APPROVAL SHEET

Title of Thesis: Information Extraction of Security related entities and concepts from unstructured text.

Name of Candidate: Ravendar Lal
Masters in Computer Science,
2013

Thesis and Abstract Approved: _____

Dr. Tim Finin
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: MY-FULL-NAME.

Permanent Address: MY-FULL-ADDRESS.

Degree and date to be conferred: DEGREE-NAME, GRADUATION-MONTH GRADUATION-YEAR.

Date of Birth: MY-BIRTHDATE.

Place of Birth: MY-PLACE-OF-BIRTH.

Secondary Education: MY-HIGH-SCHOOL, MY-HIGH-SCHOOLS-CITY, MY-HIGH-SCHOOLS-STATE.

Collegiate institutions attended:

University of Maryland Baltimore County, DEGREE-NAME MY-MAJOR, GRADUATION-YEAR.
MY-OTHER-DEGREES.

Major: MY-MAJOR.

Minor: MY-MINOR.

Professional publications:

FULL-CITATION-INFORMATION.
FULL-CITATION-INFORMATION.

Professional positions held:

EMPLOYMENT-INFO. (START-DATE – END-DATE).
EMPLOYMENT-INFO. (START-DATE – END-DATE).

ABSTRACT

Title of Thesis: Information Extraction of Security related entities and concepts from unstructured text

Ravendar Lal, Masters in Computer Science, 2013

Thesis directed by: Tim Finin, Professor
Department of Computer Science and
Electrical Engineering

Cyber Security has been a big concern especially in past one decade where it is witnessed that targets ranging from large number of internet users to government agencies are being attacked because of vulnerabilities present in the system. Even though these vulnerabilities are identified and published publicly but response has always been slow in covering up these vulnerabilities because there is no automatic mechanism to understand and process this unstructured text that is published on internet. Our system will be tackling this problem of processing unstructured text by identifying the security related terms including entities and concepts from various unstructured data sources. This information extraction task will help expediting the process of understanding and realizing the vulnerabilities and thus making systems secure at faster rate.

This work will be describing a system that automatically extracts the terms from Cybersecurity blogs and security bulletins using Natural Language Processing (NLP) and text mining methods. Our NLP model is trained on manually annotated data using open-ended blogs and more structured text like company's official security bulletins. This manually annotated data is unique of its kind since no such previous work has been done using this methodology and it can be a significant contribution for people working in this domain. Our named entity recognition model is trained on conditional random fields (CRFs) based Stanford NER. This automation system will be able to help administrators of organiza-

tions and governments to prioritize the task of beefing up their security and moreover track unofficial data sources like chat rooms and twitter for zero day attacks.

Information Extraction of cyber security related terms and concepts from unstructured text

by

RAVENDAR LAL

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
MASTER OF SCIENCE
2013

Dedicated to my Parents who supported me when no one did, My siblings who helped me whenever I needed, My teachers and Mentors who gave me wisdom to accomplish this.

ACKNOWLEDGMENTS

Write your acknowledgment here.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Computer Security was once considered as 'good to have' feature for system. This was probably before the advent of Internet when Systems used to work in isolated way. There was little or no communication between the systems during that time. Everything was safe even if some system with in organization is infected with virus. Usually the mode of communication or exchanging data between different systems was magnetic disks.

World has changed with the advent of Internet. Systems were connected, world became Global, reaching as well as breaching boundaries become far easier. This threat was largely ignored in 90's (first decade of Internet). Software Vulnerabilities, defined as loopholes present in software product, were neither widely exploited nor there was a wide spread sharing between hackers.

But if we see in past one decade, Cyber security has emerged as an individual field and focus point among researchers and practitioners. This can be estimated by the fact that U.S Federal Cybersecurity is planning to invest $65.5 Billion in next 5 years (2013-2018) (2).

Vulnerabilities

Software Vulnerabilities are defined as security breaches or loopholes that are present in the system. Information about these vulnerabilities are published at regular level by software companies and they also releases updates and patches to cover them up. Information about these vulnerabilities does not come first from Software Companies who develops these products but most of the time hackers are first to note and exploit them. It is mainly because Software Companies test their products vigorously to make sure there is no security loophole in system, according to them, before releasing the products. Hackers are the ones that looks for these loopholes in system to achieve their objectives. When hackers come across these loopholes, they post these updates at various blogs and forums in order to spread the vulnerability. That is how Software Companies also come to know about these vulnerabilities since they continuously track these public hacker forums to secure their systems.

Since these vulnerabilities are mentioned in the form of security bulletins or blogs which are all unstructured form of text, thus it is a cumbersome task for System Administrator to track and read all these textual resources. But since it is very critical task thus it is impossible to avoid this job as well.

Twitter is also another example of unstructured text where people tweets about any widespread virus or Trojan. Tracking and analyzing this source of Information will also help. People tweet about any problem that they face and if problem is widespread people will tweet on same hashtag which basically group people facing same problem. Analyzing these tweets can help in early detection of any vulnerability or attack.

We approached this problem as a task of text analysis. In this work, we are propos-

ing a methodology to analyze the text from unstructured data sources like blogs, security bulletins and CVEs. Analysis of this large chunk of data helps to classify irrelevant and relevant information from it.

We envisioned a system that will be able to infer information any piece of article. Inferring such text will help to make detection of vulnerabilities faster and more efficient. It should be noted that Detection is first and most important step of prevention.

In this work, we are proposing a system that identifies important and relevant entities and concepts from text which mainly discusses about cyber-attacks and software vulnerabilities. Filtering relevant information from text is the first step to inference of information.

We built a Named Entity Recognizer (NER) that identifies technical jargon and characteristics of an attack. We defined various classes after analyzing the text. These classes are mentioned in detailed in following chapters. Dataset was collected from various places which are mentioned in detail in System Architecture.

Chapter 2

BACKGROUND AND RELATED WORK

A lot of work has been done in the domain of Information extraction where people have applied this problem of Named Entity Recognition on various domains. In this section, we will be reviewing not only how people have various methods to train NER system but we will also see other approaches people have adopted to solve the problem of cyber security.

Work of Undercoffer et al. (14) described a system where ontology was defined to extract the characteristics of an attack. This work was based on the revision of over 4000 intrusions and strategies followed by them to attack a system. This work introduces the concept of Means and Consequences classes which are also used in our work. This work followed the rule based approach by using best practices of semantic web .

One recent work (12) further improved above mentioned system where authors introduced a machine learning approach to detect similar concepts. This system developed a support vector machine (SVM) based classifier to detect relevant text description. Once relevant text description is selected it is passed to a knowledge base, Wikitology (13) which helps to detect entities. This work further used those entities to make RDF triples on which

one can make assertions to infer from it. This work was quite impressive except for the fact that SVM classifier can be further improved and also it was identifying Means and Consequences only. Due to this shortcoming, system was missing very important relevant information from text.

One other recent work (10) tried to improve the system and proposed an end-to-end approach to convert text description into RDF triples. But this system used off-the-shelf NER component, Open Calais (4), which is primarily designed for people, places and organizations. Open Calais doesn't do a satisfactory job and misses many important terms and concepts from text description.

Dazhi et al. (7) introduce the work where they developed a system to filter Biomedical Literature using methods of Natural Language Processing. In this work, they trained Stanford Named Entity Recognizer in order to filter the information from biomedical text. Similarly, Manaal et al. (5) described a system where they trained a NER for German Language using Stanford NER.

Shabana et al. (11) described a system to extract information from text description where they used very basic TFID score in order to extract information from the CVEs.

One other survey work by Lev et al. (9) described various Design challenges and Misconceptions present in NER. This work also shows how various dataset should be tagged and what are the advantages and disadvantages of various methods used for training purpose. This paper points to 4 important key design decision in any NER system and that representation of text chunks, inference algorithm, modeling non-local dependencies in text and using external knowledge resources in NER.

Lutz et al. presents another approach to classify SOA vulnerabilities. They used simple string matching on CVEs to classify dataset in various categories and didn't use any machine learning techniques to classify CVEs. This work is mainly interested in Service Oriented Architecture (SOA) vulnerabilities.

Chapter 3

SYSTEM ARCHITECTURE

In this section, we are describing training as well as annotation middle ware that was designed as a part of architecture.

3.1 Training Module

In order to train a Named Entity Recognizer, we followed an approach given in figure 3.1.

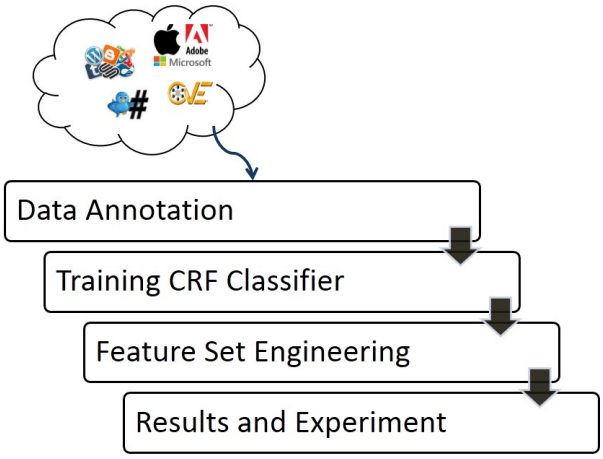

FIG. 3.1. Training Module

In following sections, we will be describing these modules.

3.1.1 Data Collection

Since this idea of implementing Information Extraction framework for Cybersecurity was quite new thus we could not find any data that can be used for training purpose. Thus, we have to collect data from various sources that should accommodate the information that we require. Initially, data was collected from various resources that including:

- Antivirus Websites
- Blogs
- Official Security Bulletins
- Common Vulnerability Enumerations (CVE)

We collected the data from over 60 blogs from various blogs and forums. Having this dataset helps us to understand the problems of unstructured text and how different people write differently. After that, we collected over 100 articles from official security bulletins released by Microsoft and Adobe. The bulletins were chosen carefully from this dataset as we wanted to have a dataset mentioning various products and cyber-attacks.

Common Vulnerability Enumerations (CVEs) are maintained and released by National Vulnerability Database (NVD) [1]. We selected over 300 CVEs which collectively was an excellent dataset for some of the major classes like means and consequences.

[1] `http://nvd.nist.gov/`

3.1.2 Data Analysis

In order to build an information extraction system that filters relevant information from the text, we needed to identify classes and concepts that can be used to make inference from given text or atleast understand the text to the level that human or machine can understand.

After rigorous analysis of various texts from above mentioned datasets, it was realized that following classes will serve the purpose for this task:

1. Software
    (a) Operating_System
2. Network_Terms
3. Attack (If you can't decide between means and consequence)
    (a) Means – Immediate system reaction to input
    (b) Consequences – final result of an attack
4. File_Name
5. Hardware
6. Other_Technical_Terms
7. NER_Modifier

Following is the description of each class and purpose :

1. Software: This class identifies software products including the version number. There was a significant focus on version number because it was noted that vulnerabilities are not necessarily present in every version of a software product.

   (a) Operating_System: Software was further divided in Operating System as it was identified as an important class. Attacks are generally targeted on specific software products designed for some specific Operating System.

2. Network_Terms: This class was introduced after analyzing the text where it was noted that recently attacks using network and internet are wide spread and thus this class was introduced to extract those terms.

3. Attack: This is an important class which characterizes the information of attack in any given text. Two subclasses were further introduced after seeing the work of (14).

   (a) Means: This class mainly describes the immediate reaction of the system to the input. Basically telling what happens to a system once attacker gives malicious input to system. You will find some information from given example.

   (b) Consequences: This class describes the final result of an attack. An example snippet will describe it better.

4. File_Name: Some attacks are particularly targeted using some specific file name. This it is very important to identify such files mentioned in given text.

5. Hardware: Vulnerabilities can also be present in hardware or in software that are designed for some particular hardware. In those cases, identifying these hardware mentioned in text is equally important. Though our experimental experiences show that mention of any hardware was minimal among the texts but nevertheless there is still a significance of this class.

6. Other_Technical_Terms: There are some technical terms that cannot be accommodated by all the classes that are mentioned but these terms can be useful in some

cases and since they are still technical jargon thus we introduced a class that helps to catch those terms which can't be classified in any other class.

7. NER_Modifier: This class was introduced after analyzing the blogs where it was noted that sometimes text descriptions mentioned more than just one software version and mostly this is written in common language instead of technical words. Taking an example of one such text description:

   "This vulnerability is present in Adobe Acrobat X and earlier versions....."

   In this piece of text, "and earlier versions" is indicating that all the adobe acrobat version before X are also vulnerable to the threat. These words hold key information about various versions that are also vulnerable. NER_Modifier class identifies such text and generally it is followed by Software class or Operating_System subclass.

3.1.3 Annotation

Since the idea is quite novel thus there was no off-the-shelf dataset that can be used for training purpose. Thus the dataset that we mentioned above had manually annotated by domain experts since it is such a task where one requires domain knowledge.
We tested various tools to annotate large corpus of data and finally found brat (1) to be very useful and easy to setup.

Brat is an open source web-based tool used for text annotation. It is an excellent tool for annotation if you have large text like articles to be annotated. One can find the installation instructions at Brat website (1).

Brat was made online so that annotators can annotate online. Different users helped in this annotation. Users were provided with training module along with description of task and system. Since system was easy very to use thus it took little time for training purpose.

Most of the annotators were graduate students from Computer Science Department. We chose Graduate Students from Computer Science department because of their domain knowledge required by this task. Since they are already aware of technical jargon in these articles thus one can rely on their annotations. These annotations will work as gold standard in testing along with for training purposes.

Given figure 3.2 is a screenshot of the brat.

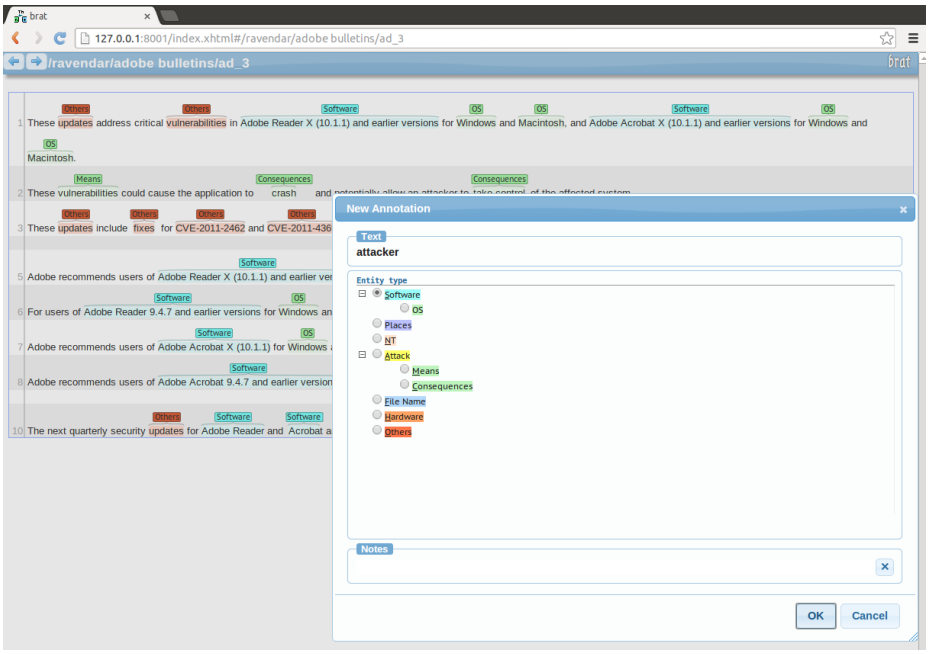

FIG. 3.2. Brat Annotation Tool

3.1.4 Stanford Named Entity Recognizer

Stanford NER (6) is a java based Named Entity Recognizer, part of Stanford NLP toolkit, that is designed for primarily for the task of People, Places and Organizations. It is also known as CRFClassifier since it provides a general implementation of Conditional Random Field (CRF) sequence models (8). It also provides wide variety of feature sets that one can use to train its model. It is provided with base implementation and it can be trained for any kind of data or language.

We chose Stanford Named Entity Recognizer as it provides a perfect base implementation for Conditional Random Fields with wide variety of options for features. Since our task is also to make a named recognizer but cybersecurity domain thus we chose Stanford NER base implementation to train upon.

3.1.5 Feature Set Engineering

Features are those properties that will help to build model. Feature set are combination of features that is used in training model in order to identify those properties from text that characterizes it. Model works based on these features thus it is very important to choose these features carefully.

Choosing features set proved to be a very tedious task. There are over 75 features provided by Stanford NER that can be used to train model on. It didn't provide much documentation for each of these features which makes this task even more complex. Though there are algorithms that helps to identify best features and automate this process of feature set engineering but those algorithms are not generally applied to dataset with large text since requirements from same text can be different. For instance, all the article that are

used for training purpose in our case are written in English Language with some technical jargon thus automating this task of feature set engineering is not a good idea for large text.

So finally, we had to analyze each feature that we understood will be useful for our system. Following are the set of the features that we are using in our system:

1. UseTaggySequences (i.e. HMM of classes): This is very important feature that uses the sequence of classes instead of sequences of words. This Hidden Markov Model (HMM) of classes finds patterns between classes and based on these patterns it

2. UseNGrams: This feature make substrings of word and make features based on those substrings.

3. MaxNGramLeng: This feature defines maximum length of NGram. No substrings can be made bigger than this. In our case, we have defined maxNGramLeng as 6.

4. UsePrev: This results in features based on relationship between current word and a pair (previous word, class of previous word). This feature is very useful if there are classes which generally follows each other.

5. UseNext: This is similar to UsePrev except for the fact that it uses pair for next work.

6. UseWordPairs: ives features based on two pair: (Previous word, current word, class) and (current word, next word, class))

7. UseGazettes: If true, this feature will except a list of gazette list by using following feature.

8. Gazette: This feature is very handy and works like a directory. In this feature, you pass a list of files containing class members and their class type. We used this feature

for the class of Software and Operating System and it results in big improvement for these classes. It was also noted that members of Means and Consequences classes are always not the same and it is highly dependent upon the context.

9. CleanGazette: Stanford NER provides two options to implement Gazette feature.

   - CleanGazette: If true, this feature fires only when whole word is matched in Gazette. For example, if there is a word "Windows XP" in gazette then whole word should be matched in document.
   - SloppyGazette: If true, it fires when even one token is matched in Gazette. For example, if "Windows" is matched with "Windows XP", it will return positive flag for that class.

     For our system, CleanGazette proved to be a very good choice as it increases the results for Operating_System and Software class.

   Note: We have defined important but limited number of features that we have used in our system. Description of all the features can be found at Java Doc of NERFeatureFactory library (3).

3.2 Architecture

Once system was trained, given figure 3.3 represent the architecture of working module:

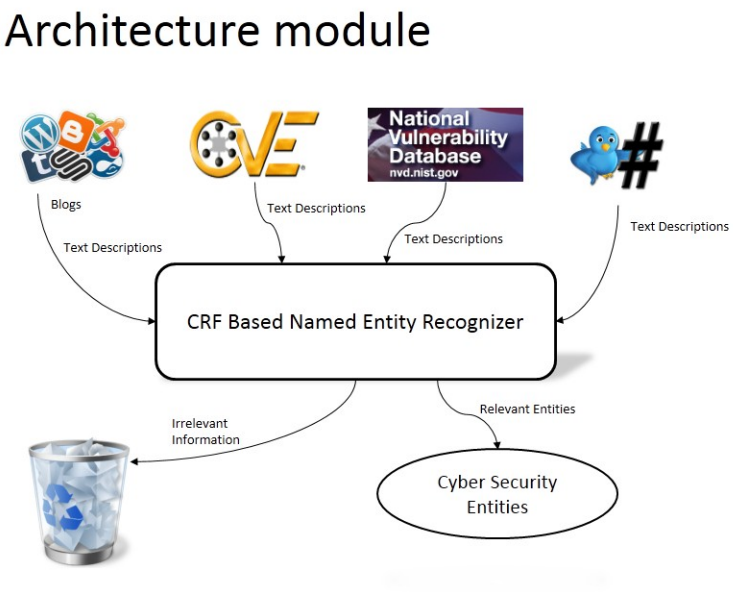

FIG. 3.3. System Architecture

3.3 Middleware - Brat to Stanford NER

Once data was collected there was a problem of integrating the data since the data we get after annotation task on Brat was not compatible with input of Stanford NER. We designed a middle ware that converted annotated data into Stanford NER input for training purpose.

Figure 3.4 shows the architecture for middleware:

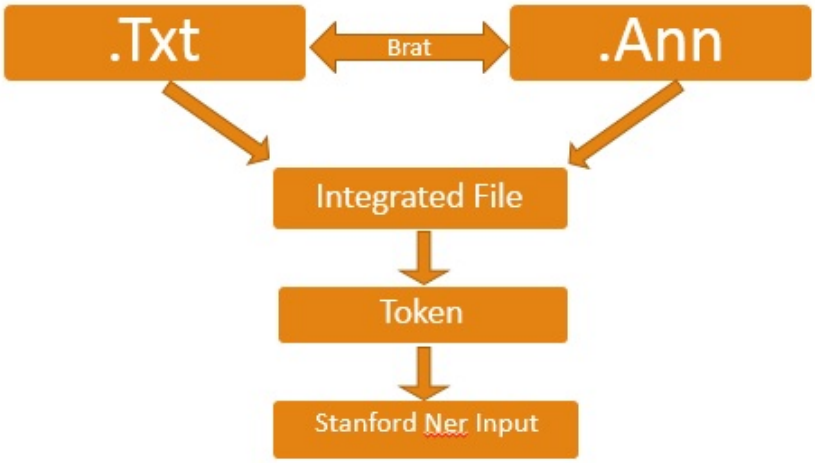

FIG. 3.4. Brat to Stanford NER - Middleware

In the process, we use 5 types of files:

1. Text file and Annotation Files (.Txt and .Ann)
2. Integrated files (.int)
3. Token File (.tok)
4. Stanford NER Input (.tsv)

3.3.1 Text Files and Annotation Files

Text file (.txt) is the text description of an article that we provide to user. Annotation file (.ann) is the annotated file that is made by Brat tool once user annotates the data. Annotation file consists of all the annotations with the starting string, ending string, class name and class member.

Here is an example of output of one annotation file with few entities:

T6 Operating_System 302 311 Windows 7

T7 Operating_System 313 335 Windows Server 2008 R2

T13 Software 538 557 Internet Explorer 6

3.3.2 Integration File

Integration file (.int) integrates the annotated and text files. It is integrated using the delimiters between the class and class member.

Here is one of an example where text file and annotations are integrated in one file:

"This security update is rated Important for all supported editions of
Windows@Vista#OPERATINGSYSTEM,
Windows@Server@2008#OPERATINGSYSTEM,
Windows@7#OPERATINGSYSTEM"

Space between class members is delimited by "@" sign whereas class is delimited by "#" sign. It is done so that classes and class members can be differentiated for next step.

3.3.3 Token File

Token file (.tok) uses integrated file and converts them into tokens. A token program breaks whole file where each word occupies each line. So token file of above example sentence will look like this:

This

security

update

is

rated

Important
for
all
supported
editions
of
Windows@Vista#OPERATINGSYSTEM,
Windows@Server@2008#OPERATINGSYSTEM,
Windows@7#OPERATINGSYSTEM

3.3.4 Stanford NER Input file

Stanford NER has its specific input file format to train on. The whole purpose of this middle ware system was to convert the annotated data into training data for Stanford NER that has its specific input file format for training the system. In its format, each word act as token this is tab-separated by class name and for all those words that has no class, it still has to be specified as null class by some common symbol which in our case is "O".

Following is one example of one such file:

This O
security O
update O
is O
rated O
Important O
for O

all O

supported O

editions O

of O

Windows OPERATINGSYSTEM,

Vista OPERATINGSYSTEM,

Windows OPERATINGSYSTEM,

Server OPERATINGSYSTEM,

2008 OPERATINGSYSTEM,

Windows OPERATINGSYSTEM,

7 OPERATINGSYSTEM,

This is a snipet of input file to Stanford NER. As stated above, first column shows class member and second column shows class.

Chapter 4

RESULTS AND EVALUATION

In order to evaluate system, we decided to evaluate the system in following perspectives:

1. Inter-Annotator Agreement
2. System Performance
3. Performance against other systems

4.1 Inter-Annotator Agreement

Once data was collected with the help graduate students of Computer Science, it is important to verify the quality of annotations. This quality check helps to identify the problems in annotations including for which classes annotators largely agreed upon and for which classes they didn't.

Our experiment is based on the fact sampling method where we took 10% of annotated data and asked other user to annotate it. This sample was taken from various datasets including CVEs, Microsoft Bulletins and Adobe Bulletins. It was strictly imposed that user

who has annotated similar dataset previously will not be annotating the same dataset. So for this experiment, we randomly selected 60 CVES, 12 Microsoft Bulletins and 12 Adobe bulletins. We also selected 6 graduate students to annotate this data. Users for both datasets had equal domain knowledge which was also reflected in the experiments.

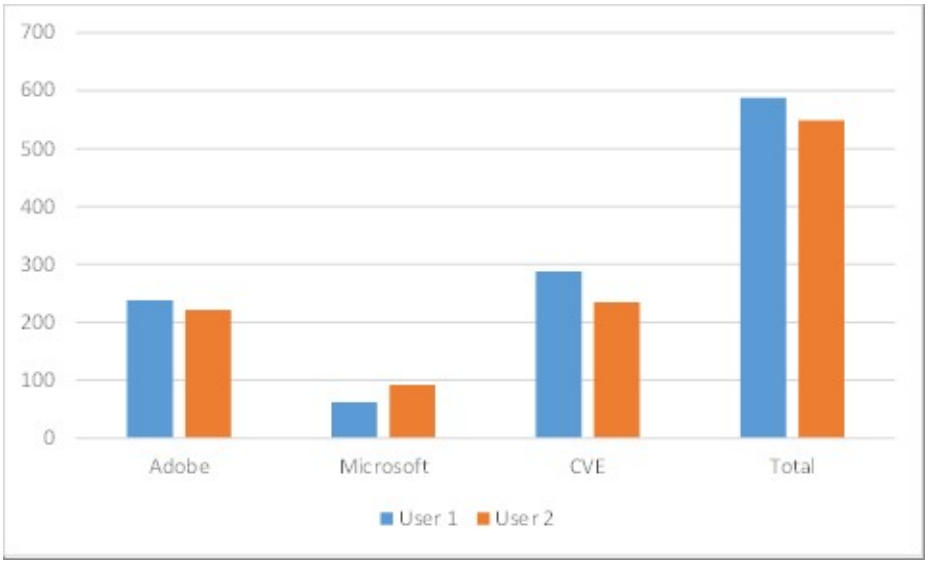

FIG. 4.1. Inter-annotator agreement based on datasets

Given figure 4.1 shows the Inter-annotator agreement between two users based on whole data set selected for experiment purpose.

It was noted users had agreement by in-large as shown by results of individual dataset inter-annotation agreement as well as total dataset agreement.

We also did analysis of Inter-annoation agreement at class level where we tried to evaluate how much users agreed at class level. We evaluated for 4 major classes which are Software, Operating System, Means and Consequences. It was found that there was large agreement on Software and Operating System but there was quite a disagreement in

Means and Consequences but agreement on the class of Means was probably the lowest as it can be noted in given figure 4.2. It was also experienced during annotation phase when annotators found this class of Means very confusing.

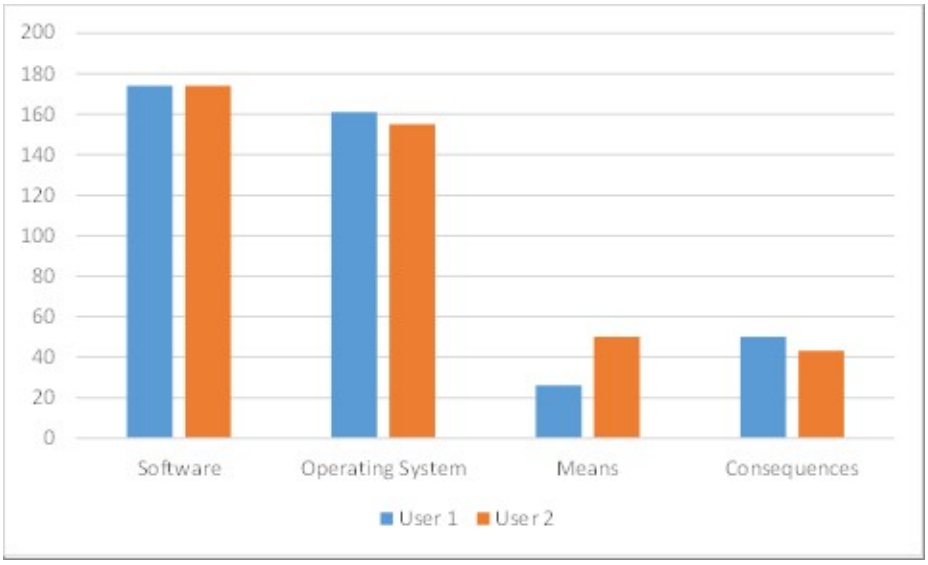

FIG. 4.2. Inter-annotator agreement based on classes

4.2 System Evaluation

In Information Retrieval and extractions systems, evaluation is done by using Precision and recall method. This method basically tells how relevant retrieved information is. Precision, Recall and F measure are defined by true positives, false positives and false negatives. It is very important to understand these terms in order to under evaluation model adopted in our work.

- True Positive: It is a collection of those class members which are correctly labeled as belonging to a particular class. [1]

[1] http://en.wikipedia.org/wiki/Precision_and_recall

- False Positive: It is a collection of those class members which are mislabeled as belonging to a particular class. [2]

- False Negative: It is a collection of those items which are not labeled for any class by system but actually they belong to some class. [3]

Precision is given by:

$$\frac{TruePositives}{TruePositives+FalsePositives}$$

Recall is given by:

$$\frac{TruePositives}{TruePositives+FalseNegatives}$$

F-measure score is given by:

$$\frac{2*Precision*Recall}{Precision+Recall}$$

System was evaluated by comparing the results of a trained module against GOLD standard data - annotated dataset by humans. In training module, we used 80% of annotated data for training purpose and 20% data for testing purpose. To verify the results, we adopted an approach of 4-fold cross validation approach where 4 Cybersecurity Named Entity Recognizers were trained using different chunks of dataset in order to verify the consistency of a system.

Once all the dataset with annotations is converted into Stanford NER input files, we divided the data 5 chunks - 4 for training and 1 for testing. For 4 different systems, we used different combinations of these chunks of data for training and testing purpose. It was made assure that no testing dataset file should be used for training purpose.

[2] http://en.wikipedia.org/wiki/Precision_and_recall
[3] http://en.wikipedia.org/wiki/Precision_and_recall

In following subsections, we will be sharing results of our each trained system based on different datasets as well as a whole system. Training set for all 4 different systems were of equal size.

4.2.1 System 1

We trained our first system using a training set of 38,000 tokens and testing set of 8,300 tokens and over 1200 entities. Shared results are based on performance of system for all entities as well as for different datasets.

Given figure 4.3 shows overall performance by system 1 against testing dataset where as figure 4.4 shows performance of system 1 against various datasets.

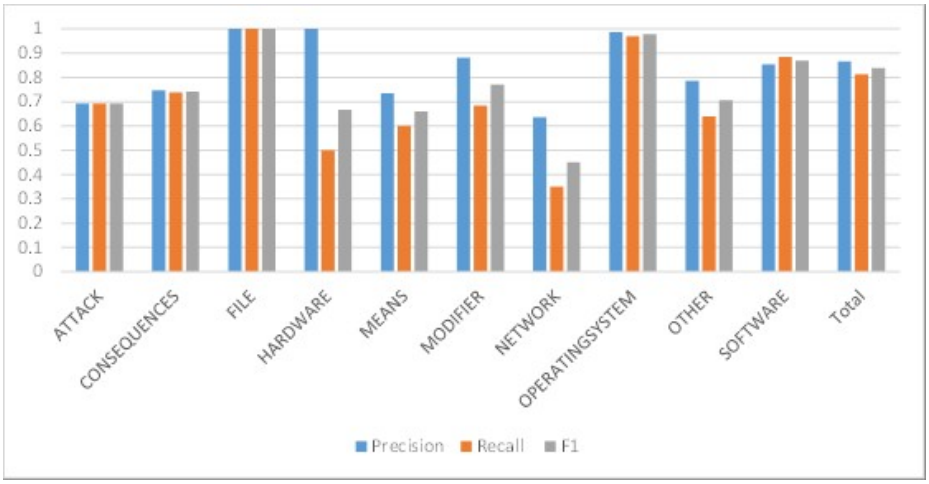

FIG. 4.3. Overall Performance of NER System 1

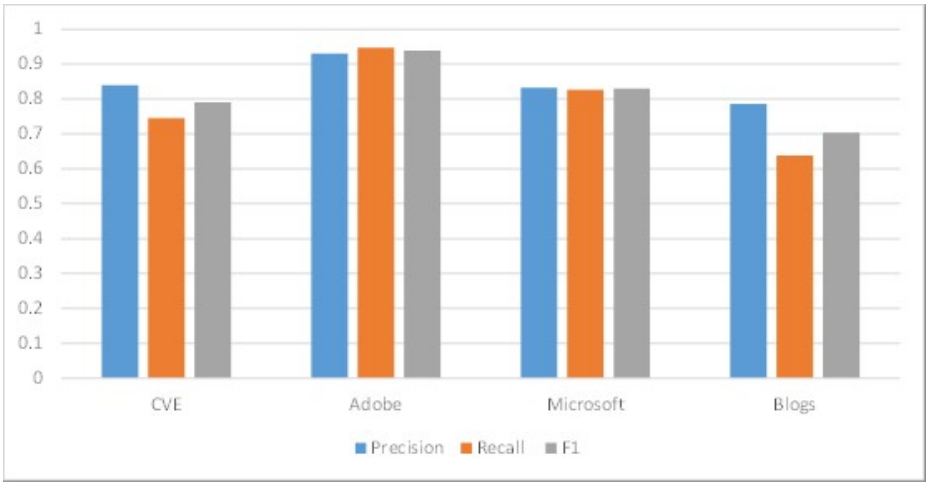

FIG. 4.4. Performance of NER System 1 for various datasets

4.2.2 System 2

In second system, testing dataset consists of over 8000 tokens and 1050 entities.

Given figure 4.5 shows overall performance by system 2 against testing dataset where as figure 4.6 shows performance of system 2 against various datasets.

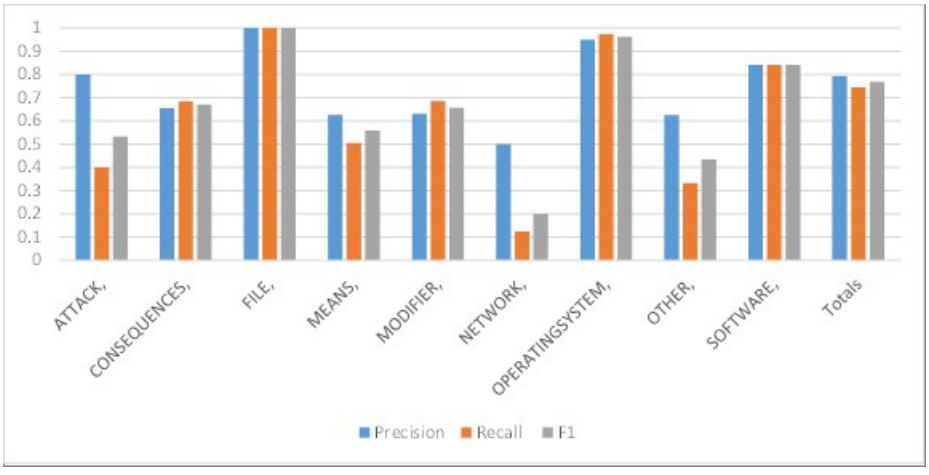

FIG. 4.5. Overall Performance of NER System 2

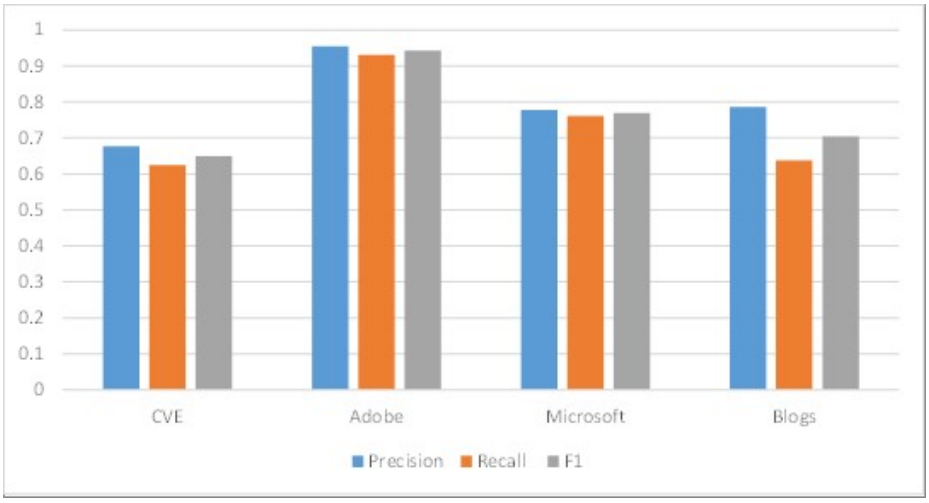

FIG. 4.6. Performance of NER System 2 for various datasets

4.2.3 System 3

For 3rd System, we used the testing dataset consisting of over 8900 tokens with 1150 entities.

Given figure 4.7 shows overall performance by system 3 against testing dataset where as figure 4.8 shows performance of system 3 against various datasets.

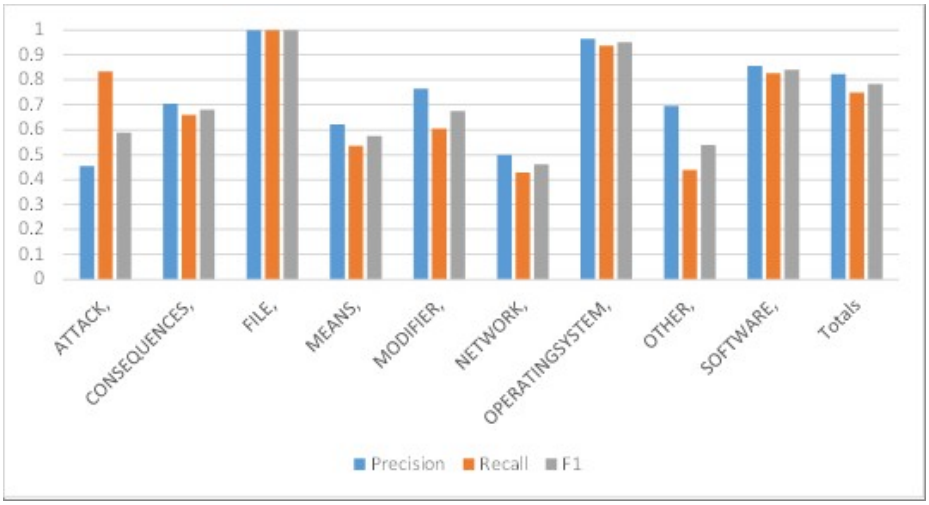

FIG. 4.7. Overall Performance of NER System 3

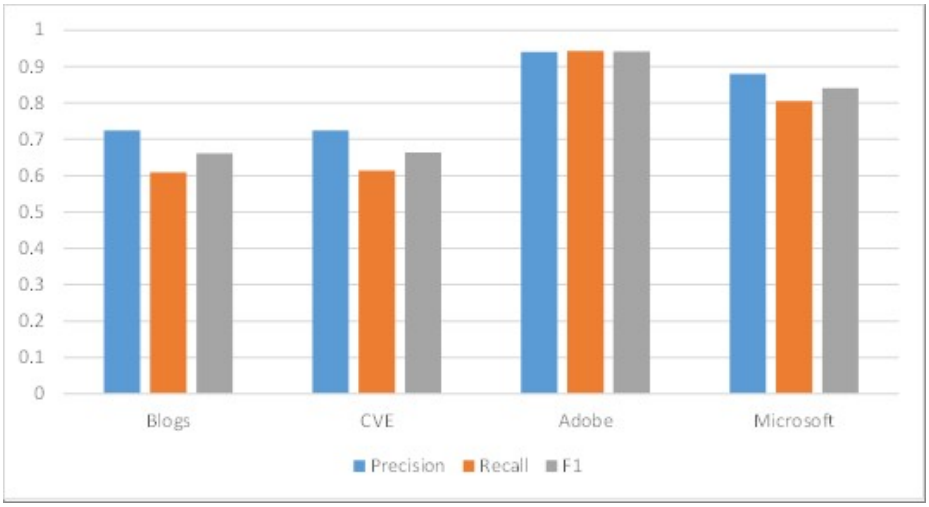

FIG. 4.8. Performance of NER System 3 for various datasets

4.2.4 System 4

For last system, we used the testing dataset consisting of over 9050 tokens and 1158 entities.

Given figure 4.9 shows overall performance by system 4 against testing dataset where as figure 4.10 shows performance of system 4 against various datasets.

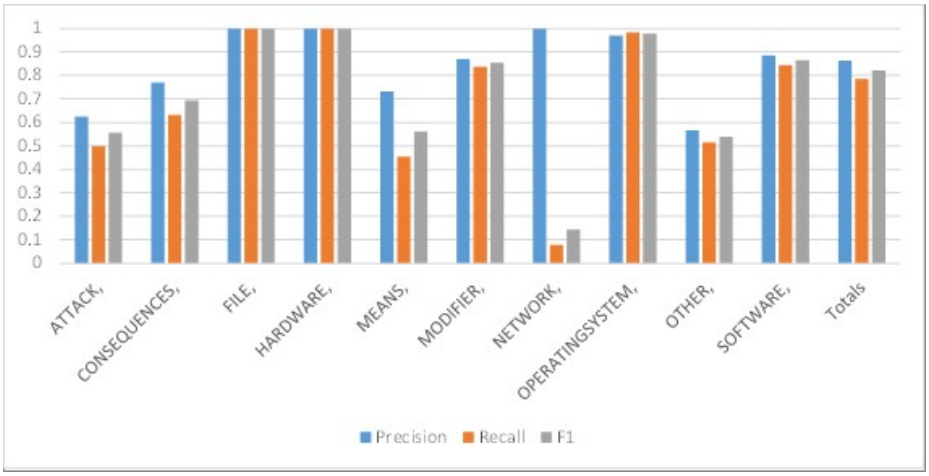

FIG. 4.9. Overall Performance of NER System 4

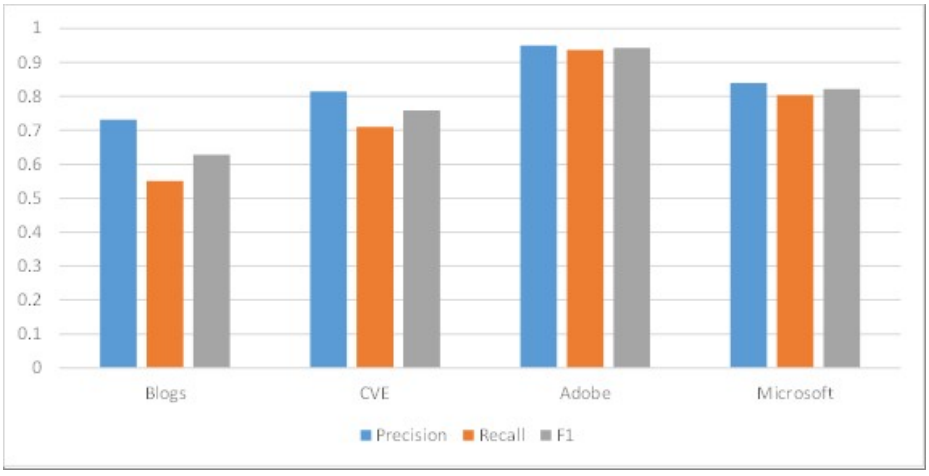

FIG. 4.10. Performance of NER System 4 for various datasets

4.2.5 Analyzing all Systems

It is noted that results are quite consistent in all 4 systems that were trained for cross validation; Precision is in range of 0.78 to 0.86, recall is in range of 0.64 to 0.81 and F1-score is in range of 0.70 to 0.83. Given figure 4.11 compares all 4 systems and also shows average of precision, recall and F1 measure based on 4 systems.

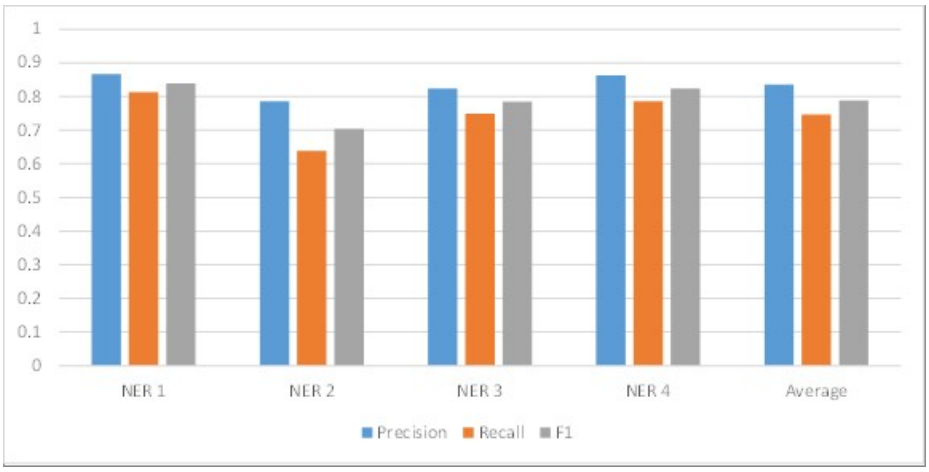

FIG. 4.11. Performance comparison of all 4 systems

Chapter 5

CONCLUSION AND FUTURE WORK

We have described an Information Extraction framework to extract cybersecurity related entities and concepts from any piece of given text. This framework helps to filter relevant information from given text. This enhances the capacity of automating the process of retrieving information from web on cyberattacks and software vulnerabilities.

This system also produced the uniquely annotated dataset that can be used for various other purposes. This dataset can be used to train different models as there are over 9 classes that are used and one can train new system with fewer classes using the same dataset.

We found out that Stanford NER provides a good base implementation of Conditional Random Field (CRF) sequence model with variety of features. We also provided a annotation middle-ware that converts annotated data into Stanford NER input format. This middle-ware is designed to be flexible enough that it can be easily modified to make it useful for any other annotation tool.

In further work, this work can be easily extended in different possible ways. One could be in order to improve results, part of speech (POS) tagging can be used which will give

further information about relationships between different identified entities and concepts. This can be easily done using tools and approaches used in this work and Stanford Toolkit provides libraries for POS processing. Other could be inference and summarization where our work can be further extended to make it capable of inferencing information from any piece of given text. Doing this will help further in automating the process of processing webtext talking cyber security domain.

Appendix A

APPENDIX-TITLE

A.1 Annotation Guidelines

A.2 Properties File

A.3 System validation results in detail

A.3.1 System 1

Table A.1. Overall performance of System 1 on test dataset

Entity	Precision	Recall	F1	TP	FP	FN
ATTACK	0.6923	0.6923	0.6923	9	4	4
CONSEQUENCES	0.7468	0.7375	0.7421	59	20	21
FILE	1	1	1	9	0	0
HARDWARE	1	0.5	0.6667	2	0	2
MEANS	0.7347	0.6	0.6606	36	13	24
MODIFIER	0.8816	0.6837	0.7701	67	9	31
NETWORK	0.6364	0.35	0.4516	7	4	13
OPERATINGSYSTEM	0.9867	0.9696	0.9781	223	3	7
OTHER	0.7869	0.64	0.7059	48	13	27
SOFTWARE	0.854	0.8849	0.8691	269	46	35
Total	0.8668	0.8127	0.8389	729	112	168

Table A.2. Performance of System 1 on Adobe Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	0.7273	0.7273	0.7273	8	3	3
CONSEQUENCES,	0.5556	0.3846	0.4545	5	4	8
FILE,	1	1	1	2	0	0
OPERATINGSYSTEM,	1	1	1	128	0	0
OTHER,	0.6667	0.6667	0.6667	2	1	1
SOFTWARE,	0.9167	1	0.9565	121	11	0
Totals	0.9301	0.9466	0.9383	266	20	15

Table A.3. Performance of System 1 on CVE Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	1	0.5	0.6667	1	0	1
CONSEQUENCES,	0.7407	0.8163	0.7767	40	14	9
FILE,	1	1	1	7	0	0
HARDWARE,	1	0.5	0.6667	2	0	2
MEANS,	0.7778	0.5714	0.6588	28	8	21
MODIFIER,	0.8816	0.6907	0.7746	67	9	30
NETWORK,	0.3333	0.3333	0.3333	1	2	2
OPERATINGSYSTEM,	1	0.9348	0.9663	43	0	3
OTHER,	0.8333	0.625	0.7143	35	7	21
SOFTWARE,	0.8182	0.8438	0.8308	81	18	15
Totals	0.8402	0.7457	0.7902	305	58	104

Table A.4. Performance of System 1 on Microsoft Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
CONSEQUENCES,	0.875	0.875	0.875	14	2	2
MEANS,	0.7273	0.8889	0.8	8	3	1
NETWORK,	0.8571	0.4	0.5455	6	1	9
OPERATINGSYSTEM,	0.9744	0.9268	0.95	38	1	3
OTHER,	0.7333	0.7333	0.7333	11	4	4
SOFTWARE,	0.7708	0.881	0.8222	37	11	5
Totals	0.8321	0.8261	0.8291	114	23	24

A.3.2 System 2

Table A.5. Overall performance of System 2 on test dataset

Entity	Precision	Recall	F1	TP	FP	FN
ATTACK,	0.8	0.4	0.5333	4	1	6
CONSEQUENCES,	0.6556	0.686	0.6705	59	31	27
FILE,	1	1	1	9	0	0
MEANS,	0.625	0.5056	0.559	45	27	44
MODIFIER,	0.6301	0.6866	0.6571	46	27	21
NETWORK,	0.5	0.125	0.2	1	1	7
OPERATINGSYSTEM,	0.9512	0.975	0.963	156	8	4
OTHER,	0.625	0.3333	0.4348	20	12	40
SOFTWARE,	0.8413	0.8413	0.8413	265	50	50
Totals	0.794	0.7442	0.7683	605	157	208

Table A.6. Performance of System 2 on Adobe Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	1	0.5714	0.7273	4	0	3
CONSEQUENCES,	0.5556	0.625	0.5882	5	4	3
FILE,	1	1	1	1	0	0
MEANS,	1	1	1	1	0	0
OPERATINGSYSTEM,	0.9912	0.9826	0.9869	113	1	2
OTHER,	1	0.5	0.6667	1	0	1
SOFTWARE,	0.9489	0.9774	0.963	130	7	3
Totals	0.9551	0.9307	0.9427	255	12	19

Table A.7. Performance of System 2 on CVE Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	0	0	0	0	1	3
CONSEQUENCES,	0.6515	0.6825	0.6667	43	23	20
FILE,	1	1	1	8	0	0
MEANS,	0.5833	0.4861	0.5303	35	25	37
MODIFIER,	0.6571	0.697	0.6765	46	24	20
NETWORK,	0	0	0	0	1	5
OPERATINGSYSTEM,	0.8824	0.8824	0.8824	15	2	2
OTHER,	0.7368	0.3415	0.4667	14	5	27
SOFTWARE,	0.7009	0.7353	0.7177	75	32	27
Totals	0.6762	0.6243	0.6492	236	113	142

Table A.8. Performance of System 2 on Microsoft Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
CONSEQUENCES,	0.7333	0.8462	0.7857	11	4	2
MEANS,	0.9	0.6429	0.75	9	1	5
NETWORK,	1	1	1	1	0	0
OPERATINGSYSTEM,	0.8667	1	0.9286	13	2	0
OTHER,	0.4545	0.3125	0.3704	5	6	11
SOFTWARE,	0.8857	0.8857	0.8857	31	4	4
Totals	0.7778	0.7609	0.7692	70	20	22

A.3.3 System 3

Table A.9. Overall performance of System 3 on test dataset

Entity	Precision	Recall	F1	TP	FP	FN
ATTACK,	0.4545	0.8333	0.5882	5	6	1
CONSEQUENCES,	0.7045	0.6596	0.6813	62	26	32
FILE,	1	1	1	16	0	0
MEANS,	0.6216	0.5349	0.575	46	28	40
MODIFIER,	0.7639	0.6044	0.6748	55	17	36
NETWORK,	0.5	0.4286	0.4615	3	3	4
OPERATINGSYSTEM,	0.9648	0.9366	0.9505	192	7	13
OTHER,	0.6957	0.4384	0.5378	32	14	41
SOFTWARE,	0.8559	0.8267	0.841	291	49	61
Totals	0.8239	0.7484	0.7844	702	150	236

Table A.10. Performance of System 3 on Adobe Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	0.4	1	0.5714	4	6	0
CONSEQUENCES,	0.4444	0.5	0.4706	4	5	4
FILE,	1	1	1	9	0	0
OPERATINGSYSTEM,	0.9837	0.9918	0.9878	121	2	1
OTHER,	0.6	0.6	0.6	3	2	2
SOFTWARE,	0.9752	0.9691	0.9721	157	4	5
Totals	0.9401	0.943	0.9415	298	19	18

Table A.11. Performance of System 3 on CVE Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	1	0.5	0.6667	1	0	1
CONSEQUENCES,	0.6866	0.6389	0.6619	46	21	26
FILE,	1	1	1	7	0	0
MEANS,	0.6094	0.5417	0.5735	39	25	33
MODIFIER,	0.7639	0.6111	0.679	55	17	35
NETWORK,	1	0.3333	0.5	1	0	2
OPERATINGSYSTEM,	0.9259	0.7353	0.8197	25	2	9
OTHER,	0.6786	0.413	0.5135	19	9	27
SOFTWARE,	0.7255	0.6916	0.7081	74	28	33
Totals	0.7236	0.6138	0.6642	267	102	168

Table A.12. Performance of System 3 on Microsoft Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
CONSEQUENCES,	1	1	1	12	0	0
MEANS,	0.875	0.7	0.7778	7	1	3
NETWORK,	0.4	1	0.5714	2	3	0
OPERATINGSYSTEM,	0.9706	0.9706	0.9706	33	1	1
OTHER,	0.8333	0.4762	0.6061	10	2	11
SOFTWARE,	0.8378	0.8158	0.8267	31	6	7
Totals	0.8796	0.8051	0.8407	95	13	23

A.3.4 System 4

Table A.13. Overall performance of System 4 on test dataset

Entity	Precision	Recall	F1	TP	FP	FN
ATTACK,	0.625	0.5	0.5556	5	3	5
CONSEQUENCES,	0.7692	0.6316	0.6936	60	18	35
FILE,	1	1	1	11	0	0
HARDWARE,	1	1	1	1	0	0
MEANS,	0.7317	0.4545	0.5607	30	11	36
MODIFIER,	0.871	0.8372	0.8538	108	16	21
NETWORK,	1	0.0769	0.1429	1	0	12
OPERATINGSYSTEM,	0.9704	0.985	0.9777	197	6	3
OTHER,	0.5672	0.5135	0.539	38	29	36
SOFTWARE,	0.8861	0.8443	0.8647	358	46	66
Totals	0.8625	0.7862	0.8226	809	129	220

Table A.14. Performance of System 4 on Adobe Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	0.6	0.4286	0.5	3	2	4
CONSEQUENCES,	0.625	0.5556	0.5882	5	3	4
FILE,	1	1	1	3	0	0
OPERATINGSYSTEM,	0.987	0.9935	0.9902	152	2	1
SOFTWARE,	0.9563	0.9722	0.9642	175	8	5
Totals	0.9494	0.9363	0.9428	338	18	23

Table A.15. Performance of System 4 on CVE Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
ATTACK,	0.6667	0.6667	0.6667	2	1	1
CONSEQUENCES,	0.7222	0.5821	0.6446	39	15	28
FILE,	1	1	1	8	0	0
HARDWARE,	1	1	1	1	0	0
MEANS,	0.7097	0.4314	0.5366	22	9	29
MODIFIER,	0.9	0.8438	0.871	108	12	20
NETWORK,	1	0.1429	0.25	1	0	6
OPERATINGSYSTEM,	0.5	1	0.6667	1	1	0
OTHER,	0.619	0.5	0.5532	26	16	26
SOFTWARE,	0.8481	0.8221	0.8349	134	24	29
Totals	0.8143	0.7095	0.7583	342	78	140

Table A.16. Performance of System 4 on Microsoft Bulletins Dataset

Entity	P	R	F1	TP	FP	FN
CONSEQUENCES,	1	0.9412	0.9697	16	0	1
MEANS,	1	0.6154	0.7619	8	0	5
OPERATINGSYSTEM,	1	1	1	31	0	0
OTHER,	0.6316	0.75	0.6857	12	7	4
SOFTWARE,	0.75	0.75	0.75	27	9	9
Totals	0.8393	0.8034	0.821	94	18	23

REFERENCES

[1] Brat nlp annotatiol tool. http://brat.nlplab.org/index.html.

[2] Cyber security market research. http://www.marketresearchmedia.com/?p=206.

[3] Ner feature factory. http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/NERFeatureFactory.html.

[4] Opencalais.

[5] FARUQUI, M., PADÓ, S., AND SPRACHVERARBEITUNG, M. Training and evaluating a German named entity recognizer with semantic generalization. *g Semantic Approaches in Natural Language Processing* (2010), 129.

[6] FINKEL, J. R., GRENAGER, T., AND MANNING, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (2005), Association for Computational Linguistics, pp. 363–370.

[7] JIAO, D., AND WILD, D. J. Extraction of cyp chemical interactions from biomedical literature using natural language processing methods. *Journal of chemical information and modeling 49*, 2 (2009), 263–269.

[8] LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning* (2001), Morgan Kaufmann, pp. 282–289.

[9] LOWIS, L., AND ACCORSI, R. On a classification approach for soa vulnerabilities. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International* (2009), vol. 2, IEEE, pp. 439–444.

[10] MORE, S., MATTHEWS, M., JOSHI, A., AND FININ, T. A knowledge-based approach to intrusion detection modeling. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on* (2012), IEEE, pp. 75–81.

[11] REHMAN, S. Software Design Level Vulnerability Classification Model Full text.

[12] SYED, Z., FININ, T., MULWAD, V., AND JOSHI, A. Exploiting a web of semantic data for interpreting tables. In *Proceedings of the Second Web Science Conference* (2010).

[13] SYED, Z. S., AND ADVISER-FININ, T. W. *Wikitology: a novel hybrid knowledge base derived from wikipedia*. University of Maryland at Baltimore County, 2010.

[14] UNDERCOFFER, J., PINKSTON, J., JOSHI, A., AND FININ, T. A target-centric ontology for intrusion detection. In *18th International Joint Conference on Artificial Intelligence* (2004), pp. 9–15.