A Supervised Learning Algorithm for Information Extraction from Textual Data

Tianhao Wu, Lars E. Holzman, William M. Pottenger and Daniel J. Phelps
Computer Science and Engineering at Lehigh University and Eastman Kodak Company
{tiw2, leh7, billp}@lehigh.edu, daniel.phelps@kodak.com

Abstract

In this article we present a supervised learning algorithm for the discovery of finite state automata in the form of regular expressions in textual data. The automata generate languages that consist of various representations of features useful in information extraction. We have successfully applied this learning technique in the extraction of textual features from police incident reports [2]. In this article we present the result of the application of our algorithm in extraction of the 'problem solved' in patents. The 'problem solved' in a patent identifies the particular solution to an insufficiency in prior art that the patent addresses.

1. Introduction

Regular expressions can be used as patterns to extract features from semi-structured and narrative text [1]. After studying hundreds of police incident reports and patents, we find that regular expressions can be readily employed to express patterns of features. For example, in a police incident report a suspect's height might be recorded as "{CD} feet {CD} inches tall", where {CD} is the part of speech tag for a numeric value. Alternatively, if a sentence in a patent matches the regular expression "invention .* (to)? .* improve", then it likely contains information about the particular solution that addresses an insufficiency in prior art. We have developed an algorithm for automatic discovery of regular expressions of this nature.

At Lehigh University we are conducting information extraction research in collaboration with the Information Mining Group at Eastman Kodak Company and Lockheed Martin M&DS in conjunction with the Pennsylvania State Police.

In our work with Lockheed Martin for the Pennsylvania State Police, our target is to develop a system that extracts features related to criminal modus operandi and physical description for suspects as recorded in narrative incident reports. Our results in [2] demonstrate that our algorithm has good testing performance on ten features important in homeland defense.

Our goal in the work with the Eastman Kodak Company is to develop technology capable of automatically extracting sentences in patents that identify the problem that a given patent addresses. We term sentences of this nature *problem solved identifiers* (PSIs). This is an important domain given the commercial value of information automatically extracted from patents [3].

This article is organized as follows. In section 2, we summarize related work. Following this, we provide definitions in section 3. In section 4, we present our regular expression discovery algorithm. Following this in section 5 we discuss our preliminary experimental results in the patent domain. We also provide a summary of the result of applying our regular expression discovery algorithm to police incident reports from [2]. Finally, we discuss conclusions and future work in section 6 and acknowledge those who have contributed to this work in section 7.

2. Related Work

Although much work has been done in the field of information extraction, relatively little has focused on the automatic discovery of regular expressions. In this section, we highlight a few efforts that are related to regular expression discovery. We also touch on related work in citation analysis of patents.

Stephen Soderland developed a supervised learning algorithm, WHISK [1], which uses regular expressions as patterns to extract features from semi-structured and narrative text. In each iteration of the learning process, WHISK requires that a human expert label specific features in instances and then generates rules based on these labels. WHISK uses segments such as clauses, sentences, or sentence fragments as its instances. A crucial difference between WHISK and our approach is that WHISK requires the user to identify the precise location of features for labeling while our approach requires only that instances be labeled. As noted this represents a significant reduction in the effort required to develop a training set.

Eric Brill [7] applied his transformation-based learning (TBL) framework to learn reduced regular expressions for correction of grammatical errors in text. Although Brill does not perform explicit information extraction, the correction process involves identifying grammatical errors. There are three major differences between Brill's approach and ours. First, the reduced regular expressions generated by Brill do not include the logical "OR" operator. We have found that the "OR" operator is necessary to achieve high accuracies in information extraction. Secondly, like the aforementioned work by Soderland, Brill's approach requires intensive feature-specific labeling to create the ground truth used in TBL. Finally, our approach does not require domain experts to create templates because it is not based on TBL

Michael Chau, Jennifer J. Xu, and Hsinchun Chen have published results of research on extracting entities from narrative police reports [13]. They employed a neural network to extract persona names, addresses, narcotic drugs, and items of personal property from these reports. Noun phrases are candidates for name entities. Although not readily apparent in [13], they evidently employ a similar approach to other researchers in that feature-specific labeling is required in training set development. Their cross-validation results vary from a low of 46.8% to a high of 85.4% for various entities. In our approach, however, we achieve significantly better results without limiting ourselves to noun phrases. In addition, we are able to extract a larger number of features that can be used for analysis in several ways, including matching on modus operandi.

In [2] we describe our approach to feature extraction from police incident reports. We note that our algorithm is *semi-supervised* because it requires significantly less effort to develop a training set than other approaches. Instead of labeling the exact location of features in a training set, the training set developer need only record whether a specific feature of interest occurs in a segment. From this training data our algorithm automatically discovers regular expressions that can be used on previously unseen data for information extraction.

Similarly, PSI extraction is a semi-supervised classification problem. Instead of requiring that the domain expert exactly identify the textual feature or features that describe the problem solved, the training set developer need only determine whether a sentence includes problem solved information or not. One difference between these two applications is the method of segmentation. In our previous work we found it necessary to break sentences into segments using periods and commas as segment boundary markers. In PSI extraction, however, we determined that sentences form natural segment boundaries.

Much work has been done in patent citation analysis (e.g., [4] [5]). For example, patent citation frequencies are employed to ascertain the relative importance of patents. An approach of this nature, however, does not shed light on the content of the patent. To the best of our knowledge, our effort to develop technology to extract content-based PSIs from patents is novel.

3. Definitions

In this section, we start with the standard definition of a regular expression, and then define a reduced regular expression as used in our algorithm. Following this, we define terms used in this article.

Regular expression: "Given a finite alphabet Σ, the set of regular expressions over that alphabet is defined as:

1) $\forall a \in \Sigma$, a is a regular expression and denotes the set {a}.
2) if r and s are regular expressions denoting the languages R and S, respectively, then (r+s), (rs), and (r*) are regular expressions that denote the sets R $\cup$ S, RS and R* respectively." [6, 7]

Reduced regular expression (RRE): Our reduced regular expression is at first glance similar to that defined in [6]. However, there are some significant differences. Given a finite alphabet Σ, our reduced regular expression is defined as a set, where the *star* '*' indicates that the character immediately to its left may be repeated any number of times, including zero, and the question mark '?' indicates that the character immediately to its left may be repeated either zero times or one time.

- $\forall a \in \Sigma$, a is a RRE and denotes the set {a}.
- ~a* is a RRE and denotes the Kleene closure of the set $\Sigma - a$.
- $\wedge \in \Sigma$, $\$ \in \Sigma$, where ^ is the start of a line, and $ is the end of a line.
- $\{\backslash s, \backslash S, \backslash w, \backslash W\} \subset \Sigma$, where \s ([\t\n\r\f]) is any white space, \S ([^ \t\n\r\f]) is any character except white space, \w ([0-9a-zA-Z]) is any alphanumeric character, and \W ([^0-9a-zA-Z]) is any non-alphanumeric character.
- All words in the lexicon and all part of speech tags in the Penn tag corpus [8] belong to ? .
- (\w)* is a RRE denoting the Kleene closure of the set {\w}.
- (\w){i,j} is a RRE denoting that \w is repeated between i and j times, where i=0, and j=i.
- a? is a RRE and denotes that a is an optional part of the RRE.
- if r and s are RREs denoting the languages R and S, respectively, then (r+s) and (rs) are RREs that denote the sets R $\cup$ S and RS, respectively.

Some examples of regular expressions that are not RREs are: "a*", "(ab)*", and "a+". We have not found it necessary to support such regular expressions to achieve high accuracies.

Feature: A feature is the smallest unit of information extracted. Examples include values for the attributes *height, weight, age, gender, time, location, PSI,* etc.

Segment: A segment is (a portion of) a sentence. As noted previously, we found it necessary to use periods to mark sentence boundaries and commas to mark segment boundaries. The only exception is a comma that separates two numbers, as in "$1,000". The comma in the phrase "…in his twenties, with brown eyes…", on the other hand, is a segment boundary. The textual string between any two segment boundaries is a segment.

Item: An item is a document from which features are extracted. In the experiments described herein this is either a police incident report or a full text patent.

True set: If the system is learning a RRE for a feature *f*, then the true set consists of all segments labeled *f* in training set. After each iteration in learning, the true set is updated by removing those segments that have been covered. In this sense our approach uses a covering algorithm.

False set If the system is learning a RRE for a feature *f*, then the false set consists of all segments that are not labeled *f* in the training set. For a given feature *f*, the false set does not change during learning.

Element: Words in the RRE with frequency in the true set higher than a threshold e_{Word} and part of speech tags in the RRE with frequency in the true set higher than a threshold e_{Tag} are termed elements of the RRE.

Root: We term the first element found by the algorithm in an RRE the root of the RRE.

"AND" learning process: All learning iterations that employ the logical "AND" operator.

"OR" learning process: All learning iterations that employ the logical "OR" operator.

"NOT" learning process: All learning iterations that employ the logical "NOT" operator.

"Optional" learning process: All learning iterations that employ an optional operator.

R_{and}: The RRE learned after completion of the "AND" learning process.

N: The number of elements in R_{and}.

S: S is the set of words in the lexicon employed in our approach combined with the part of speech tags in the Penn tag set [8]. |S| is the total number of words and tags in S.

4. Approach

In this section we present our approach to the discovery of RREs from a small set of labeled training segments. The process begins with the separation of the input text into segments. This step is performed automatically. Next, a domain expert labels segments. Finally, we apply a greedy algorithm to discover RREs for both applications. We then perform 10-fold cross-validation to evaluate the performance of the model. We detail these steps in what follows.

4.1. Pre-Processing

Each item is split into segments at this stage. Each segment becomes an instance in our system. We use the technique presented in [9] to detect sentence boundaries.

4.2. Segment Labeling

During training set development, each segment is evaluated manually and assigned the proper labels. After labeling, each feature has its own true set and false set.

4.3. Part Of Speech Tagging

Part of speech tags are also used in our reduced regular expression discovery algorithm. Each word in the training set must be assigned its correct part of speech tag before the learning process begins. Currently, we are using Eric Brill's part of speech tagger to tag our training sets [12]. Brill's tagger uses the Penn tag set [8] (Table 1). We have enhanced the lexicon to include extra tags for feature extraction from police incident reports [2] (Table 2). However, we use the original lexicon in PSI extraction.

Tag	Category	Example
CD	Cardinal number	3, fifteen
IN	Preposition	in, for
PRP$	Determiner, possessive	their, your
DT	Determiner, article	a, the
JJ	Adjective	happy, bad
NN	Noun, singular	aircraft, data
NNP	Proper Noun	London, Reston

VBG	Verb, present participle	taking, living

Table 1: Example tags from Penn tag set

Tag	Category	Example
CDS	Plural number	Teens, twenties
GN	Gender	man, boy
WEEKDAY	Weekday	Monday, Tuesday

Table 2: Examples of our own tags

4.4. Learning Reduced Regular Expressions

The goal of our algorithm is to discover sequences of words and/or part of speech tags that have high frequency in a collection of segments, while having low frequency outside the segments. The algorithm first discovers the most common element of an RRE, termed the root of the RRE. The algorithm then extends the 'length' of the RRE in the "AND" learning process. During the "OR" learning process, the 'width' of the RRE is extended. Next, optional elements are discovered during the "Optional" learning process. The algorithm then proceeds with the "NOT" learning process, and finally discovers the start and the end of the current RRE. Figure 1 depicts the entire learning process.

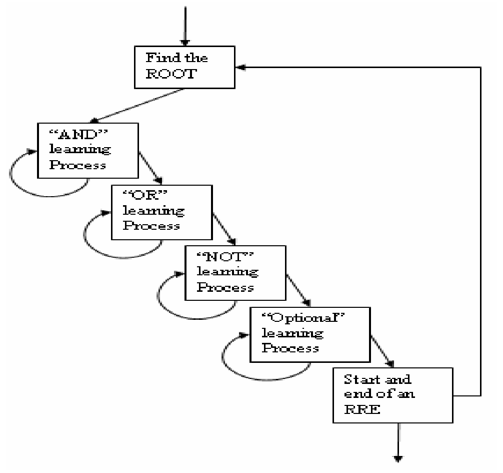

Figure 1: RRE Discovery

We use an example in Figure 2 to illustrate our algorithm in details through the rest of this section. The characters between "{" and "}" are part-of-speech tags.

Our approach employs a covering algorithm. After one RRE is discovered, the algorithm removes all segments covered by the RRE from the true set. The remaining segments become a new true set and the steps in Figure 1 repeat. The learning process stops when the number of segments left in the true set is less than or equal to a threshold d . We use this threshold because over-fitting results if too few segments are used to discover a RRE. d is a parameter in our system, and is currently set to two. We depict the details of each step of the algorithm in what follows.

Our approach is a semi-supervised learning for feature extraction from police incident reports. Instead of labeling the exact location of features in a training set, the training-set developer need only record whether a specific feature of interest occurs in a sentence segment. The rules learned by the algorithm are for exact feature extraction from police reports. Therefore, the approach is a semi-supervised learning in this case. On the other hand, the approach is a supervised learning for PSI because PSI does not require exact match. Training-set developer tags whether a sentence is a PSI or not. The rules learned by the algorithm are also used to tag whether a sentence is a PSI or not. Thus, PSI is a supervised classification problem.

```
in/{IN} her/{GN} twenties/{CDS}
in/{IN} his/{GN} forties/{CDS}
in/{IN} his/{GN} early/{JJ} teens/{CDS}
```

True set

```
a/{DT} man/{GN} in/{IN} his/{GN} car/{NN}

in/{IN} the/{DT} roaring/{VBG} twenties/{CDS}

in/{IN} the/{DT} Reston/{NNP} area/{NN}
……
```

False set

Figure 2: Example training set for feature *Age*

4.4.1. Discovering the Root of a RRE

This step matches each word and/or part of speech tag (specified as a simple RRE) in each true set segment against all segments. The performance of each such RRE in terms of F-measure (Equation 1 from [10]) is considered. In this formula, P = precision = TP/(TP+FP) and R = recall = TP/(TP+FN), where TP are true positives, FP are false positives, and FN are false negatives. The parameter b enables us to place greater or lesser emphasis on precision, depending on our needs.

$$F_b = \frac{(b^2+1)PR}{b^2P+R}$$

Equation 1: F-measure

The element with the highest score is chosen as the 'root' of the RRE. The algorithm discovers a word or part of speech tag that has a high frequency of occurrence in segments with the desired feature. It must also have low frequency in segments that do not contain the desired feature.

Our approach places less emphasis on precision and more on recall during the root discovery process. We use the parameter b_{Root} to control this. Naturally this results in a larger set of segments that match the root. These segments, however, are not necessarily all true positives. As a result, the "AND", "OR", and "NOT" learning phases all prune false positives from the set of strings that match the root RRE. The result is both high precision and high recall. For the example in Figure 2, the root of the RRE is {CDS}.

4.4.2. "AND" Learning Process

After the root is discovered, the algorithm inserts additional words and/or part of speech tags before and after the root. The algorithm places new candidate elements immediately before and after the root, thereby forming two new RREs. Any element (other than the root itself) can be used in this step. The RRE with the highest F-measure score will replace the previous RRE.

The new RRE is then extended in the same way. Adjacency implies the use of an "AND" operator. As before, candidate elements are inserted into all possible positions in the RRE. The algorithm measures the performance of each new RRE and the one with the highest score is selected if its score is greater than or equal to the previous best score. In this sense our algorithm is greedy.

The overall complexity of the "AND" learning process depends on both the number of candidate elements for a position in a given pass during the "AND" learning phase, and on the number of elements in R_{and}. If there are N (N>0) elements in a RRE, then there are N+1 possible positions to insert a new element. For example, if the initial "AND" learning phase learns "in {CDS}" as a RRE, then "$\underline{0}$", "$\underline{1}$", and "$\underline{2}$" in "$\underline{0}$ in $\underline{1}$ {CDS} $\underline{2}$" mark the three possible positions for the second "AND" learning pass to insert elements.

Equation 2 depicts C_{AND}, the complexity of the "AND" process, where S_{ij} is the set of candidate elements for the jth position in the ith pass of the "AND" learning process in a single iteration of Figure 1. In the previous example, S_{21} is the set of candidates that can be used in position $\underline{1}$ in another pass through the "AND" learning process that produced "$\underline{0}$ in $\underline{1}$ {CDS} $\underline{2}$". There are N passes to discover a RRE of size N, and $0 \le |S_{ij}| \le |S|$. This implies that $0 \le C_{AND} \le |S|\left(\frac{N(N+1)}{2}+N\right)$. Therefore, $O(0) \le C_{AND} \le O(N^2)$.

$$C_{AND} = \sum_{i=1}^{N}\sum_{j=0}^{i} |S_{ij}|$$

Equation 2: "AND" Learning Complexity

To improve the performance of the algorithm, we limit the candidates for each position to actual elements that occur in segments. For example, suppose we want to extend "in {CDS}" by inserting an element "_" between "in" and "{CDS}". The system matches "in .* {CDS}" against all segments in the current true set using only those elements that actually occur between "in" and "{CDS}". This approach limits candidate elements to a small set, for the example, the candidates are "his", "their", "twenties", "forties", "only", "teens", {IN}, {PRP$}, {GN}, and {JJ}. Moreover, the number of candidate elements becomes smaller as the RRE grows. To avoid the case in which N is too long for a certain RRE, we use a parameter l to limit N such that $N \le l$.

As noted, during the "AND" learning phase, our algorithm weights precision more heavily than recall. Since we employ a covering algorithm, our goal is to maximize precision during each iteration of Figure 1. Thus we employ different values of b during training to ensure that the resulting RRE is optimal. The parameter b_{Others} controls the b value used in the "AND", "OR", "NOT", and optional learning processes. For the example in Figure 2, the following iterations show how R_{and} is learned. "{CDS}"➔ "in {CDS}"➔ "in{IN} {CDS}"➔ "in{IN} his {CDS}"➔ "in{IN} his{GN} {CDS}".

4.4.3. "OR" Learning Process

After the "AND" learning process is complete, the algorithm extends the RRE with candidate elements using the "OR" operator. For each element discovered during the "AND" learning process, the algorithm uses the "OR" operator to combine it with other candidate elements. If the newly discovered RRE has a better F-measure than the previous RRE, the new RRE will replace the old one.

For the example, suppose "in{IN} his{GN} {CDS}" has an F-measure of 0.78. If "in{IN} (his|her){GN} {CDS}" has a score of 1, then "in{IN} (his|her){GN} {CDS}" will replace "in{IN} his{GN} {CDS}" as the current RRE. After that, "in{IN} his{GN} {CDS}" will be evaluated as an extension to element "his". The extension of element "his" stops when no higher score can be found after all candidate elements have been evaluated. Every element in R_{and} will be extended in turn. In the example, no other elements in R_{and} could be extended except "his". Therefore, the RRE after "OR" learning process is "in{IN} (his|her){GN} {CDS}". As is the case in the "AND" learning process, for performance reasons the selection of candidate elements for the "OR" learning process is also data driven.

C_{OR}, the time complexity of the "OR" learning process, is depicted in Equation 3, where Q_i is the number of candidate elements in position i (each element in R_{and} is a position for the purposes of the "OR" learning process). $0 \le |Q_i| \le |S|$ ➔ $0 \le C_{OR} \le |S|N$ ➔ $O(0) \le C_{OR} \le O(N)$

$$C_{OR} = \sum_{i=1}^{N} Q_i$$

Equation 3: "OR" Learning Complexity

4.4.4. "Optional" Learning Process

The gap between any two adjacent elements in R_{and} is a position for an optional element in a RRE. Therefore, there are N-1 positions that could have optional elements. Each gap can have zero or one optional element. For each gap, the system generates a set of candidate elements using a similar method as that described in section 4.4.2. From the candidate set, the system selects one element that occurs most often in the true set. If the frequency is higher than a threshold γ, the element becomes an optional part of the RRE. γ is a parameter of our system that is currently set to half the total number of instances in the true set in a given iteration of Figure 1. The time complexity for the optional learning process is depicted in equation 4, where P_i is the number of candidate elements in position i (position i is the gap between element i and element i+1). Since $0 \le |P_i| \le |S|$, $0 \le C_{optional} \le |C|(N-1)$. Therefore, $O(0) \le C_{optional} \le O(N)$

$$C_{optional} = \sum_{i=1}^{N-1} P_i$$

Equation 4: Optional Element Learning Complexity

Optional elements can improve neither precision nor recall. In other words, this phase of the learning process cannot improve either training or testing F-measure scores based on segment recognition. However, we have discovered that this phase can improve performance when extracting features from police incident reports. For example, suppose we are interested in extracting the *height* feature. Furthermore, suppose there are only two elements, the part of speech tag "CD" and the word "tall", in R_{and}. This means that any number can be followed by any word or part of speech tag except "CD", followed by the word "tall". Meanwhile, suppose the string for a person's *height* is "five feet six inches tall". Unfortunately, this RRE cannot exactly match a *height* – instead it will match "six inches tall". However, if there is an optional element "feet" between "CD" and "tall", the RRE will achieve both optimal segment accuracy and effectively extract the feature of interest, "five feet six inches tall". After this process, the example RRE becomes "in{IN} (his|her){GN} (early)? {CDS}".

4.4.5. "NOT" Learning Process

For each element generated in the processes described in sections 4.4.2, 4.4,3, and 4.4.4, the system evaluates the insertion of a "NOT" operator on the element immediately following. The last element is an exception – the "NOT" operator is not applied to it because it marks the end of the RRE. For instance, if three elements found for the feature *height* are "CD", "feet", and "tall", then a new RRE that includes ~"CD" and ~"feet" is generated (where ~ is the "NOT" operator). The new RRE replaces the RRE discovered earlier based on the F-measure score. The time complexity of applying the "NOT" operator is depicted in Equation 5. Here, $C_{NOT} = O(N')$, where N' is the total number of elements in the RRE.

$$C_{NOT} = N'$$

Equation 5: "NOT" Learning Complexity

In our implementation we used Perl. Perl, however, does not support a "NOT" operator for multi-character tokens, so we implemented the "NOT" operator as follows: we use the "NOT" operator on each single character in an element, and then use the "OR" operator to combine them. For example, ~"feet" is expressed as "([^f]|f[^e]|fe[^e]|fee[^t]|feet[^\s])".

The "NOT" operation enabled the extraction of features of interest. For example, consider a RRE that includes three elements "CD", "feet", and "tall". This RRE accepts the string "weighing/NN 180/CD pounds/NNS and/CC five/CD feet/NNS six/CD inches/NNS tall/JJ", where "NN","CD", "NNS", "CC", and "JJ" are part of speech tags. If we use "CD(\s)*feet(\s)*tall" as the RRE, it will find nothing. If we use "CD.*feet.*tall" as the RRE, it will find "180/CD pounds/NNS and/CC five/CD feet/NNS six/CD inches/NNS tall/JJ". If we use "CD([^C]|C[^D]|CD[^\s\/])*feet([^f]|f[^e]|fe[^e]|fee[^t]|feet[^\s\/])*tall([^t]|t[^a]|ta[^l]|tal[^l]|tall[^\s\/])*", the string accepted is "five/CD feet/NNS six/CD inches/NNS tall/JJ". Obviously, the last result is the one we want for the feature *height*. For the *Age* example in this section, the RRE after the "NOT" learning process is:

"in(~in) {IN}(~{IN}) (his(~his)|her(~her)) {GN}(~{GN}) (early(~early))? {CDS}(~{CDS})"

4.4.6. Handling the start and the end of the RRE

If the first element of a RRE is a part of speech tag, then our algorithm ensures that the RRE also covers the word before the tag by including "(\S)*" before the tag. For example, if the single element "CD" is discovered, then the RRE becomes ""(\S)*{CD}". This ensures that the RRE will accept strings such as "20{CD}".

The start symbol "^" and end symbol "$" of a segment also proved to be useful in some cases. As a result, our algorithm tests whether the current RRE should include "^" or "$". We simply insert "^" at the beginning of the RRE to form a new one. If the resulting RRE has equal or better performance compared to the previous RRE, then the RRE starting with "^"replaces the previous RRE. We deal with "$" in a similar manner.

4.5. Post Processing

After each loop in Figure 1, (sections 4.4.1 to 4.4.6), one RRE is generated. This RRE is considered a sub-pattern of the current feature. After all RREs have been discovered for the current feature (i.e., all segments labeled by the feature are covered), the system uses the "OR" operator to combine the RREs. In other words, given that R_1, R_2, …, R_m are m RREs that are discovered during learning, then the final RRE will be "$(R_1)|(R_2)|\ldots|(R_m)$".

In this section we have described a greedy covering algorithm that discovers a RRE for a specific feature in narrative text. The basic idea is to find high frequency patterns in segments/sentences associated with the feature. We have applied "AND", "OR", and "NOT" operators to find elements of a RRE that accept sub-patterns of the feature under consideration. Optional elements as well as the start and the end of a sentence are also components of the RRE. Finally, RREs for all sub-patterns are combined to form a single RRE with the "OR" operator.

The "NOT" operator, optional element, and the start/end of a segment are designed especially for exact match of feature extraction from police incident reports. In PSI discovery, the algorithm only requires steps 1, 2, and 3. Any segment (sentence) that matches the RRE generated is considered a PSI. Otherwise, it is not a PSI.

5. Experimental Results

In this section, we first briefly summarize the result of feature extraction from police incident reports. Following this, we discuss how we built the training and testing datasets used in PSI extraction. Finally, we present the results of the use the widely applied technique of cross-validation to evaluate our models for PSI extraction from full text patents.

Table 3 summarizes the results of 10-fold cross validation based on 100 police incident reports consisting of 1404 segments. There are ten different features evaluated in Table 3 (first column). *Eye Color*, *Gender* and *Weekday* have perfect test performance (100%) in part because we have modified the lexicon as noted in section 4.3. The performance of *Age*, *Date*, *Time*, *Height*, *Race*, and *Weight* are also excellent (F-measure scores =90%). The performance of *Hair Color* is however, not as good. This is due to the lack of *Hair Color* segments in some folds of the training sets. However, the test performances on other folds, in which there are *Hair Color* segments, are perfect (100%). As a result, we conclude that the RREs discovered for these ten features are high-quality.

Feature	Average Precision %	Average Recall %	Average F-measure %	Average # of true positives
Age	97.27	92.38	94.34	13
Date	100	94.69	97.13	8.8
Time	100	96.9	98.32	8.9
Eye Color	100	100	100	1
Gender	100	100	100	33.6
Hair Color	60	60	60	0.8
Height	100	98	98.89	2.4
Race	95	96.67	94.67	3.3
Week day	100	100	100	9.8
Weight	90	90	90	1.9

Table 3: 10-fold cross-validation performance on police incident report data

For PSI extraction, our datasets include 55 patents, of which 15 containing 7723 segments (sentences) were used in cross-validation. These patents were retrieved in the focused domain of text mining to enable us to label the training data more easily.

Each segment in each patent was manually tagged by a human expert, thereby creating our ground truth. We split all true segments randomly into 10 folds for cross-validation. Each fold was given a roughly equal number of true segments (i.e., the folds were stratified). We did the same thing with the false segments. Table 4 depicts the makeup of folds used in cross validation.

Fold	# of true segments	# of false segments	Total # of segments
1	10	761	771
2	10	762	772
3	10	762	772
4	10	762	772
5	10	762	772
6	10	762	772
7	11	762	773
8	11	762	773
9	11	762	773
10	11	762	773

Table 4: 10 folds for cross-validation

As part of our experimental method we employed resampling to vary the ratio of true segments to false segments in the training sets. Figure 3 depicts how precision, recall, and F-measure score vary with the ratio of true to false segments.

From Figure 3 it can be seen that F-measure reaches an optimum when the ratio of true to false segments is 1:73, which in fact happens to be the original ratio in the source patents. Naturally, we do not resample the test folds – otherwise, the 10-fold cross-validation performance would not reflect the true testing performance on data representative of the domain.

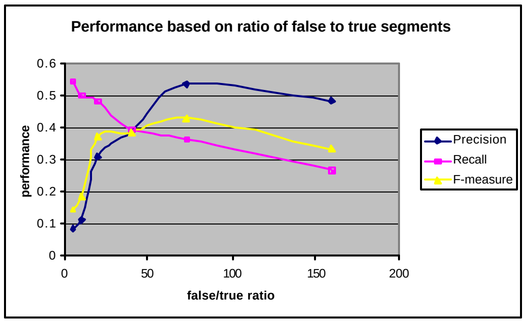

Figure 3: Performance curve of false/true segment ratio

Table 5 depicts optimal parameter values for PSI extraction. In the first column, we list several important parameters to our algorithm. In each case, we varied each parameter in turn while holding the others fixed in order to determine an optimal value. Figures 4 and 5 depict the results of varying b_{Others} as an example of this approach to optimization of parameter values. The plots in Figures 4 and 5 are drawn from the average testing results based on 10-fold cross validation.

It is interesting to note that in Figure 4, the higher the value of beta, the higher the average recall and the lower the precision. Meanwhile, the number of true positives increases as beta increases (Figure 5). One important property of precision and recall is that they are inversely related [11]. One generally has to trade off precision to increase recall and vise versa. Our intuition in PSI extraction is that it is more important to extract at least one PSI per patent than to extract several PSIs that are mixed with segments that are not PSIs. In this sense, precision is more important than recall in PSI extraction. Although this is intuitive, in this particular case we did not select 0.25 for b_{Others} in Table 5 due to the fact that there were too few true positives extracted. Moreover, the F-measure metric has the best performance when b_{Others} equals 0.5, a value that still reflects a focus on precision but has better recall. Therefore, we chose 0.5 as the optimal value for b_{Others} in Table 5.

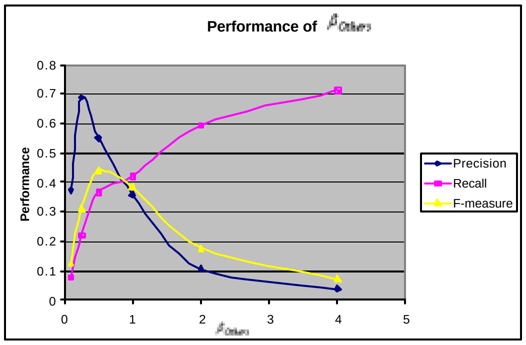

Figure 4: Performance of b_{Others}

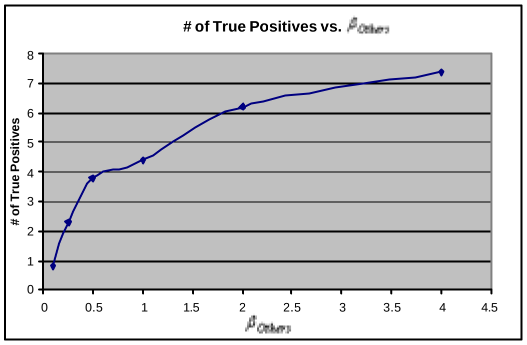

Figure 5: True positives based on b_{Others}

We tuned all the other parameters in the same way. It is possible for the parameters e_{Word} and e_{Tag} to be over 100% because the same word or part of speech tag may occur more than once in a single sentence. The final performance after all parameters have been optimized is not guaranteed to be globally optimal since our method to tune parameters is a greedy approach. The parameter values used for final cross-validation tests reported in Table 6 are depicted in Table 5.

Parameter Name	Value
d	2
b_{Root}	6
b_{Others}	0.5
e_{Word}	5%
e_{Tag}	400%
l	5

Table 5: Optimal Parameter Values

We performed 10-fold cross validation based on the configuration listed in Table 5 with a 1:73 ratio of true to false segments in the training set. The test results of the cross-validation are shown in Table 6.

Test sets	Precision	Recall	F-measure	# of true positives
1	85.71%	60.00%	70.59%	6
2	57.14%	40.00%	47.06%	4
3	62.50%	50.00%	55.56%	5
4	66.67%	40.00%	50.00%	4
5	37.50%	30.00%	33.33%	3
6	42.86%	30.00%	35.29%	3
7	40.00%	18.18%	25.00%	2
8	50.00%	27.27%	35.29%	3
9	71.43%	45.45%	55.56%	5
10	44.44%	36.36%	40.00%	4
Ave rage	55.83%	37.73%	44.77%	3.9

Table 6: 10-fold cross-validation test performance on patent data

The average precision is 55.83%. That means over half of the sentences extracted using the RREs discovered by our algorithm contained information relevant to the problem solved by patents. Considering the complexity of natural language expressions used in patents, we consider this result promising. The average recall is 37.73%. This value is acceptable because as noted previously, precision is more important than recall in this particular application. The average F-measure with ß=1 is 44.77%. This value tells us that we are currently successfully extracting PSI-related segments about half the time. Another important metric is the distribution of correctly extracted PSIs across patents. In order to assess our algorithm's performance with regard to this metric, we measured the distribution of true positives from the 10 test folds across the 15 patents used to form the training set. Ideally, we would like to extract one or more PSIs from each patent. In this case, 80% of the original 15 patents were covered by at least one PSI. This result too is quite promising.

Our experimental results provide evidence that our semi-supervised approach to RRE discovery can be usefully applied to extract features from police incident reports and PSIs from patents. With the former application we achieved very good test set performance, and with the latter we achieved reasonable and stable test set performance. Our work in the patent domain is ongoing.

6. Conclusion

We have presented a semi-supervised algorithm for feature extraction from police incident reports and patents. The algorithm can be used to learn reduced regular expressions that are used as patterns to match and extract previously unseen features with a high degree of reliability. Our experiments demonstrate that reduced regular expressions extract information useful in criminal justice, homeland defense, and patent intelligence applications.

7. Acknowledgements

The authors also gratefully acknowledge the help of co-workers, family members and friends, and co-authors William M. Pottenger and Tianhao Wu gratefully acknowledge the continuing help of their Lord and Savior, Yeshua the Messiah (Jesus Christ) in their lives.

References

[1] S. Soderland. Learning "Information Extraction Rules for Semi-structured and Free Text". Machine Learning, 34(1-3):233-272, (1999).

[2] Tianhao Wu and William M. Pottenger. "A Semi-supervised Algorithm for Pattern Discovery in Information Extraction from Textual Data". The seventh Pacific-Asia conference on Knowledge Discovery and Data Mining (PAKDD), April (2003).

[3] M. Hehenberger, P. Coupet, M. Stensmo and C. Huot. "Text Mining of Chemical and Patent Literature". Division of Chemical Information, 217th ACS National Meeting Anaheim, California, March 21-25, (1999).

[4] Bronwyn H. Hall, Adam Jaffe, Manuel Trajtenberg. "Market Value and Patent Citations: A First Look". NBER Working Paper No. W7741 Issued in June (2000).

[5] Anthony Breitzman, Patrick Thomas and Margaret Cheney. "Technological Powerhouse or Diluted Competence: Techniques for Assessing Mergers via Patent Analysis" Society of Competitive Intelligence Technical Intelligence Symposium, San Francisco, June 8-9, (2000).

[6] Hopcroft, J. and J. Ullman. "Introduction to Automata Theory". Languages and Computation, Addison-Wesley, (1979).

[7] Eric Brill. "Pattern-Based Disambiguation for Natural Language Processing". Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, (2000).

[8] Christopher D. Manning and Hinrich Schütze. "Foundations of Statistical Natural Language Processing". MIT Press, (2000).

[9] Jeffrey C. Reynar and Adwait Ratnaparkhi. "A Maximum Entropy Approach to Identifying Sentence Boundaries". In Proceedings of the Fifth Conference on Applied Natural Language Processing. Washington, D.C. March 31-April 3, (1997).

[10] Van Rijsbergen. "Information Retrieval". Butterworths, London, (1979).

[11] Cleverdon, C.W. "On the inverse relationship of recall and precision". Journal of Documentation, 28, 195-201 (1972).

[12] Brill, Eric. "Transformation-Based Error Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging". Computational Linguistics 21(94): 543-566. 1995.

[13] Michael Chau, Jennifer J. Xu, Hsinchun Chen, "Extracting Meaningful Entities from Police Narrative Reports". in Proceedings of the National Conference for Digital Government Research. Los Angeles, California, May 19-22, (2002).