or non-predators. Hence, in order to identify the predatory conversations, we had to re-label the data in a way that if at least one predator participates in a conversation, the conversation will be labeled as predatory. Since almost all of the predatory conversations have taken place between only two participants this is a reasonable way of labeling the data.

We used the open source code sparse implementation of CNN provided by Rie Johnson available at "riejohnson.com/cnn_download.html" for our experiments.

### 4.5.3. Experiments' Settings

Setting the optimal hyper parameters of a CNN is a challenging task that requires more research. Most researchers find sub-optimal choices of these parameters by trying different combinations in their corresponding domain. We classify these parameters separately and describe our choice for each as follows:

- ***Regularization parameter and dropout rate:*** These two important parameters are mainly used to prevent overfitting. L2 regularization (Ng, 2004) is the most common type of regularization used in neural networks. It is worth mentioning that the regularizations in a CNN are usually done at the top layer. Another important mechanism to prevent overfitting is the dropout that randomly deactivates a certain number of output units in the top-layer in the training process. We set the dropout rate and the coefficient of L2 regularization to be 0.5 and $10^{-4}$, respectively that are the default settings in ConText2.0 and we found it efficient.
- ***Loss function:*** We used the square loss function since it outperformed other loss functions, including logistic loss.
- ***Activation function:*** The Rectifier function was used as the activation function of convolution layer since it led to better results compared to *tanh* or *sigmoid* function.

As a side note, we did not do any preprocessing other than converting the upper case letters into lower case. The reason is the informal and colloquial nature of the chat conversations. In fact, performing the simplest preprocessing step such as stop-word removal may damage the meaning of the message. To make it more concrete, this can be seen in the message ***"i thought u wanted 2 come c me"***. While a blind noise removal procedure may omit tokens such as ***c*** in this sentence, we know that it bears important meaning.
We conducted two series of experiments on each of the above-mentioned datasets and we describe them in the following sub-sections:

### 4.5.4. Investigating the effect of convolution

In order to investigate the effect of convolution in this domain, we conducted a set of experiments in which we compare the performance of a traditional neural network with one hidden layer, to that of a CNN with one convolution layer. In order to study the effect of the convolution per se, we used a fixed number of computation units (2000) with rectifier activation functions in the hidden layer and convolution layer of NN and CNN respectively.
We compare the performance of the convolution in a CNN with depth 1 (i.e., one convolution layer) with that of the traditional neural network (NN) and baseline (SVM with linear kernel).

*Table 16. PAN-2012 dataset: Performance comparison for depth-1 CNN with baselines (Support Vector Machines (SVM) and traditional neural network (NN))*

| **Learning Scheme** | **Exp. No.** | **Settings** | **Precision** (%) | **Recall** (%) | **$F_1$-score** (%) |
|---|---|---|---|---|---|
| SVM | 1 | *linear kernel* | 78.13 | 50.06 | 61.02 |
| NN | 2 | *depth: 1, nodes: 2000, encoding: binary-encoded unigram, vocab. size=5000* | 91.49 | 69.97 | 79.30 |
| | 3 | *depth: 1, nodes: 2000, encoding: frequency of unigram, vocab. size=5000* | 91.58 | 70.40 | 79.61 |
| | 4 | *depth: 1, nodes:2000, encoding: frequency of bigram, vocab. size=7000* | 90.34 | 71.72 | 79.96 |
| CNN | 5 | *depth: 1, nodes:2000, region size:(1,2 and 3), encoding: concatenation of one-hot vectors, vocab. size=5000, pooling type: max* | 91.44 | 71.56 | 80.29 |
| | 6 | *depth: 1, nodes:2000, region size:15, encoding: binary-encoded unigram, vocab. size=5000, pooling type: max* | 91.57 | 73.65 | **81.64** |

*'Vocab. size'* is the maximum size of the vocabulary extracted from the corpus, '*binary-encoded uni/bigram'* and *'frequency of uni/bigram'* were explained as *bag-of-words feature encoding* in section 3-2-1. Similarly, *concatenation of one-hot vectors* refers to *one-hot feature encoding* described in Section 3-2-1.

As can be seen in the Table 16, unlike Johnson and Zhang's work (2015a), the methods with bag-of-words features outperform the sequential concatenation of region vectors. Also the frequency representation of words (tokens) wins over the binary vector representation in our case. Figure 23 shows the changes of training and testing errors for the outperforming approach (experiments No. 6).

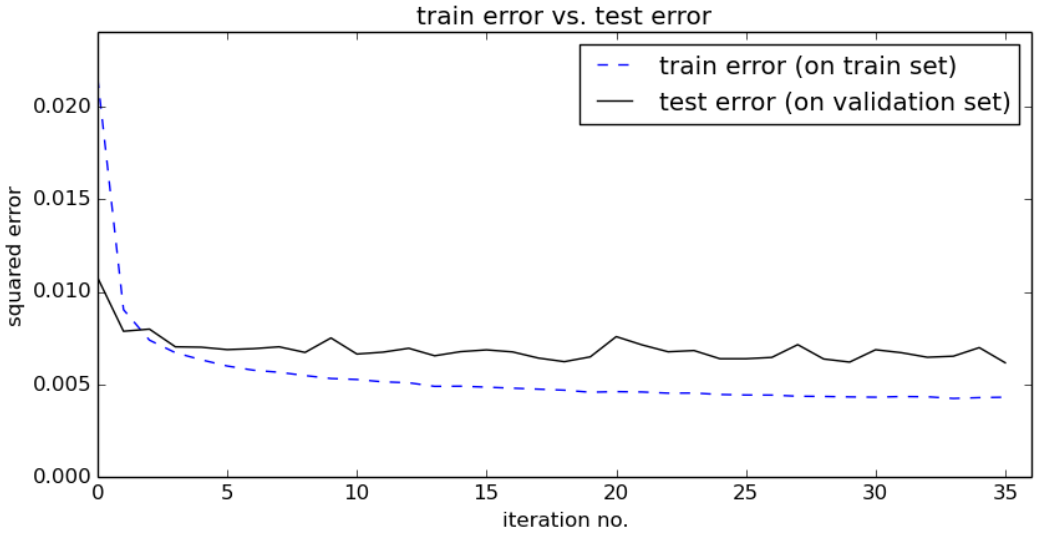


*Figure 23. Train and Test errors for 36 iterations of CNN with one convolution layer and real-valued bag-of-words features (experiment No.6). As seen in the figure, after iteration 18, the test error begins to fluctuate and does not continue to reduce.*

### 4.5.5. Adding Extra Convolution Layers

To investigate the effect of adding extra convolution layers to a CNN architecture and comparing the same effect on a traditional NN, we conducted comparative experiments in which the performance of two CNN architectures, one with a single convolution layer and the other one with two convolution layer, is compared with the performance of two original NNs with one and two hidden layers each. The results are shown in the next paragraphs.

Next, we investigate the effect of depth of the architecture. Figure 24 compares the precision recall curves for traditional NN and CNN with one and 2 hidden/convolution layers.

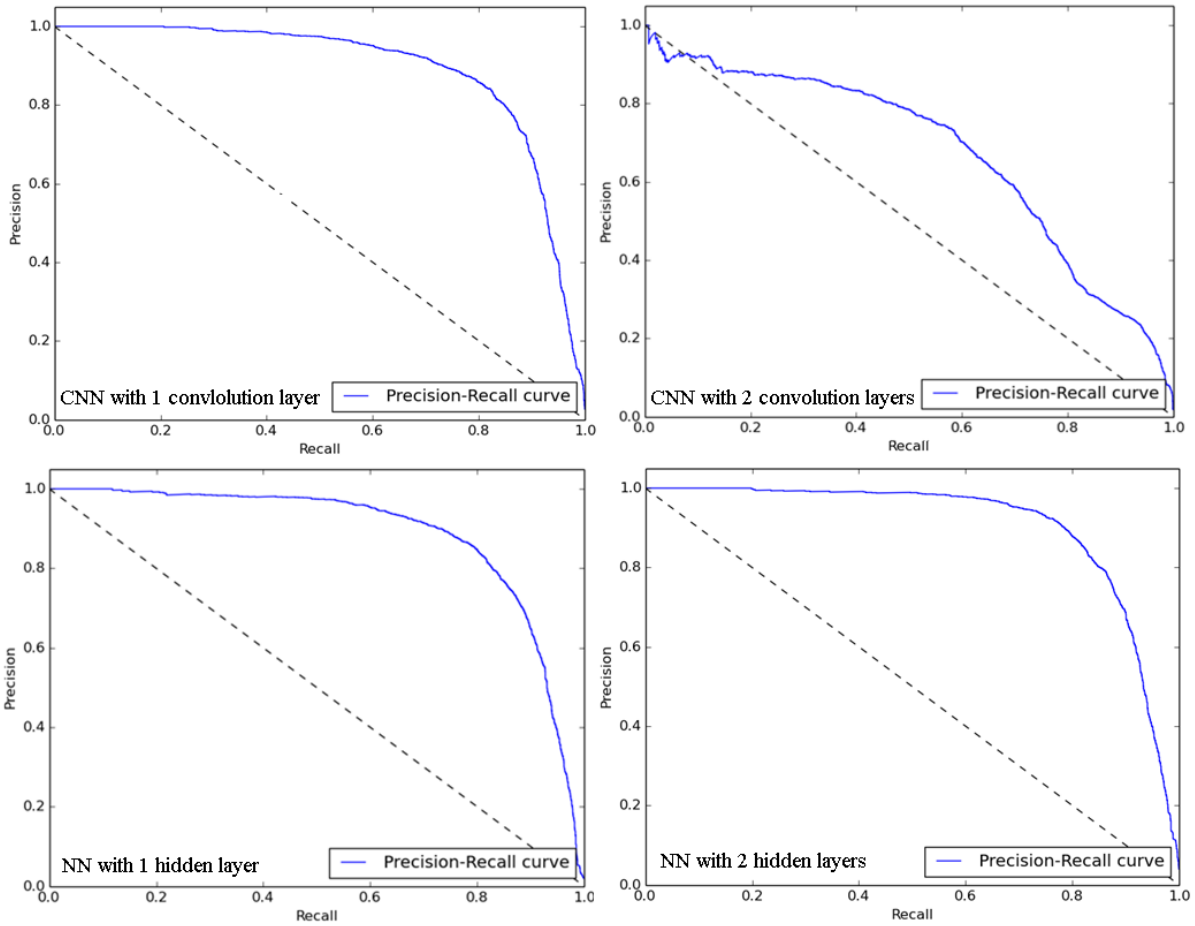


*Figure 24. Precision-Recall curves for showing the effect of extra convolution/hidden layers on CNN and NN. The performance of the CNN with two convolution layers has decreased, while that of the NN has increased.*

As seen in Figure24, the performance of the CNN with two convolution layers is lower than that of a CNN with a single convolution layer. On the contrary, the performance of the traditional NN increases with adding an extra hidden layer.

This raises the question that whether having a deep CNN would perform better than a CNN with only one single convolution layer in text classification. In our opinion, although our experiments favor upon having only one convolution layer, this might happen due to the efficiency of backpropagation method when used in a CNN with more than one convolution layer.

## 4.6. Discussion and Concluding Remarks

We showed that using one layer of convolution has a positive effect on classification accuracy. Even though one might benefit from having several convolution layers (i.e., a deeper structure) in image processing usages, according to our experiments, in natural language processing use cases, it might not be the case as the number of layers in the hierarchy increases the training algorithm (more specifically the backpropagation algorithm used in CNN) becomes inefficient. As a result, in spite of the fact that we tested countless combinations of architectures with two or more convolution layers and even ran more

iterations of the algorithm, the best performance was obtained by having only one single convolution layer in the architecture. Another important point to consider is the fact that the massive parallelism of the GPU allowed us to utilize relatively large number of neurons (several thousands) in both NN and CNN experiments. Whereas the traditional neural classifiers that run on the CPU can technically utilize a much lower number of neurons and have poorer performance (close to that of SVM) than what we obtained for NN experiments in this research. Therefore, one of the benefits of this research is the application of GPU to this domain of application.

We think it is also worth mentioning some of the best practices that we learned throughout our experiments. Although they are not claimed as being always true, they might be of help for the other researchers in this specific field.

- Unlike traditional approaches, we did not perform any preprocessing procedure other than changing the uppercase letters to lower case. We experienced that doing procedures such as stop-word removal, and removing certain numbers or symbols (e.g., *"?"* and *"!"*), decreases the classification performance.
- Rectifier activation function outperformed other activation functions including (*tanh* and *sigmoid*)
- Having only one convolution layer led to better results compared to deeper structures with more than one convolution layer (Figure 24).
- Using normalized frequencies instead of One-hot vector led to better results.
- Another interesting observation was that decreasing the step size after a certain number of iterations is usually helpful.

# CHAPTER 5

# RESOLUTE SOFTWARE ARCHITECTURE

This chapter briefly covers the software engineering aspects of the designed and implemented software prototype for being used by Sûreté du Québec as part of the Resolute project. In fact, this software tool served as a proof of concept for Chapter 3. The Resolute prototype is an standalone application written in Java. It uses Java swing for the user interface.

## 5.1. User-level Goals

The prototype has been mainly designed and implemented as a proof of concept with the goal of saving the investigator's time and reducing the burden of automatic identification of predators in chat logs. The ultimate goal of the software is finding the most likely predatory conversations in order to reduce the size of the search space in which an investigator needs to search. For sure, the final decision should be made but the human investigator and the decision produced by the tool is not meant to be a basis for judgment.

## 5.2. Software Design

### 5.2.1. Data Flow

First, we will describe the data flow of the Resolute tool to obtain a primary insight about the sequence of operations done in the prototype. The process starts by scanning a document repository (in this case a folder on local disk) and traversing each *.docx* or .txt file and extract the texts. Then normalization is done (converting upper case to lower case and removing noises at the character level), then noisy documents are removed and the resultant subset would be tokenized. Then the bigram language model is built which is a special type of vector space model and the classification or sentiment analysis can be done afterwards.

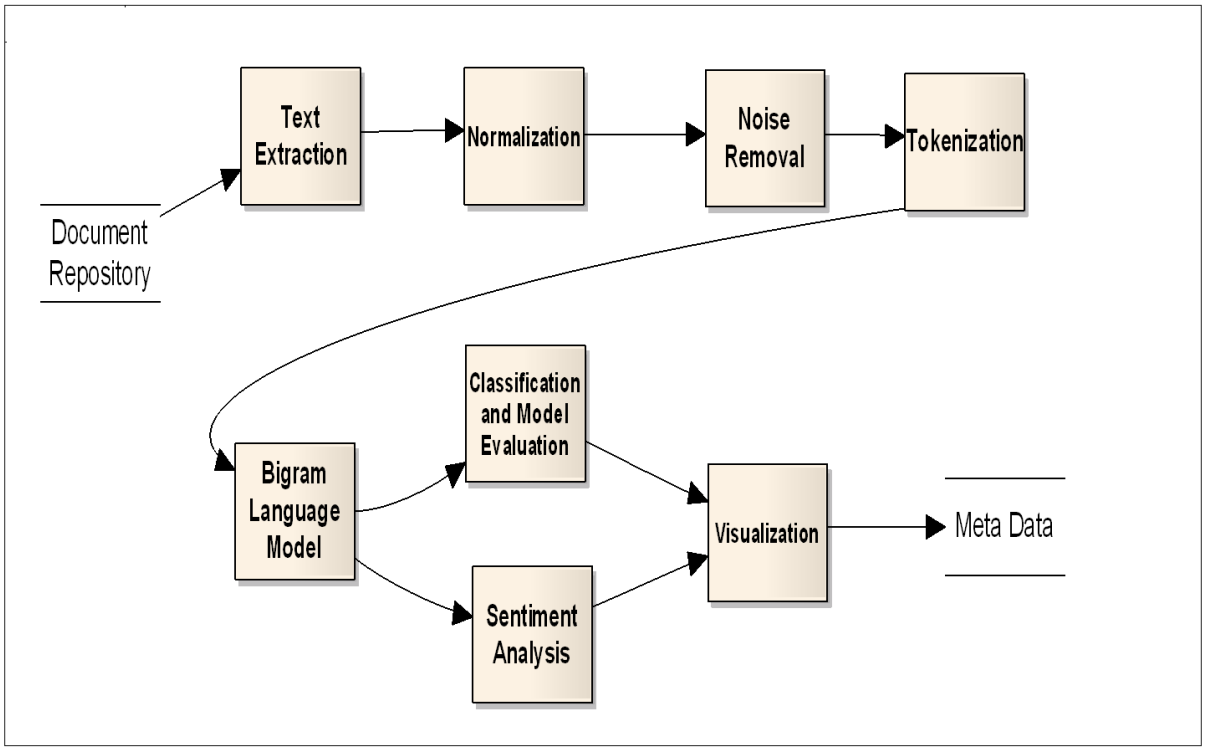


*Figure 25. The data flow of the implemented prototype*

### 5.2.2. General Architecture

Figure 25 shows the general architecture of the software. As a prototype (and also a proof of concept), the predator identification part was implemented using Rapidminer classification libraries. The implemented prototype works on the offline data stored on the local storage, but as seen in the diagram it is extendable to import data from social media as well.

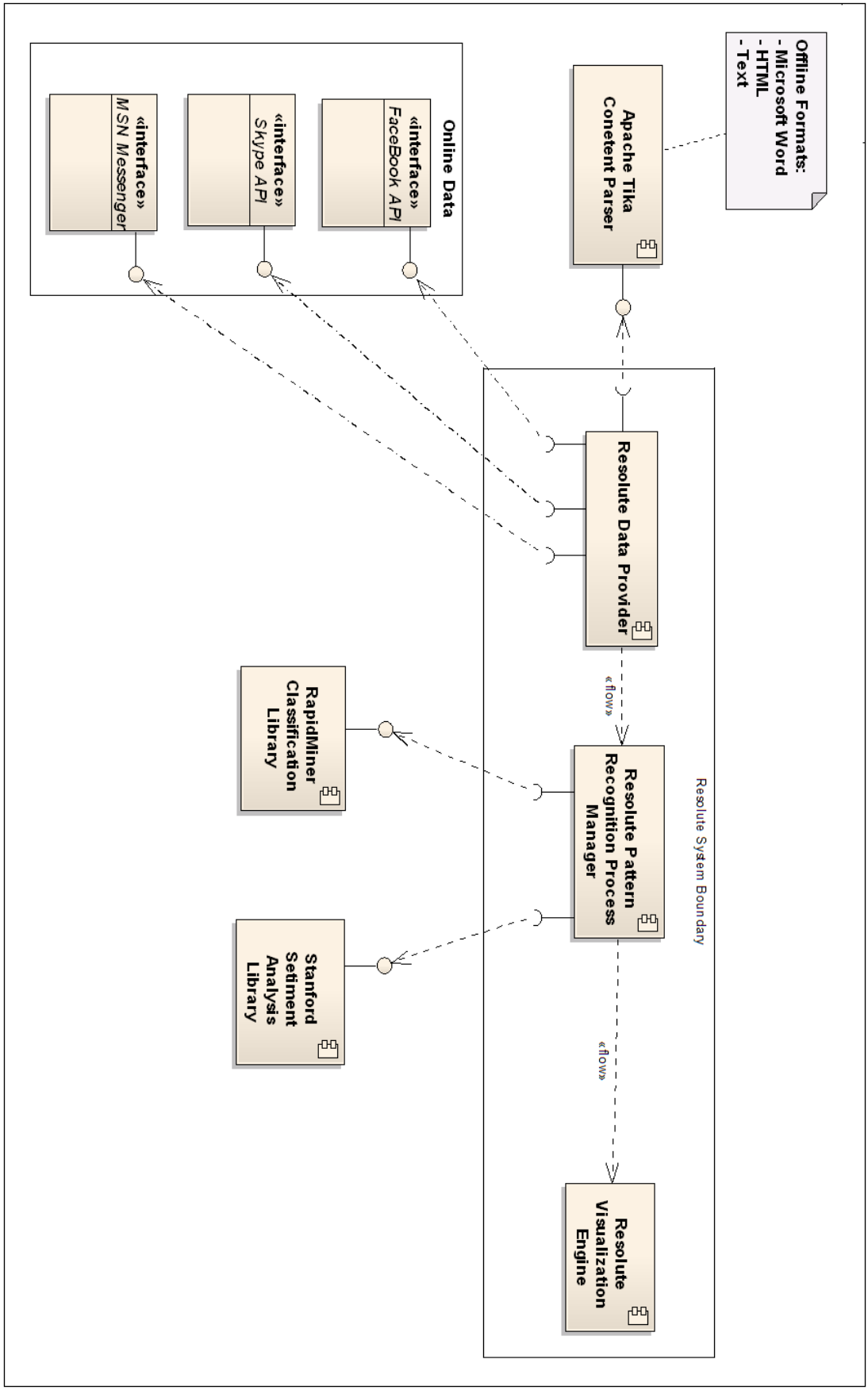


*Figure 26. Abstract architectural design of Resolute*

### 5.2.3. Design Class Diagram

Figure 27 depicts the design class diagram for the software.

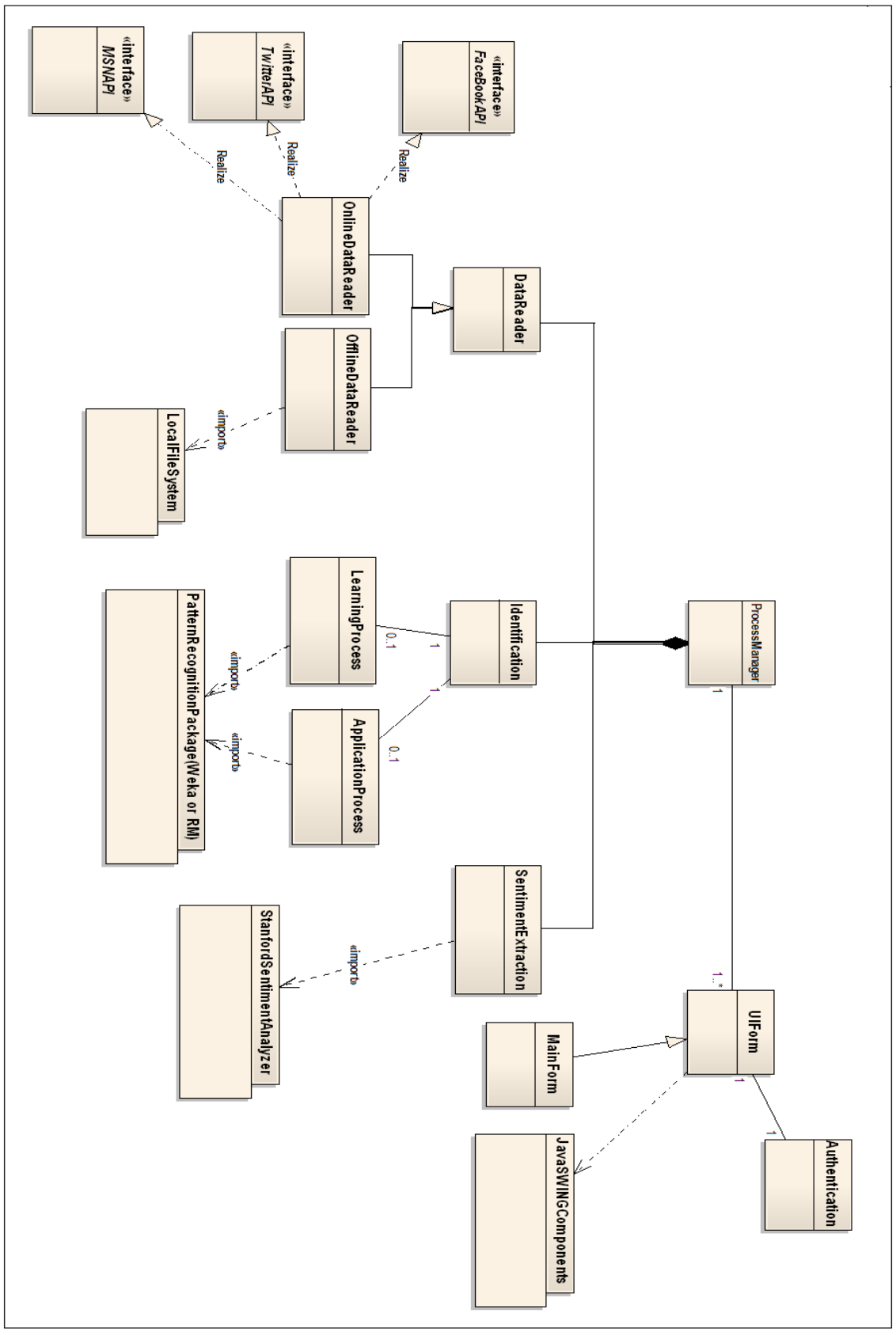


*Figure 27. Design Class Diagram (DCD) of the Resolute prototype*

## 5.3. User Interface

Figures 28 and 29 illustrate the snapshots of the user interface of the Resolute Prototype powered by Java Swing. The first snapshot shows the home page of the tool in which the user can train the classifier in two modes: 1) Regular (i.e., binary classification) and 2) Anomaly detection mode. The regular mode uses the classic SVM. The anomaly detection mode applies One-class SVM. All of the preprocessing steps mentioned in the above data flow diagram are automatically done when the user clicks on “start training button”.

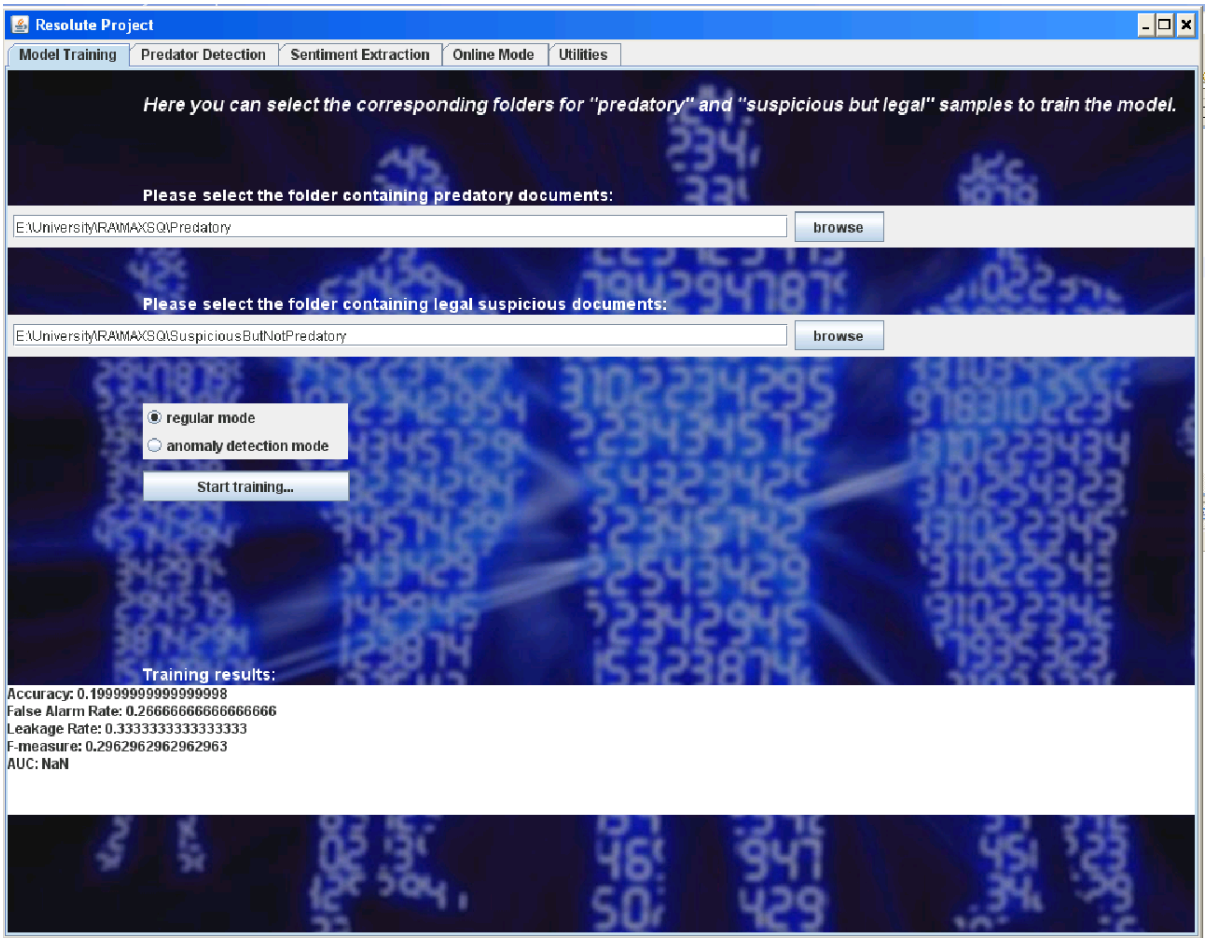


*Figure 28. The prototype’s graphical user interface for model training*

Figure 29 depicts the user interface in which the user is able to see the results of classification for some unresolved chat documents.

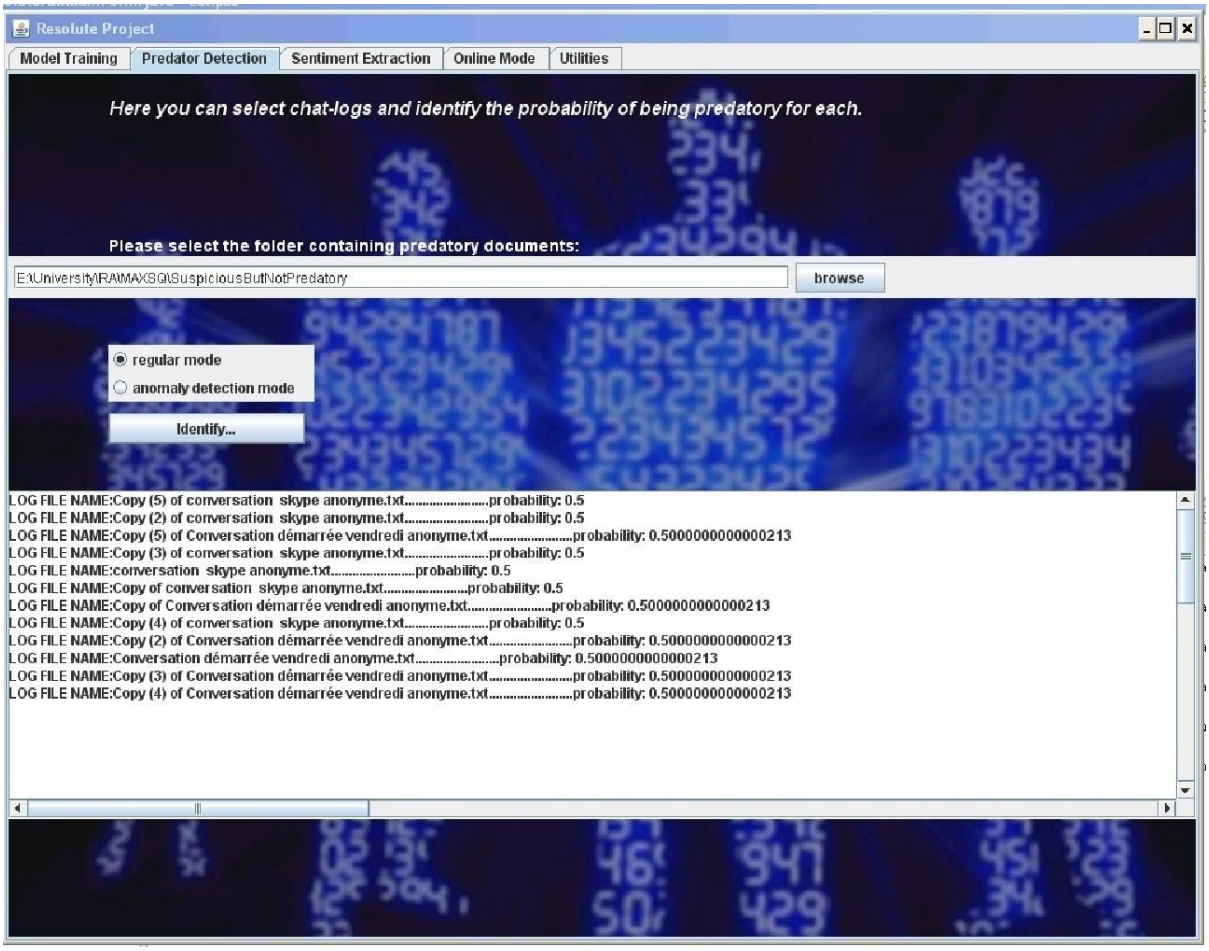


*Figure 29. The prototype's graphic user interface for applying the model on unsolved samples*

## 5.4. List of Features

The feature list of the prototype is provided as well as their corresponding requirement and use case scenarios:

- **Feature 1: Predicting predatory and non-predatory conversations**
  - ***User level Requirement:*** System shall provide the user with a prediction about an input conversation
  - ***Use case scenario 1-1:*** User chooses to train the system. Then S/he Identifies the paths to positive samples as well as negative samples and chooses one of the training algorithms (binary classification or anomaly detection). Afterwards, s/he triggers the training process. System notifies the user when the training process is finished.
  - ***Use case scenario 1-2:*** User specifies the path to the folder that contains the unknown conversations and triggers the prediction process. System notifies the result of prediction for each given sample.
- **Feature 2: Providing the confidence value for predictions**
  - ***User level Requirement:*** System shall provide the user with a real-valued confidence level assigned to each given unknown sample

***Use case scenario 2:*** Once the system finishes the prediction process, it shall notify the user with corresponding confidence value. This value shall be provided as a number between 0 and 1 which translates as the probability of being predatory.

- **Feature 3: Anonymizing the input text**
  - ***User level Requirement:*** System shall remove the specific user identifiers from the input conversations with predetermined formats
  - ***Use case scenario 3:*** User specifies the path to the folder that needs to be anonymized. The folder should be in one of the predetermined formats that discussed with Sûreté du Québec and known as *Facebook* and *Skype* format. Then user triggers the anonymizing process. System notifies the user when the anonymization is finished.

# CHAPTER 6

# CONCLUSION

Rapidly growing prevalence of online communications in juveniles' daily lives makes it vital to leverage data mining techniques for automatic identification of online predators. Automated investigation of chat logs is one of the most proactive and effective approaches that can be used to avoid the consequences of this sort of crime. The most popular preprocessing techniques including noise removal, feature selection, and dimensionality reduction were introduced. Also, different aspects of suitable feature extraction procedure for this problem domain were discussed and finally the most common data mining classification algorithms which are frequently used in OPI were introduced.

The semi-supervised anomaly detection method used in Chapter 3 led to obtaining acceptable performance in the absence of one of the class labels in training process both for PAN and SQ datasets. Finally, the Convolutional Neural Network that was used in supervised setting, pushed the performance by almost 1.7% compared to the best commonly-used classification algorithm in this domain. Finally, Chapter 3 was implemented as a java tool that can identify predatory conversations using both anomaly detection and simple SVM binary classification.

## 6.1. Summary of Research Activities

Table 17 shows the major activities carried on during this thesis in chronological order of execution.

*Table 17. The major research activities in chronological order*

| No. | Activity Description |
|---|---|
| 1. | Identifying the prominent researchers and private companies across Canada and US who had hands-on experience in the area of online predator identification and mining chat-logs. |
| 2. | Authoring and publishing a book chapter titled "Automated Identification of Child Abuse in Chat Rooms by Using Data Mining" in book "Data Mining Trends and Applications in Criminal Science and Investigations" |
| 3. | Applying anomaly detection via implementing a semi-supervised approach based on One-class SVM and applying it on a publicly available dataset (chapter 3). |
| 4. | Implementing a software prototype in Java called Resolute as a demo to Sûreté du Quebec. |
| 5. | Authoring and publishing a conference paper titled "Recognizing Predatory Chat Documents using Semi-supervised Anomaly Detection" in 23rd Document Recognition and Retrieval conference in San Francisco, US. |
| 6. | Applying Deep Learning architectures to this problem domain via Convolutional Neural Networks |
| 7. | Authoring the paper titled "Using Deep Learning for Online Predator Identification: A practical approach" to be submitted to Elsevier's Journal of Digital Investigations |
| 8. | Writing up and revising the thesis document |

## 6.2. Research questions and objectives revisited

As described in Chapters 3, 4 and 5, the following objectives were accomplished:

- In Chapter 3, a semi-supervised anomaly detection approach, which can be trained by only one class label, was applied to the application domain and it was shown that the model's performance is comparable with that of binary classification that use both positive and negative samples for training. This result supports the hypothesis 1 stated in chapter 1.
- In Chapter 4, a deep learning approach, which can deal with relatively large documents, were applied to the application domain. Leveraging the proposed architecture of Convolutional Neural Network, it revealed almost 1.7% improvement in classification performance that supports the hypothesis 2 mentioned in Chapter 1.
- In Chapter 5, the model described in Chapter 3 as well as the SVM binary classification method was implemented as an independent utility software prototype.

## 6.3. Future Research Directions

In spite of the achievements mentioned in this chapter, there are still important challenges that researchers need to tackle in the field of OPI. We predict that the future of this research line in the next decade will spin around Social Network Analysis featured by deeper linguistic analysis to understand the semantics of messages. Accordingly, we describe the potential future research lines based on our anticipation of the problem domain. In the following, we demonstrate the necessity for deeper linguistic analysis. The related challenges are discussed and then the field of Web-based Dynamic Social Networks and Recurrent Neural Networks are introduced.

### 6.3.1. Performing Deeper Linguistic Analysis on Chat logs

Mining chat logs is strongly correlated to challenging problems in the domain of NLP including Word Sense Disambiguation (identifying the sense for a polysemic part of speech), Discourse Analysis (Discovering the conversation concepts and psychological characteristics of the writer), and also Named Entity Recognition (Extraction of role-playing entities such as locations, people and organizations).

On top of these linguistic challenges, there is another important issue related to the nature of chat logs: Conversational (i.e., non-official) writing style of participants. Consider the following predatory conversation:

*<text>i'm bored</text>*
*<text>Awww babe</text>*
*<text>I'm sorwy</text>*
*<text>where u at</text>*
*<text>Vegas</text>*
*<text>5-6 hours away</text>*
*<text>dude y cant u come then!?</text>*
*<text>I'm n vegas lol</text>*
*<text>I'm n another state</text>*
*<text>I'm not ncalifornia</text>*
*<text>i thought u wanted 2 come c me</text>*
*<text>I do</text>*
*<text>But how can I went I'm n another state</text>*
*<text>when do u leave?</text>*
*<text>Dis mornin</text>*
*<text>well i guess u aint really my bf then cuz u lied</text>*

This writing style requires some additional considerations that make it different from normal text mining. A common issue that arises in such a context is the existence of non-grammatical sentences that makes the typical parsing algorithms inefficient. For instance, '*But how can I went I'm n another state'*. Another issue is regarding the use of drastically misspelled words such as in '*Dis mornin'*. Even using the sophisticated spell checkers or stemmers on such a data as a preprocessing phase would not be so efficient. Having too many different forms of writing for a single word causes the problem that is known as 'curse of dimensionality'. This makes the learning algorithms significantly inefficient.

Another important issue that might not be so related to the linguistic aspects of OPI is the imbalanced nature of chat logs data. This means usually there are too many non-predatory instances compared to predatory ones. This problem makes the learning process more challenging since it requires specific algorithms to deal with this type of imbalanced data.

### 6.3.2. Learning Deep Architectures

As discussed in chapter 2, Recurrent Neural Networks have been shown to be efficient in text analysis and other NLP tasks, such as speech recognition (please refer to chapter 2 for an introduction of Recurrent Neural Networks).The only challenge with them is the difficulty of the training process that makes them not so efficient in dealing with large documents. We anticipate that new approaches will emerge to tackle this challenge. Accordingly, we anticipate that these models will be utilized extensively in the field of OPI in the near future.
As another interesting area, we plan to test the embedding method called Paragraph2Vec proposed in (Le & Mikolov, 2014) to verify classification results using this new embedding technique.

### 6.3.3. Web-based Dynamic Social Networks

Criminal social network analysis and visualization was briefly mentioned previously. Unlike the traditional criminal networks that have strictly hierarchical structures, online pedophile networks naturally have cellular and distributed structures and usually do not have obvious leaders. These special types of networks demand the usage of approaches specifically designed for tackling with the cellular distributed crime networks. These approaches should be able to analyze smaller crime networks that do not necessarily have a specific powerful leader. A new tool for analyzing this sort of networks has been developed by Carley (2015) at Carnegie Mellon University, which might be useful for analyzing pedophile covert networks. In addition to the approaches identified by Klerks in section 1, a new branch of social network analysis called Web-based Dynamic Social Network has been revealed recently to address the mentioned requirement. In this point of view, WDSN differs from traditional social networks in the sense that they are cellular, distributed, web-based, dynamic, and may contain varying levels of uncertainty. According to Berger-Wolf and Saia (2006), dynamic network analysis enables probabilistic reasoning about changes in dynamic networked web-based communities and how such networks evolve, adapt to changes, and how they can be destabilized. Leveraging WDSN to identify pedophile covert networks and analyze their evolving network communication structure can be considered as one of the most significant directions in the field.
According to the above-mentioned challenges, we anticipate the future of this field spin around the following issues:

- Achieving a deeper understanding of text, or more generally natural language, to uncover the semantics behind the chat logs and improve the accuracy of classification models.
- Leveraging deep learning as a new trend in artificial intelligence for building more sophisticated language models from chat logs.

- Using the concepts of WDSN introduced above to identify pedophile covert networks and analyze their evolving network communication structure.

# REFERENCES


1. Aggarwal, C. C. (2015). Mining Text Data. In *Data Mining* (pp. 429–455). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-14142-8_13
2. Amer, M., Goldstein, M., Slim, A., & Abdennadher, S. (2013). Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection, Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description (pp. 8–15). Chicago, IL, USA,: ACM.
3. Anantharam, P., Thirunarayan, K., & Sheth, A. (2012). Topical anomaly detection from Twitter stream. In *4th Annual ACM Web Science Conference (WebSci '12)* (pp. 11–14). New York, NY, USA: ACM.
4. Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127. http://doi.org/10.1561/2200000006
5. Berger, A. L., Pietra, V. J. D., & Pietra, S. A. D. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, *22*(1), 39–71.
6. Berger-Wolf, T. Y., & Saia, J. (2006). A Framework for Analysis of Dynamic Social Networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 523–528). New York, NY, USA: ACM.
7. Bishop, C. (2006). *Pattern Recognition and Machine Learning*. New York: Springer-Verlag.
8. Bogdanova, D., Rosso, P., & Solorio, T. (2012a). Modelling Fixated Discourse in Chats with Cyberpedophiles. In *Proceedings of the Workshop on Computational Approaches to Deception Detection* (pp. 86–90). Association for Computational Linguistics.
9. Bogdanova, D., Rosso, P., & Solorio, T. (2012b). On the Impact of Sentiment and Emotion Based Features in Detecting Online Sexual Predators. In *Proceedings of the 3rd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis* (pp. 110–118). Association for Computational Linguistics.
10. Bogdanova, D., Rosso, P., & Solorio, T. (2014). Exploring high-level features for detecting cyberpedophilia. *Computer Speech & Language*, *28*(1), 108 – 120. http://doi.org/http://dx.doi.org/10.1016/j.csl.2013.04.007
11. Calcul Quebec. (2016). Retrieved January 20, 2016, from http://www.calculquebec.ca/en/
12. Cano, A., Fernandez, M., & Alani, H. (2014). Detecting Child Grooming Behaviour Patterns on Social Media. In L. Aiello & D. McFarland (Eds.), *Social Informatics* (Vol. 8851, pp. 412–427). Springer International Publishing.
13. Carley, K. M. (2015, May 8). DyNet. Retrieved from http://www.casos.cs.cmu.edu/projects/DyNet/dynet_info.html
14. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), Article No. 15.
15. Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 27:1–27:27.
16. Chapelle, O., Chi, M., & Zien, A. (2006). A Continuation Method for Semi-supervised SVMs. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 185–192). New York, NY, USA: ACM.

17. Chinchor, N. (1992). MUC-4 Evaluation Metrics. In *Proceedings of the 4th Conference on Message Understanding* (pp. 22–29). Stroudsburg, PA, USA: Association for Computational Linguistics.
18. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. P. (2011). Natural Language Processing (almost) from Scratch. *JMLR*, *12*, 2493–2537.
19. Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern Classification* (second ed.). Wiley-Interscience.
20. Dwyer, K. (2015). *Word Representation Learning for Detecting Malicious Chat Messages*. University of Alberta. Retrieved from https://www.mitacs.ca/en/projects/word-representation-learning-detecting-malicious-chat-messages
21. Ebrahimi, M., Suen, C. Y., Ormandjieva, O., & Krzyzak, A. (2016). Recognizing Predatory Chat Documents using Semi-supervised Anomaly Detection. In Proceedings of *Document Recognition and Retrieval XXIII* (pp. 1-9), San Francisco, CA, USA: Electronics and Imaging 2016.
22. Eriksson, G., & Karlgren, J. (2012). Features for modelling characteristics of conversations. *Notebook for PAN at CLEF 2012*, Rome, Italy: CLEF 2012.
23. Escalante, H. J., Villatoro-Tello, E., Juárez, A., Montes-y-Gómez, M., & Villaseñor, L. (2013). Sexual predator detection in chats with chained classifiers. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 46–54). Atlanta, Georgia: Association for Computational Linguistics.
24. Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, *3*, 1289–1305.
25. Forsyth, E. N., & Martell, C. H. (2007). Lexical and Discourse Analysis of Online Chat Dialog. In *International Conference on Semantic Computing, 2007. ICSC 2007.* (pp. 19–26). http://doi.org/10.1109/ICSC.2007.55
26. Görnitz, N., Kloft, M., Rieck, K., & Brefeld, U. (2013). Toward Supervised Anomaly Detection. *Journal of Artificial Intelligence Research*, *46*(1), 235–262.
27. Guzman, J., & Poblete, B. (2013). On-line relevant anomaly detection in the Twitter stream: an efficient bursty keyword detection model. In *ACM SIGKDD Workshop on Outlier Detection and Description (ODD '13)* (pp. 31–39). New York, NY, USA: ACM.
28. Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, *18*(7), 1527–1554.
29. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.
30. Hogenboom, A., Bal, D., Frasincar, F., Bal, M., de Jong, F., & Kaymak, U. (2013). Exploiting Emoticons in Sentiment Analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 703–710). New York, NY, USA: ACM.
31. Inches, G., & Crestani, F. (2012). *Overview of the International Sexual Predator Identification Competition at PAN-2012*. CLEF (working notes), Rome, Italy.
32. Iqbal, F., Fung, B. C. M., & Debbabi, M. (2012). Mining Criminal Networks from Chat Log. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on* (Vol. 1, pp. 332–337).

33. Irfan, R., King, C. K., Grages, D., Ewen, S., Khan, S. U., Madani, S. A., … Li, H. (2015). A survey on text mining in social networks. *The Knowledge Engineering Review*, *30* (Special Issue 02), 157–170.
34. İrsoy, O., & Cardie, C. (2014). Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 720–728). Doha, Qatar.
35. Johnson, R. (2016). ConText2.0 Source Code. Retrieved January 23, 2016, from http://riejohnson.com/cnn_download.html
36. Johnson, R., & Zhang, T. (2015a). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
37. Johnson, R., & Zhang, T. (2015b). Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. Proceedings of Neural Information Processing Systems, Montreal.
38. Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In Association for Computational Linguistics *2014*.
39. Kang, I.-S., Kim, C.-K., Kang, S.-J., & Na, S.-H. (2012). IR-based k-Nearest Neighbor Approach for Identifying Abnormal Chat Users. Notebook for PAN at CLEF 2012, Rome, Italy.
40. Kern, R., Klampfl, S., & Zechner, M. (2012). Vote/Veto Classification, Ensemble Clustering and Sequence Classification for Author Identification. Presented at the Notebook for PAN at CLEF 2012, Rome, Italy: CLEF 2012.
41. Keyvanpour, M., Ebrahimi, M., Genc Nayebi, N., Ormandjieva, O., & Suen, C. Y. (2016). Automated Identification of Child Abuse in Chat Rooms by Using Data Mining. In *Data Mining Trends and Applications in Criminal Science and Investigations*. IGI Global.
42. Kierkegaard, S. (2008). Cybering, online grooming and ageplay. *Computer Law & Security Review*, *24*(1), 41–55.
43. Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar.
44. Klerks, P. (2003). The network paradigm applied to criminal organizations: Theoretical nitpicking or a relevant doctrine for investigators? Recent developments in the Netherlands. In *Transnational Organized Crime Perspectives on Global Security*. London: Routledge.
45. Kontostathis, A. (2009). Toward the tracking and categorization of internet predators. In Proceedings of Text Mining Workshop 2009 held in conjunction with Ninth Siam International Conference Data Mining, Sparks, NV, USA.
46. Kontostathis, A., Edwards, L., & Leatherman, A. (2010). Text Mining and Cybercrime. In *Text Mining* (pp. 149–164). John Wiley & Sons, Ltd.
47. Kontostathis, A., Reynolds, K., Garron, A., & Edwards, L. (2013). Detecting Cyberbullying: Query Terms and Techniques. In *Proceedings of the 5th Annual ACM Web Science Conference* (pp. 195–204). New York, NY, USA: ACM.

48. Kumaraswamy, R., Wazalwar, A., Khot, T., Shavlik, J., & Natarajan, S. (2015). Anomaly Detection in Text: The Value of Domain Knowledge. In *Florida Artificial Intelligence Research Society Conference*. AAAI.
49. Lecun, Y., Chopra, S., Hadsell, R., marc' aurelio, R., & Huang, fu-J. (2006). A Tutorial on Energy-Based Learning. In G. Bakir, T. Hofman, B. schölkopf, A. Smola, & B. Taskar (Eds.), *Predicting Structured Data*. MIT Press.
50. LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. In *International Symposium on Circuits and Systems (ISCAS), Proceedings of 2010 IEEE* (pp. 253–256).
51. Le, Q. V., & Mikolov, T. (2014a). Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML* 2014, Beijing, China, 21-26 June 2014 (pp. 1188–1196).
52. Le, Q. V., & Mikolov, T. (2014b). Distributed Representations of Sentences and Documents. *CoRR*, *abs/1405.4053*. Retrieved from http://arxiv.org/abs/1405.4053
53. Liu, B., & Zhang, L. (2012). A Survey of Opinion Mining and Sentiment Analysis. In C. C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (pp. 415–463). Springer US.
54. Mcghee, I., Bayzick, J., Kontostathis, A., Edwards, L., Mcbride, A., & Jakubowski, E. (2011). Learning to Identify Internet Sexual Predation. *International Journal of Electronic Commerce*, *15*(3), 103–122.
55. Michalopoulos, D., & Mavridis, I. (2011). Utilizing document classification for grooming attack recognition. In *IEEE Symposium on Computers and Communications (ISCC), 2011* (pp. 864–869).
56. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 935–940). New York, NY, USA: ACM.
57. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association,* Makuhari, Chiba, Japan, September 26-30, 2010 (pp. 1045–1048).
58. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality (pp. 3111–3119). Presented at the Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.
59. Morris, C. (2013, January 30). *Identifying Online Sexual Predators by SVM Classification with Lexical and Behavioral Features* (Master of Science Thesis). University of Toronto, Canada. Retrieved from ftp.cs.toronto.edu/pub/gh/Morris,Colin-MSc-thesis-2013.pdf
60. Ng, A. Y. (2004). Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning* (p. 78). New York, NY, USA: ACM.
61. Ng, A. Y., & Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and Naive Bayes. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* (pp. 841–848). MIT Press.

62. Olson, L. N., Daggs, J. L., Ellevold, B. L., & Rogers, T. K. K. (2007). Entrapping the Innocent: Toward a Theory of Child Sexual Predators' Luring Communication. *Communication Theory*, *17*(3), 231–251.
63. Pendar, N. (2007). Toward spotting the pedophile telling victim from predator in text chats (pp. 235–241). Washington, USA,: IEEE.
64. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)* (pp. 1532–1543).
65. Popescu, M., & Grozea, C. (2012). Kernel Methods and String Kernels for Authorship Analysis. Presented at the Notebook for PAN at CLEF 2012, Rome, Italy.
66. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, *323*(6088), 533–536.
67. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85 – 117. http://doi.org/http://dx.doi.org/10.1016/j.neunet.2014.09.003
68. Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution, *13*(7).
69. Schölkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New Support Vector Algorithms. *Neural Computation*, *12*(5), 1207–1245.
70. Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., & Platt, J. (2000). Support Vector Method for Novelty Detection. *Advances in Neural Information Processing Systems*, *12*, 582–588.
71. Socher, R., Lin, C. C.-Y., Ng, A. Y., & Manning, C. D. (2011). Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In L. Getoor & T. Scheffer (Eds.), *International Conference on Machine Learning* (pp. 129–136). Omnipress.
72. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1631–1642). Seattle, Washington, USA: Association for Computational Linguistics.
73. Tan, S. (2005). Neighbor-weighted K-nearest Neighbor for Unbalanced Text Corpus. *Expert Systems Applications*, *28*(4), 667–671. http://doi.org/10.1016/j.eswa.2004.12.023
74. Toews, V. (2013). *Royal Canadian Mounted Police Report on Plans and Priorities 2013-2014*. Canada: RCMP. Retrieved from http://www.rcmp-grc.gc.ca/rpp/2013-2014/rpp-eng.htm
75. Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc.
76. Villatoro-Tello, E., Juárez-González, A., Escalante, H. J., Montes-y-Gómez, M., & Villaseñor-Pineda, L. (2012). A Two-step Approach for Effective Detection of Misbehaving Users in Chats. Notebook for PAN at CLEF'12, Rome, Italy,: CLEF'12.
77. Yang, Y., & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 412–420). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645526.657137

78. Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Cornell University Library *(CoRR, abs/1510.03820)*, Retrieved from http://arxiv.org/abs/1510.03820

# APPENDIX A.

# Process Definitions in RapidMiner

Using the process definitions in this part, the similar results as obtained in chapter 3 can be reproduced.

### Anomaly Detection Train Process

```
<?xmlversion="1.0"encoding="UTF-8"standalone="no"?>
<processversion="5.3.012">
<context>
<input/>
<output/>
<macros/>
</context>
<operatoractivated="true"class="process"compatibility="5.3.012"expanded="tr
ue"name="Process">
<processexpanded="true">
<operatoractivated="true"class="text:process_document_from_file"compatibili
ty="5.3.002"expanded="true"height="76"name="Process Documents from
Files"width="90"x="45"y="75">
<listkey="text_directories">
<parameterkey="p"value="E:\University\RA\p"/>
<parameterkey="np"value="E:\University\RA\p"/>
</list>
<processexpanded="true">
<operatoractivated="true"class="text:transform_cases"compatibility="5.3.002
"expanded="true"name="Transform Cases (2)"/>
<operatoractivated="true"class="text:replace_tokens"compatibility="5.3.002"
expanded="true"name="Replace Tokens (2)">
<listkey="replace_dictionary">
<parameterkey="\*\*\*.*\*\*\*"value=" "/>
</list>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"name="RemoveDocumentForSkype">
<parameterkey="deletion_regex"value="\[[^a-z]*\]"/>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"name="Remove DocumentForFB">
<parameterkey="deletion_regex"value="[0-9|/|\s]*:[0-9][0-9]"/>
</operator>
<operatoractivated="true"class="text:tokenize"compatibility="5.3.002"expand
ed="true"name="Tokenize">
<parameterkey="mode"value="regular expression"/>
<parameterkey="characters"value=".:, !#%&amp;amp;()|\/`'\n'\t'"/>
<parameterkey="expression"value="[^(\p{L}0-9)]|\(|\)"/>
</operator>
<operatoractivated="true"class="text:filter_stopwords_french"compatibility=
"5.3.002"expanded="true"name="Filter Stopwords (French)"/>
<operatoractivated="true"class="text:filter_by_length"compatibility="5.3.00
2"expanded="true"name="Filter Tokens (by Length)">
<parameterkey="min_chars"value="2"/>
<parameterkey="max_chars"value="20"/>
</operator>
<connectfrom_port="document"to_op="Transform Cases (2)"to_port="document"/>
<connectfrom_op="Transform Cases (2)"from_port="document"to_op="Replace
Tokens (2)"to_port="document"/>
```

<connectfrom_op="Replace Tokens
(2)"from_port="document"to_op="RemoveDocumentForSkype"to_port="document"/>
<connectfrom_op="RemoveDocumentForSkype"from_port="document"to_op="Remove
DocumentForFB"to_port="document"/>
<connectfrom_op="Remove
DocumentForFB"from_port="document"to_op="Tokenize"to_port="document"/>
<connectfrom_op="Tokenize"from_port="document"to_op="Filter Stopwords
(French)"to_port="document"/>
<connectfrom_op="Filter Stopwords
(French)"from_port="document"to_op="Filter Tokens (by
Length)"to_port="document"/>
<connectfrom_op="Filter Tokens (by
Length)"from_port="document"to_port="document 1"/>
<portSpacingport="source_document"spacing="0"/>
<portSpacingport="sink_document 1"spacing="0"/>
<portSpacingport="sink_document 2"spacing="0"/>
</process>
</operator>
<operatoractivated="true"class="filter_examples"compatibility="5.3.012"expa
nded="true"height="76"name="OnlyPositiveExamples
(2)"width="90"x="179"y="75">
<parameterkey="condition_class"value="attribute_value_filter"/>
<parameterkey="parameter_string"value="label=p"/>
</operator>
<operatoractivated="true"class="select_attributes"compatibility="5.3.012"ex
panded="true"height="76"name="Select Attributes
(2)"width="90"x="313"y="75">
<parameterkey="attribute_filter_type"value="single"/>
<parameterkey="attribute"value="label"/>
<parameterkey="invert_selection"value="true"/>
<parameterkey="include_special_attributes"value="true"/>
</operator>
<operatoractivated="true"class="generate_attributes"compatibility="5.3.012"
expanded="true"height="76"name="Generate Attributes
(2)"width="90"x="179"y="165">
<listkey="function_descriptions">
<parameterkey="label"value=""p""/>
</list>
</operator>
<operatoractivated="true"class="set_role"compatibility="5.3.012"expanded="t
rue"height="76"name="Set Role (2)"width="90"x="313"y="165">
<parameterkey="attribute_name"value="label"/>
<parameterkey="target_role"value="label"/>
<listkey="set_additional_roles"/>
</operator>
<operatoractivated="true"class="support_vector_machine_libsvm"compatibility
="5.3.012"expanded="true"height="76"name="One-Class SVM
(2)"width="90"x="447"y="165">
<parameterkey="svm_type"value="one-class"/>
<parameterkey="kernel_type"value="linear"/>
<parameterkey="nu"value="0.1"/>
<parameterkey="epsilon"value="0.0010"/>
<listkey="class_weights"/>
</operator>
<operatoractivated="true"class="write_model"compatibility="5.3.012"expanded
="true"height="60"name="Write Model"width="90"x="581"y="75">
<parameterkey="model_file"value="C:\resolute\model_Anomaly"/>
</operator>
<connectfrom_op="Process Documents from Files"from_port="example
set"to_op="OnlyPositiveExamples (2)"to_port="example set input"/>

```
<connectfrom_op="OnlyPositiveExamples (2)"from_port="example set
output"to_op="Select Attributes (2)"to_port="example set input"/>
<connectfrom_op="Select Attributes (2)"from_port="example set
output"to_op="Generate Attributes (2)"to_port="example set input"/>
<connectfrom_op="Generate Attributes (2)"from_port="example set
output"to_op="Set Role (2)"to_port="example set input"/>
<connectfrom_op="Set Role (2)"from_port="example set output"to_op="One-
Class SVM (2)"to_port="training set"/>
<connectfrom_op="One-Class SVM (2)"from_port="model"to_op="Write
Model"to_port="input"/>
<connectfrom_op="Write Model"from_port="through"to_port="result 1"/>
<portSpacingport="source_input 1"spacing="0"/>
<portSpacingport="sink_result 1"spacing="0"/>
<portSpacingport="sink_result 2"spacing="0"/>
</process>
</operator>
</process>
```

**Anomaly Detection Test Process**

```
<?xmlversion="1.0"encoding="UTF-8"standalone="no"?>
<processversion="5.3.012">
<context>
<input/>
<output/>
<macros/>
</context>
<operatoractivated="true"class="process"compatibility="5.3.012"expanded="tr
ue"name="Process">
<processexpanded="true">
<operatoractivated="true"class="text:process_document_from_file"compatibili
ty="5.3.002"expanded="true"height="76"name="Process Documents from
Files"width="90"x="112"y="165">
<listkey="text_directories">
<parameterkey="dummy"value="E:\University\RA\p"/>
</list>
<processexpanded="true">
<operatoractivated="true"class="text:transform_cases"compatibility="5.3.002
"expanded="true"name="Transform Cases"/>
<operatoractivated="true"class="text:replace_tokens"compatibility="5.3.002"
expanded="true"name="Replace Tokens">
<listkey="replace_dictionary">
<parameterkey="\*\*\*.*\*\*\*"value=" "/>
</list>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"name="RemoveDocumentForSkype">
<parameterkey="deletion_regex"value="\[[^a-z]*\]"/>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"name="Remove DocumentForFB">
<parameterkey="deletion_regex"value="[0-9|/|\s]*:[0-9][0-9]"/>
</operator>
<operatoractivated="true"class="text:tokenize"compatibility="5.3.002"expand
ed="true"name="Tokenize">
<parameterkey="mode"value="regular expression"/>
<parameterkey="characters"value=".:, !#%&amp;;()|\/`'\n'\t'"/>
<parameterkey="expression"value="[^(\p{L}0-9)]|\(|\)"/>
</operator>
<operatoractivated="true"class="text:filter_stopwords_french"compatibility=
"5.3.002"expanded="true"name="Filter Stopwords (French)"/>
<operatoractivated="true"class="text:filter_by_length"compatibility="5.3.00
2"expanded="true"name="Filter Tokens (by Length)">
<parameterkey="min_chars"value="2"/>
<parameterkey="max_chars"value="20"/>
</operator>
<connectfrom_port="document"to_op="Transform Cases"to_port="document"/>
<connectfrom_op="Transform Cases"from_port="document"to_op="Replace
Tokens"to_port="document"/>
<connectfrom_op="Replace
Tokens"from_port="document"to_op="RemoveDocumentForSkype"to_port="document"
/>
<connectfrom_op="RemoveDocumentForSkype"from_port="document"to_op="Remove
DocumentForFB"to_port="document"/>
<connectfrom_op="Remove
DocumentForFB"from_port="document"to_op="Tokenize"to_port="document"/>
<connectfrom_op="Tokenize"from_port="document"to_op="Filter Stopwords
(French)"to_port="document"/>
```

```
<connectfrom_op="Filter Stopwords
(French)"from_port="document"to_op="Filter Tokens (by
Length)"to_port="document"/>
<connectfrom_op="Filter Tokens (by
Length)"from_port="document"to_port="document 1"/>
<portSpacingport="source_document"spacing="0"/>
<portSpacingport="sink_document 1"spacing="0"/>
<portSpacingport="sink_document 2"spacing="0"/>
</process>
</operator>
<operatoractivated="true"class="read_model"compatibility="5.3.012"expanded=
"true"height="60"name="Read Model"width="90"x="112"y="75">
<parameterkey="model_file"value="C:\resolute\model_Anomaly"/>
</operator>
<operatoractivated="true"class="apply_model"compatibility="5.3.012"expanded
="true"height="76"name="Apply Model"width="90"x="380"y="75">
<listkey="application_parameters"/>
</operator>
<connectfrom_op="Process Documents from Files"from_port="example
set"to_op="Apply Model"to_port="unlabelled data"/>
<connectfrom_op="Read Model"from_port="output"to_op="Apply
Model"to_port="model"/>
<connectfrom_op="Apply Model"from_port="labelled data"to_port="result 2"/>
<connectfrom_op="Apply Model"from_port="model"to_port="result 1"/>
<portSpacingport="source_input 1"spacing="0"/>
<portSpacingport="sink_result 1"spacing="0"/>
<portSpacingport="sink_result 2"spacing="0"/>
<portSpacingport="sink_result 3"spacing="0"/>
</process>
</operator>
</process>
```

**SVM Train Process**

```
<?xmlversion="1.0"encoding="UTF-8"standalone="no"?>
<processversion="5.3.012">
<context>
<input/>
<output/>
<macros/>
</context>
<operatoractivated="true"class="process"compatibility="5.3.012"expanded="tr
ue"name="Process">
<processexpanded="true">
<operatoractivated="true"class="text:process_document_from_file"compatibili
ty="5.3.002"expanded="true"height="76"name="Process Documents from
Files"width="90"x="112"y="75">
<listkey="text_directories">
<parameterkey="p"value="E:\University\RA\p"/>
<parameterkey="np"value="E:\University\RA\p"/>
</list>
<processexpanded="true">
<operatoractivated="true"class="text:transform_cases"compatibility="5.3.002
"expanded="true"height="60"name="Transform Cases"width="90"x="45"y="30"/>
<operatoractivated="true"class="text:replace_tokens"compatibility="5.3.002"
expanded="true"height="60"name="Replace Tokens"width="90"x="179"y="30">
<listkey="replace_dictionary">
<parameterkey="\*\*\*.*\*\*\*"value=" "/>
</list>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"height="60"name="RemoveDocumentForSkype"width="90"x="
179"y="120">
<parameterkey="deletion_regex"value="\[[^a-z]*\]"/>
</operator>
<operatoractivated="true"class="text:remove_document_parts"compatibility="5
.3.002"expanded="true"height="60"name="Remove
DocumentForFB"width="90"x="179"y="210">
<parameterkey="deletion_regex"value="[0-9|/|\s]*:[0-9][0-9]"/>
</operator>
<operatoractivated="true"class="text:tokenize"compatibility="5.3.002"expand
ed="true"height="60"name="Tokenize"width="90"x="313"y="30">
<parameterkey="mode"value="regular expression"/>
<parameterkey="characters"value=".:, !#%&amp;amp;()|\/`'\n'\t'"/>
<parameterkey="expression"value="[^(\p{L}0-9)]|\(|\)"/>
</operator>
<operatoractivated="true"class="text:filter_stopwords_french"compatibility=
"5.3.002"expanded="true"height="60"name="Filter Stopwords
(French)"width="90"x="447"y="30"/>
<operatoractivated="true"class="text:filter_by_length"compatibility="5.3.00
2"expanded="true"height="60"name="Filter Tokens (by
Length)"width="90"x="514"y="120">
<parameterkey="min_chars"value="2"/>
<parameterkey="max_chars"value="20"/>
</operator>
<connectfrom_port="document"to_op="Transform Cases"to_port="document"/>
<connectfrom_op="Transform Cases"from_port="document"to_op="Replace
Tokens"to_port="document"/>
<connectfrom_op="Replace
Tokens"from_port="document"to_op="RemoveDocumentForSkype"to_port="document"
/>
<connectfrom_op="RemoveDocumentForSkype"from_port="document"to_op="Remove
DocumentForFB"to_port="document"/>
```

```
<connectfrom_op="Remove
DocumentForFB"from_port="document"to_op="Tokenize"to_port="document"/>
<connectfrom_op="Tokenize"from_port="document"to_op="Filter Stopwords
(French)"to_port="document"/>
<connectfrom_op="Filter Stopwords
(French)"from_port="document"to_op="Filter Tokens (by
Length)"to_port="document"/>
<connectfrom_op="Filter Tokens (by
Length)"from_port="document"to_port="document 1"/>
<portSpacingport="source_document"spacing="0"/>
<portSpacingport="sink_document 1"spacing="0"/>
<portSpacingport="sink_document 2"spacing="0"/>
</process>
</operator>
<operatoractivated="true"class="x_validation"compatibility="5.3.012"expande
d="true"height="112"name="Validation"width="90"x="313"y="75">
<processexpanded="true">
<operatoractivated="true"class="support_vector_machine_libsvm"compatibility
="5.3.012"expanded="true"height="76"name="SVM"width="90"x="112"y="30">
<parameterkey="kernel_type"value="linear"/>
<parameterkey="C"value="100.0"/>
<listkey="class_weights"/>
</operator>
<connectfrom_port="training"to_op="SVM"to_port="training set"/>
<connectfrom_op="SVM"from_port="model"to_port="model"/>
<portSpacingport="source_training"spacing="0"/>
<portSpacingport="sink_model"spacing="0"/>
<portSpacingport="sink_through 1"spacing="0"/>
</process>
<processexpanded="true">
<operatoractivated="true"class="apply_model"compatibility="5.3.012"expanded
="true"height="76"name="Apply Model"width="90"x="45"y="30">
<listkey="application_parameters"/>
</operator>
<operatoractivated="true"class="performance_binominal_classification"compat
ibility="5.3.012"expanded="true"height="76"name="Performance"width="90"x="1
79"y="30">
<parameterkey="AUC (optimistic)"value="true"/>
<parameterkey="AUC"value="true"/>
<parameterkey="AUC (pessimistic)"value="true"/>
<parameterkey="precision"value="true"/>
<parameterkey="recall"value="true"/>
<parameterkey="f_measure"value="true"/>
<parameterkey="false_positive"value="true"/>
<parameterkey="false_negative"value="true"/>
<parameterkey="true_positive"value="true"/>
<parameterkey="true_negative"value="true"/>
</operator>
<connectfrom_port="model"to_op="Apply Model"to_port="model"/>
<connectfrom_port="test set"to_op="Apply Model"to_port="unlabelled data"/>
<connectfrom_op="Apply Model"from_port="labelled
data"to_op="Performance"to_port="labelled data"/>
<connectfrom_op="Performance"from_port="performance"to_port="averagable
1"/>
<portSpacingport="source_model"spacing="0"/>
<portSpacingport="source_test set"spacing="0"/>
<portSpacingport="source_through 1"spacing="0"/>
<portSpacingport="sink_averagable 1"spacing="0"/>
<portSpacingport="sink_averagable 2"spacing="0"/>
</process>
</operator>
```

```
<operatoractivated="true"class="write_model"compatibility="5.3.012"expanded
="true"height="60"name="Write Model"width="90"x="447"y="30">
<parameterkey="model_file"value="C:\resolute\model"/>
</operator>
<operatoractivated="true"class="write_performance"compatibility="5.3.012"ex
panded="true"height="60"name="Write Performance"width="90"x="447"y="120">
<parameterkey="performance_file"value="C:\resolute\performance"/>
</operator>
<connectfrom_op="Process Documents from Files"from_port="example
set"to_op="Validation"to_port="training"/>
<connectfrom_op="Validation"from_port="model"to_op="Write
Model"to_port="input"/>
<connectfrom_op="Validation"from_port="training"to_port="result 2"/>
<connectfrom_op="Validation"from_port="averagable 1"to_op="Write
Performance"to_port="input"/>
<connectfrom_op="Write Model"from_port="through"to_port="result 1"/>
<connectfrom_op="Write Performance"from_port="through"to_port="result 3"/>
<portSpacingport="source_input 1"spacing="0"/>
<portSpacingport="sink_result 1"spacing="0"/>
<portSpacingport="sink_result 2"spacing="0"/>
<portSpacingport="sink_result 3"spacing="0"/>
<portSpacingport="sink_result 4"spacing="0"/>
</process>
</operator>
</process>
```

# APPENDIX B.

# Bash Scripts samples to run ConText2 on GPU

Using the Linux bash script samples in this part, the similar results in chapter 4 can be obtained. It is necessary to make sure that the appropriate environment is set up before running these scripts (refer to chapter4).

## Training and Testing CNN with 1 Hidden Layer and 3 region sizes

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/software-gpu/cuda/7.5.18/lib64

gpu=-1  # <= change this to, e.g., "gpu=0" to use a specific GPU.
mem=23   # pre-allocate 2GB device memory
gpumem=${gpu}:${mem}

prep_exe=../bin/prepText
cnn_exe=../bin/conText

options="LowerCase UTF8"

  #Generate vocabulary
echoGeneraing vocabulary from training data ...

max_num=5000
vocab_fn=data/NEWPAN_63_trn-${max_num}.vocab
  #stopword_fn=data/stopwords
  $prep_exegen_vocabinput_fn=data/PAN-
train.tokvocab_fn=$vocab_fnmax_vocab_size=$max_num \
                            $options WriteCount

  #Generate region files (data/*.xsmatvar) and target files (data/*.y) for
training and testing CNN.

echo Generating region files with region size 2 and 3 ...

forpch_sz in 1 2 3; do
for set in train test; do
rnm=data/NEWPAN_63_${set}-patch${pch_sz}
      $prep_exegen_regions \
region_fn_stem=$rnminput_fn=data/PAN-${set} vocab_fn=$vocab_fn \
        $options text_fn_ext=.tok label_fn_ext=.cat \
label_dic_fn=data/PAN_cat.dic \
patch_size=$pch_szpatch_stride=1 padding=$((pch_sz-1))
done
done

  #Training and testing
log_fn=log_output/NEWPAN_63-seq.log
perf_fn=perf/NEWPAN_63-seq-perf.csv
echo
```

```
echo Training CNN and testing ...
echo This takes a while.  See $log_fn and $perf_fn for progress and see
param/seq-bow.param for the rest of the parameters.
nodes=2000  # number of neurons (weight vectors) in the convolution layer.
  $cnn_exe $gpumemcnn extension=multi conn0=0-top conn1=1-top conn2=2-top \
data_dir=data trnname=NEWPAN_63_train- tstname=NEWPAN_63_test- \
         reg_L2=0 top_reg_L2=1e-4 step_size=0.05 top_dropout=0.5 \
nodes=$nodes resnorm_width=$nodes \
LessVerbosetest_interval=1
evaluation_fn=$perf_fnsave_fn=output/NEWPAN_63_model \
loss=Square num_iterations=100 step_size_scheduler=Few \
step_size_decay=0.1 step_size_decay_at=80_90 mini_batch_size=100 \
         0dataset_no=0 1dataset_no=1 2dataset_no=2  data_ext0=patch1
data_ext1=patch2 data_ext2=patch3 \
layers=3 pooling_type=Max num_pooling=1 activ_type=Rect \
random_seed=1 datatype=sparse_multix_ext=.xsmatvary_ext=.y \
momentum=0.9 init_weight=0.01 init_intercept=0  \
resnorm_type=Cross resnorm_alpha=1 resnorm_beta=0.5 > ${log_fn}


../bin/conText -1 cnn_predictmodel_fn=output/NEWPAN_63_model.ite100
prediction_fn=output/NEWPAN_63_prediction.txt WriteText extension=multi
datatype=sparse_multitstname=NEWPAN_63_test- data_ext0=patch1
data_ext1=patch2  data_ext2=patch3  data_dir=data x_ext=.xsmatvar> output-
test
```

# Output of Training and Testing CNN with 1 Hidden Layer

```
MaxThreadsPerBlock=1024
MaxBlockDimX=1024
MaxGridDimX=2147483647
MaxSharedMemoryPerBlock=49152
Allocating device memory: 2.46961e+10
Pre-allocation of device memory failed with "out of memory".  Disabling device memory handler ...
Using GPU#0
---------------------
extension=multi conn0=0-top conn1=1-top conn2=2-top data_dir=data trnname=NEWPAN_63_train-
tstname=NEWPAN_63_test- reg_L2=0 top_reg_L2=1e-4 step_size=0.05 top_dropout=0.5 nodes=2000
resnorm_width=2000 LessVerbosetest_interval=1 evaluation_fn=perf/NEWPAN_63-seq-perf.csv
save_fn=output/NEWPAN_63_model loss=Square num_iterations=100 step_size_scheduler=Few
step_size_decay=0.1 step_size_decay_at=80_90 mini_batch_size=100 0dataset_no=0 1dataset_no=1
2dataset_no=2 data_ext0=patch1 data_ext1=patch2 data_ext2=patch3 layers=3 pooling_type=Max
num_pooling=1 activ_type=Rectrandom_seed=1 datatype=sparse_multix_ext=.xsmatvary_ext=.y
momentum=0.9 init_weight=0.01 init_intercept=0 resnorm_type=Cross resnorm_alpha=1 resnorm_beta=0.5
---------------------
"cnn":
datatype=sparse_multi
trnname=NEWPAN_63_train-
tstname=NEWPAN_63_test-
  data_ext0=patch1
  data_ext1=patch2
  data_ext2=patch3

data_dir=data
x_ext=.xsmatvar
y_ext=.y
num_batches=1

extension=multi
evaluation_fn=perf/NEWPAN_63-seq-perf.csv
Log:ON
CusparseIndex:ON
CusparseFprop:ON
gpu_max_threads=1024
gpu_max_blocks=2147483647
Mon Jan 18 11:05:46 2016: NEWPAN_63_train-patch1 sparsec batch#1
Mon Jan 18 11:05:47 2016:   #row=5000 #col=7221662 nz per col=1
Mon Jan 18 11:05:47 2016: #data = 59599
Mon Jan 18 11:05:47 2016: target-min,max=0,1
Mon Jan 18 11:05:47 2016: NEWPAN_63_train-patch2 sparsec batch#1
Mon Jan 18 11:05:48 2016:   #row=10000 #col=7779398 nz per col=1.85661
Mon Jan 18 11:05:48 2016: #data = 59599
Mon Jan 18 11:05:48 2016: NEWPAN_63_train-patch3 sparsec batch#1
Mon Jan 18 11:05:49 2016:   #row=15000 #col=7877590 nz per col=2.7502
Mon Jan 18 11:05:49 2016: #data = 59599
Mon Jan 18 11:05:49 2016: NEWPAN_63_test-patch1 sparsec batch#1
Mon Jan 18 11:05:52 2016:   #row=5000 #col=17008012 nz per col=1
Mon Jan 18 11:05:52 2016: #data = 138338
Mon Jan 18 11:05:52 2016: target-min,max=0,1
Mon Jan 18 11:05:52 2016: NEWPAN_63_test-patch2 sparsec batch#1
Mon Jan 18 11:05:54 2016:   #row=10000 #col=18443054 nz per col=1.84438
Mon Jan 18 11:05:54 2016: #data = 138338
Mon Jan 18 11:05:54 2016: NEWPAN_63_test-patch3 sparsec batch#1
```

```
Mon Jan 18 11:05:57 2016:   #row=15000 #col=18706255 nz per col=2.72765
Mon Jan 18 11:05:57 2016: #data = 138338
Mon Jan 18 11:05:57 2016: Start ... #train=59599, #test=138338
--------------------
Mon Jan 18 11:05:57 2016: Data signature: [0]dim:1;channel:5000;size0:-1;[1]dim:1;channel:10000;size0:-
1;[2]dim:1;channel:15000;size0:-1;
Mon Jan 18 11:05:57 2016: #class=2

layers=3
save_fn=output/NEWPAN_63_model
initial_iteration=0
test_interval=1
num_iterations=100
random_seed=1
mini_batch_size=100
LessVerbose:ON
loss=Square

step_size_scheduler=Few
step_size_decay=0.1
step_size_decay_at=80_90

test_mini_batch_size=100

  conn0=0-top
  conn1=1-top
  conn2=2-top

  0dataset_no=0
Cold-starting (variable-size input) layer#0

  0init_weight=0.01
  0init_intercept=0
  0reg_L2=0

  0step_size=0.05
  0step_sizeb_coeff=1
  0momentum=0.9
  0FastFlush:ON

  0nodes=2000

  0activ_type=Rect

  0pooling_type=Max
  0num_pooling=1

  0resnorm_type=Cross
  0resnorm_alpha=1
  0resnorm_beta=0.5
  0resnorm_one=1
  0resnorm_width=2000
--------  weights  --------
input dim: 5000
output dim: 2000
  #weights: 10000000
  ---------------------------
```

```
  1dataset_no=1
Cold-starting (variable-size input) layer#1

  1init_weight=0.01
  1init_intercept=0
  1reg_L2=0

  1step_size=0.05
  1step_sizeb_coeff=1
  1momentum=0.9
  1FastFlush:ON

  1nodes=2000

  1activ_type=Rect

  1pooling_type=Max
  1num_pooling=1

  1resnorm_type=Cross
  1resnorm_alpha=1
  1resnorm_beta=0.5
  1resnorm_one=1
  1resnorm_width=2000
--------  weights  --------
input dim: 10000
output dim: 2000
  #weights: 20000000
  ---------------------------

  2dataset_no=2
Cold-starting (variable-size input) layer#2

  2init_weight=0.01
  2init_intercept=0
  2reg_L2=0

  2step_size=0.05
  2step_sizeb_coeff=1
  2momentum=0.9
  2FastFlush:ON

  2nodes=2000

  2activ_type=Rect

  2pooling_type=Max
  2num_pooling=1

  2resnorm_type=Cross
  2resnorm_alpha=1
  2resnorm_beta=0.5
  2resnorm_one=1
  2resnorm_width=2000
--------  weights  --------
input dim: 15000
```

```
output dim: 2000
  #weights: 30000000
  ---------------------------
Cold-starting connector#4  (0,1,2) -> (3)
Cold-starting the top layer

top_init_weight=0.01
top_init_intercept=0
  top_reg_L2=0.0001

top_step_size=0.05
top_step_sizeb_coeff=1
top_momentum=0.9
top_FastFlush:ON

top_dropout=0.5
------  top layer  ------
input: 1
--------  weights  --------
input dim: 6000
output dim: 2
  #weights: 12000
  ---------------------------
Mon Jan 18 11:06:04 2016: Checking word-mapping ...
Mon Jan 18 11:06:04 2016: supervised training: #hidden=3
Mon Jan 18 11:06:04 2016: Resetting step-sizes to s0 (initial value) ...
Mon Jan 18 11:09:12 2016:  ite,1,0.026724, test-loss,0.0172853, perf:err,0.0186572
Mon Jan 18 11:12:21 2016:  ite,2,0.0208258, test-loss,0.0158059, perf:err,0.0186572
Mon Jan 18 11:15:28 2016:  ite,3,0.0185253, test-loss,0.0139906, perf:err,0.0186572
Mon Jan 18 11:18:35 2016:  ite,4,0.0160518, test-loss,0.012231, perf:err,0.0186427
Mon Jan 18 11:21:43 2016:  ite,5,0.0139379, test-loss,0.0108161, perf:err,0.0148188
Mon Jan 18 11:24:51 2016:  ite,6,0.012372, test-loss,0.00978405, perf:err,0.0115153
Mon Jan 18 11:27:57 2016:  ite,7,0.0112392, test-loss,0.00924679, perf:err,0.00954908
Mon Jan 18 11:31:04 2016:  ite,8,0.0103501, test-loss,0.00841079, perf:err,0.00903584
Mon Jan 18 11:34:11 2016:  ite,9,0.00968023, test-loss,0.00795974, perf:err,0.00865995
Mon Jan 18 11:37:19 2016:  ite,10,0.00921873, test-loss,0.00777427, perf:err,0.00821177
Mon Jan 18 11:40:25 2016:  ite,11,0.00877768, test-loss,0.00748177, perf:err,0.0081684
Mon Jan 18 11:43:32 2016:  ite,12,0.00847411, test-loss,0.00720118, perf:err,0.00795154
Mon Jan 18 11:46:39 2016:  ite,13,0.0081221, test-loss,0.00706029, perf:err,0.00772022
Mon Jan 18 11:49:47 2016:  ite,14,0.00791285, test-loss,0.00709969, perf:err,0.00740216
Mon Jan 18 11:52:54 2016:  ite,15,0.00771824, test-loss,0.00690093, perf:err,0.00765516
Mon Jan 18 11:56:01 2016:  ite,16,0.00753093, test-loss,0.00673981, perf:err,0.00756119
Mon Jan 18 11:59:08 2016:  ite,17,0.00734683, test-loss,0.00670893, perf:err,0.00745999
Mon Jan 18 12:02:16 2016:  ite,18,0.00716445, test-loss,0.00663079, perf:err,0.00731542
Mon Jan 18 12:05:23 2016:  ite,19,0.00708644, test-loss,0.00652256, perf:err,0.00740939
Mon Jan 18 12:08:31 2016:  ite,20,0.00692339, test-loss,0.00647773, perf:err,0.00721421
Mon Jan 18 12:11:38 2016:  ite,21,0.0067227, test-loss,0.00640317, perf:err,0.00722867
Mon Jan 18 12:14:45 2016:  ite,22,0.00663982, test-loss,0.00636471, perf:err,0.00712747
Mon Jan 18 12:17:53 2016:  ite,23,0.00647401, test-loss,0.00632413, perf:err,0.00711301
Mon Jan 18 12:21:00 2016:  ite,24,0.0063725, test-loss,0.00629216, perf:err,0.00707687
Mon Jan 18 12:24:07 2016:  ite,25,0.00626494, test-loss,0.00626715, perf:err,0.00704796
Mon Jan 18 12:27:14 2016:  ite,26,0.00614435, test-loss,0.00627004, perf:err,0.00695398
Mon Jan 18 12:30:21 2016:  ite,27,0.00605459, test-loss,0.00620749, perf:err,0.00691061
Mon Jan 18 12:33:28 2016:  ite,28,0.00597598, test-loss,0.00617772, perf:err,0.00689615
Mon Jan 18 12:36:34 2016:  ite,29,0.00583246, test-loss,0.006159, perf:err,0.00691061
Mon Jan 18 12:39:42 2016:  ite,30,0.00576746, test-loss,0.00614621, perf:err,0.00691061
Mon Jan 18 12:42:49 2016:  ite,31,0.00562373, test-loss,0.0061467, perf:err,0.00685278
```

Mon Jan 18 12:45:56 2016:  ite,32,0.00556077, test-loss,0.00610209, perf:err,0.00686001
Mon Jan 18 12:49:03 2016:  ite,33,0.00547382, test-loss,0.00609899, perf:err,0.00680941
Mon Jan 18 12:52:10 2016:  ite,34,0.00537371, test-loss,0.00606347, perf:err,0.00685278
Mon Jan 18 12:55:18 2016:  ite,35,0.00528744, test-loss,0.00605293, perf:err,0.00683109
Mon Jan 18 12:58:24 2016:  ite,36,0.0052427, test-loss,0.00609292, perf:err,0.00673712
Mon Jan 18 13:01:33 2016:  ite,37,0.00513019, test-loss,0.00603802, perf:err,0.00683109
Mon Jan 18 13:04:40 2016:  ite,38,0.00510573, test-loss,0.00604124, perf:err,0.00676604
Mon Jan 18 13:07:48 2016:  ite,39,0.00495817, test-loss,0.00601465, perf:err,0.00667206
Mon Jan 18 13:10:55 2016:  ite,40,0.00490143, test-loss,0.00598208, perf:err,0.00674435
Mon Jan 18 13:14:01 2016:  ite,41,0.00484445, test-loss,0.00599459, perf:err,0.00676604
Mon Jan 18 13:17:08 2016:  ite,42,0.00477985, test-loss,0.00597816, perf:err,0.00673712
Mon Jan 18 13:20:16 2016:  ite,43,0.00470749, test-loss,0.00599564, perf:err,0.00659978
Mon Jan 18 13:23:23 2016:  ite,44,0.00463818, test-loss,0.00595287, perf:err,0.00669375
Mon Jan 18 13:26:30 2016:  ite,45,0.00457485, test-loss,0.00593857, perf:err,0.00667929
Mon Jan 18 13:29:38 2016:  ite,46,0.00448207, test-loss,0.00598888, perf:err,0.00657809
Mon Jan 18 13:32:45 2016:  ite,47,0.00445995, test-loss,0.00591223, perf:err,0.00660701
Mon Jan 18 13:35:51 2016:  ite,48,0.00438375, test-loss,0.00591687, perf:err,0.00658532
Mon Jan 18 13:38:58 2016:  ite,49,0.00437371, test-loss,0.00591442, perf:err,0.00653472
Mon Jan 18 13:42:06 2016:  ite,50,0.00429192, test-loss,0.00618477, perf:err,0.00675158
Mon Jan 18 13:45:14 2016:  ite,51,0.00423354, test-loss,0.00594921, perf:err,0.00654918
Mon Jan 18 13:48:21 2016:  ite,52,0.00419088, test-loss,0.00590846, perf:err,0.00647689
Mon Jan 18 13:51:28 2016:  ite,53,0.00413698, test-loss,0.00596121, perf:err,0.00663592
Mon Jan 18 13:54:35 2016:  ite,54,0.004087, test-loss,0.00587985, perf:err,0.0065058
Mon Jan 18 13:57:42 2016:  ite,55,0.00401574, test-loss,0.00587879, perf:err,0.00662146
Mon Jan 18 14:00:49 2016:  ite,56,0.00397106, test-loss,0.00586431, perf:err,0.00646966
Mon Jan 18 14:03:57 2016:  ite,57,0.00390843, test-loss,0.00585044, perf:err,0.00648412
Mon Jan 18 14:07:05 2016:  ite,58,0.00386737, test-loss,0.00595497, perf:err,0.00654918
Mon Jan 18 14:10:12 2016:  ite,59,0.00384201, test-loss,0.00584586, perf:err,0.0064552
Mon Jan 18 14:13:19 2016:  ite,60,0.00381128, test-loss,0.00584796, perf:err,0.00653472
Mon Jan 18 14:16:27 2016:  ite,61,0.00375044, test-loss,0.00584148, perf:err,0.0064046
Mon Jan 18 14:19:34 2016:  ite,62,0.00368803, test-loss,0.00584561, perf:err,0.00638292
Mon Jan 18 14:22:41 2016:  ite,63,0.00368615, test-loss,0.00583726, perf:err,0.00641906
Mon Jan 18 14:25:48 2016:  ite,64,0.00363295, test-loss,0.00582152, perf:err,0.00641906
Mon Jan 18 14:28:55 2016:  ite,65,0.00358791, test-loss,0.0058231, perf:err,0.0064552
Mon Jan 18 14:32:02 2016:  ite,66,0.00357559, test-loss,0.00583398, perf:err,0.00644798
Mon Jan 18 14:35:10 2016:  ite,67,0.00349453, test-loss,0.00596487, perf:err,0.00659978
Mon Jan 18 14:38:17 2016:  ite,68,0.003471, test-loss,0.00581963, perf:err,0.0064552
Mon Jan 18 14:41:23 2016:  ite,69,0.00345658, test-loss,0.00581807, perf:err,0.00641906
Mon Jan 18 14:44:31 2016:  ite,70,0.00341461, test-loss,0.0058233, perf:err,0.0064046
Mon Jan 18 14:47:40 2016:  ite,71,0.00339703, test-loss,0.00581688, perf:err,0.00642629
Mon Jan 18 14:50:48 2016:  ite,72,0.00334747, test-loss,0.00582024, perf:err,0.00636123
Mon Jan 18 14:53:54 2016:  ite,73,0.0033365, test-loss,0.00580703, perf:err,0.00639015
Mon Jan 18 14:57:03 2016:  ite,74,0.00328377, test-loss,0.00584041, perf:err,0.00639737
Mon Jan 18 15:00:12 2016:  ite,75,0.00324534, test-loss,0.0058078, perf:err,0.0064046
Mon Jan 18 15:03:20 2016:  ite,76,0.00322428, test-loss,0.00580896, perf:err,0.00636846
Mon Jan 18 15:06:28 2016:  ite,77,0.00321345, test-loss,0.00583242, perf:err,0.00638292
Mon Jan 18 15:09:35 2016:  ite,78,0.00320331, test-loss,0.0058002, perf:err,0.00636846
Mon Jan 18 15:12:41 2016:  ite,79,0.00315703, test-loss,0.00579791, perf:err,0.00646243
Mon Jan 18 15:15:48 2016:  ite,80,0.00313677, test-loss,0.00578705, perf:err,0.00643352
Mon Jan 18 15:15:48 2016: Setting step-sizes to s0 times 0.1
Mon Jan 18 15:18:56 2016:  ite,81,0.00304006, test-loss,0.00578541, perf:err,0.00641906
Mon Jan 18 15:22:02 2016:  ite,82,0.00300443, test-loss,0.00578387, perf:err,0.00642629
Mon Jan 18 15:25:10 2016:  ite,83,0.00299182, test-loss,0.00578651, perf:err,0.00637569
Mon Jan 18 15:28:18 2016:  ite,84,0.00300224, test-loss,0.0057842, perf:err,0.00641183
Mon Jan 18 15:31:25 2016:  ite,85,0.00300212, test-loss,0.00578414, perf:err,0.00641906
Mon Jan 18 15:34:32 2016:  ite,86,0.00297744, test-loss,0.00578662, perf:err,0.00636846
Mon Jan 18 15:37:42 2016:  ite,87,0.00300568, test-loss,0.00578585, perf:err,0.00641183

Mon Jan 18 15:40:49 2016:  ite,88,0.00294359, test-loss,0.00578694, perf:err,0.00642629
Mon Jan 18 15:43:57 2016:  ite,89,0.00300074, test-loss,0.00579143, perf:err,0.00637569
Mon Jan 18 15:47:04 2016:  ite,90,0.00298773, test-loss,0.00578399, perf:err,0.00639015
Mon Jan 18 15:47:04 2016: Setting step-sizes to s0 times 0.01
Mon Jan 18 15:50:11 2016:  ite,91,0.00295923, test-loss,0.00578439, perf:err,0.00639737
Mon Jan 18 15:53:19 2016:  ite,92,0.00295836, test-loss,0.00578379, perf:err,0.00639015
Mon Jan 18 15:56:26 2016:  ite,93,0.00298553, test-loss,0.0057854, perf:err,0.00636123
Mon Jan 18 15:59:33 2016:  ite,94,0.00295286, test-loss,0.00578473, perf:err,0.00637569
Mon Jan 18 16:02:41 2016:  ite,95,0.00296693, test-loss,0.00578416, perf:err,0.00638292
Mon Jan 18 16:05:48 2016:  ite,96,0.00296608, test-loss,0.00578433, perf:err,0.00639015
Mon Jan 18 16:08:55 2016:  ite,97,0.00297198, test-loss,0.00578418, perf:err,0.00638292
Mon Jan 18 16:12:04 2016:  ite,98,0.00297666, test-loss,0.00578406, perf:err,0.00638292
Mon Jan 18 16:15:15 2016:  ite,99,0.00297813, test-loss,0.00578418, perf:err,0.00637569
Mon Jan 18 16:18:24 2016:  ite,100,0.00298606, test-loss,0.00578387, perf:err,0.00638292
Mon Jan 18 16:18:24 2016: Saving the model to output/NEWPAN_63_model.ite100
Mon Jan 18 16:18:27 2016: Done ...
elapsed: 18237.4

## Training and Testing CNN with 2 Hidden Layers

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/software-gpu/cuda/7.5.18/lib64

gpu=-1  # <= change this to, e.g., "gpu=0" to use a specific GPU.
mem=22   # pre-allocate 2GB device memory
gpumem=${gpu}:${mem}

prep_exe=../bin/prepText
cnn_exe=../bin/conText

options="LowerCase UTF8"

#---  Generate vocabulary
echoGeneraing vocabulary from training data ...

max_num=5000
vocab_fn=data/NEWPAN_54_trn-${max_num}.vocab
  $prep_exegen_vocabinput_fn=data/PAN-
train.tokvocab_fn=$vocab_fnmax_vocab_size=$max_num  $options WriteCount

#---  Generate region files (data/*.xsmatvar) and target files (data/*.y)
for training and testing CNN.

echo Generating region files ...

pch_sz=3

for set in train test; do
rnm=data/NEWPAN_54_${set}-patch${pch_sz}
    $prep_exegen_regions \
region_fn_stem=$rnminput_fn=data/PAN-${set} vocab_fn=$vocab_fn \
      $options text_fn_ext=.tok label_fn_ext=.cat \
label_dic_fn=data/PAN_cat.dic \
patch_size=$pch_szpatch_stride=1 padding=$((pch_sz-1))
done


#---  Training and testing
log_fn=log_output/NEWPAN_54-seq.log
perf_fn=perf/NEWPAN_54-seq-perf.csv
echo
echo Training CNN and testing ...
   $cnn_exe $gpumemcnnrandom_seed=1 test_interval=100\
      x_ext=.xsmatvary_ext=.y datatype=sparse data_dir=data
      trnname=NEWPAN_54_train-patch${pch_sz} tstname=NEWPAN_54_test-
      patch${pch_sz}\
      layers=2 activ_type=Rect 0nodes=500 0resnorm_width=500
      0pooling_type=Max 0pooling_size=2 0pooling_stride=1 1nodes=400
      1patch_size=3 1patch_stride=1 1padding=2 1pooling_type=Max
      1num_pooling=1 loss=Square mini_batch_size=100 momentum=0.9
      step_size=0.05 top_dropout=0.5 reg_L2=1e-4 num_iterations=100
      step_size_scheduler=Few step_size_decay=0.1 step_size_decay_at=80_90
      \
```

```
save_fn=output/NEWPAN_54_model evaluation_fn=$perf_fn> ${log_fn}

  ../bin/conText -1 cnn_predictmodel_fn=output/NEWPAN_54_model.ite100
prediction_fn=output/NEWPAN_54_prediction.txt WriteText datatype=sparse
tstname=NEWPAN_54_test-patch3 data_dir=data x_ext=.xsmatvar> output-test
```

# APPENDIX C.

# RapidMiner's Source Code Formula for TFIDF Calculation

The following java class belongs to the "vectorcreation" package in Text processing plug-in of Rapidminer 5.03. The class is responsible for generating the TFIDF-weighted word vector.

```
/*
 *  RapidMiner Text Processing Extension
 *
 *  Copyright (C) 2001-2013 by Rapid-I and the contributors
 *
 *  Complete list of developers available at our web site:
 *
 *       http://rapid-i.com
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU Affero General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU Affero General Public License for more details.
 *
 *  You should have received a copy of the GNU Affero General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 */
package com.rapidminer.operator.text.io.vectorcreation;

import com.rapidminer.operator.text.WordList;

/**
 * This class represents a mechanism to create TFIDF word vectors. The resulting
vectors are normalized.
 *
 * @author Michael Wurst
 */
public class TFIDF implements VectorCreator {

     public double[] createVector(float[] frequencies, WordList wordList) {

        // Obtain the total number of documents and the document frequencies
        int numDocuments = wordList.getNumberOfDocuments();
        int[] docFrequencies = wordList.getDocumentFrequencies();
        double totalTermNumber = 0;
        for (float value: frequencies)
             totalTermNumber += value;


        // Create the result structure
        double[] wv = new double[docFrequencies.length];

        // Create the vector

        // If the document contains at least one term
        if (totalTermNumber > 0) {
            double length = 0.0;
            for (int i = 0; i < wv.length; i++) {
                // Note: docFrequencies[i] is always > 0 as otherwise the word
                // would not be in the word list, it is also always smaller as
                // the total number of documents
```

```
                double idf = Math.log(((double) numDocuments) / ((double)
docFrequencies[i]));
                wv[i] = (frequencies[i] / totalTermNumber) * idf;
                length += wv[i] * wv[i];
            }
            length = Math.sqrt(length);

            // Normalize the vector
            if (length > 0.0)
                for (int i = 0; i < wv.length; i++)
                    wv[i] = wv[i] / length;
        }
        return wv;
    }
}
```