A Semi-Supervised Active Learning Algorithm for Information Extraction from Textual Data

Tianhao Wu and William M. Pottenger
Computer Science and Engineering at Lehigh University, 19 Memorial Drive West Bethlehem, PA 18015. E-mail: {tiw2, billp}@lehigh.edu

In this article we present a semi-supervised active learning algorithm for pattern discovery in information extraction from textual data. The patterns are reduced regular expressions composed of various characteristics of features useful in information extraction. Our major contribution is a semi-supervised learning algorithm that extracts information from a set of examples labeled as relevant or irrelevant to a given attribute. The approach is semi-supervised because it does not require precise labeling of the exact location of features in the training data. This significantly reduces the effort needed to develop a training set. An active learning algorithm is used to assist the semi-supervised learning algorithm in order to further reduce training set development effort. The active learning algorithm is seeded with a single positive example of a given attribute. The context of the seed is used to automatically identify candidates for additional positive examples of the given attribute. Candidate examples are manually pruned during the active learning phase, and our semi-supervised learning algorithm automatically discovers reduced regular expressions for each attribute. We have successfully applied this learning technique in the extraction of textual features from police incident reports, university crime reports, and patents. The performance of our algorithm compares favorably with competitive extraction systems being used in criminal justice information systems.

1. Introduction

Homeland Defense is an important application domain for information extraction and mining technologies. Law enforcement agencies around the world generate numerous reports, many of them in narrative (unstructured) textual form. Information extraction techniques can be employed to automatically identify and extract data from such unstructured text and store it in fielded, relational form in databases. Once stored in relational form, the extracted information is useful in a variety of everyday law enforcement applications such as search and retrieval. The extracted information can also be used to map modus operandi to physical descriptions of criminal suspects, an application of advanced modeling useful in suspect identification.

Regular expressions can be used as patterns to extract features from semi-structured and narrative text (Soderland, 1999). After studying hundreds of police incident reports, patents and other unstructured data, we found 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 "<u>{CD} feet {CD} inches tall</u>", where {CD} is the part-of-speech tag for a numeric value. 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 Lockheed Martin M&DS, the Pennsylvania State Police and the City of Bethlehem Police Department. Our target is to develop a system that extracts features related to criminal modus operandi and physical descriptions of suspects as recorded in narrative incident reports. Our results in (Wu & Pottenger, 2003a) demonstrate that our semi-supervised learning algorithm achieves excellent performance on ten features important in homeland defense. In this article we present the results of extending the algorithm presented in (Wu & Pottenger, 2003a) by combining it with an active learning algorithm.

Our combined semi-supervised active learning algorithm requires significantly less effort to develop a training set than other approaches. A training-set developer does not need to label an inordinate amount of data. Rather, the algorithm automatically generates a small set of candidate segments for the developer to label. A second benefit of the algorithm is that the training-set developer need only record whether a specific feature of interest occurs somewhere in a segment as opposed to labeling the exact location of the feature in the segment. For instance, given that the feature of interest is *Height* and given a segment "The suspect is five feet eight inches tall weighing 180 pounds", the training-set developer need only assign this segment the label *Height*. Our semi-supervised active learning algorithm discovers a reduced regular expression that precisely matches and extracts the feature “five feet eight inches tall” from this segment.

This article is organized as follows. In section 2, we summarize previous work in which we conducted a manual study of regular expressions for use in extracting features from police incident reports. Following this, we provide definitions in section 3. In section 4, we present a semi-supervised reduced regular expression discovery algorithm and analyze biases of the algorithm. Following this in section 5 we describe our active learning approach. In section 6, we discuss experimental results for the combined semi-supervised active learning algorithm. We introduce related work in section 7 and discuss future work in section 8. Finally, we give conclusions in section 9 and acknowledge those who have contributed to this work.

2. Previous Work

In our work with Lockheed Martin for the Pennsylvania State Police, our first purpose was to ascertain whether regular expressions are suitable for information extraction from unstructured (narrative) police reports.

To address this question we studied voluminous police incident reports and manually developed regular expressions for extracting key attributes useful in criminal justice information systems. Our source data was drawn from Fairfax County, Virginia police incident reports. Based on this training data, we manually generated regular expressions for the seven attributes listed in Table 1. An independent test dataset was used to assess the performance of the regular expressions.

As can be seen in Table 1, the test performance was promising. We achieved 100% precision for *Time*, *Age*, *Height*, *Hair Color*, *Eye Color*, and *Weight*. *Race* also has high precision (94%). In addition, recall was above 90% for all of the features. In fact, we achieved 100% for both precision and recall for *Time*, *Height*, *Eye Color*, and *Weight*.

TABLE 1. Test Performance of Manual Expressions

Attribute	Precision	Recall	F_β (β=1)
Time	100%	100%	100%
Race	94%	97.9%	95.91%
Age	100%	94.8%	97.33%
Height	100%	100%	100%
Hair Color	100%	90.6%	95.07%
Eye Color	100%	100%	100%
Weight	100%	100%	100%

Based on these results we concluded that regular expressions are suitable for the extraction of attributes in police incident reports. In the course of this work, however, we confirmed that it is both time consuming and tedious to create regular expressions for information extraction manually. Thus, we embarked on a research effort to develop a data-driven algorithm to automatically discover regular expressions based on a small training set. We describe the resulting semi-supervised active learning algorithm in sections 4 and 5. In section 3, we define terminology used in this article.

3. Definitions

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

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

1) $\forall\alpha\in\Sigma$, α is a regular expression and denotes the set {α}.
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.” (Hopcroft & Ullman, 1979)

Reduced regular expression (RRE): Our reduced regular expression is defined as follows: Given a finite alphabet Σ, our reduced regular expression is defined as a set:

- $\forall\alpha\in\Sigma$, α is a RRE and denotes the set {α}.
- All words in our lexicon and all part-of-speech tags in the Penn tag set (Manning & Schütze, 2000) belong to Σ.
- $\wedge\in\Sigma$, $\$\in\Sigma$, where ^ is the start of a line, and $ is the end of a line.
- [0-9]$\in\sum$, where [0-9] represents any single numeric digit.
- [A-Z]$\in\sum$, where [A-Z] represents any single alphabetic character (upper or lower case).
- All punctuation characters belong to $\sum$.
- All white space characters belong to $\sum$.
- $\varepsilon\in\sum$ is the null symbol (i.e., the empty string).
- (a|ε) is a RRE, where $a\in\sum$ is any single alphanumeric or punctuation character.
- if r and s are RREs denoting the languages R and S, respectively, then (rs) is an RRE that denotes the set RS.

The major difference between regular expressions and reduced regular expressions is that reduced regular expressions do not support Kleene closure (i.e., ‘*’). Examples of regular expressions that are not RREs include: α*, (α|β), and α+, where α and β are arbitrary word or part-of-speech members of Σ. We have not found it necessary to support such regular expressions to achieve high accuracies.

Feature: The smallest unit of information extracted. Examples include values for the attributes *Height, Weight, Age,* etc.

Segment: A (portion of) a sentence. Commas mark segment boundaries. An 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: 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 an attribute *a*, then the true set consists of all segments labeled *a* in the training set.

False set If the system is learning a RRE for an attribute *a*, then the false set consists of all segments that are not labeled *a* in the training set.

Element: A word in the RRE with frequency in the true set higher than a threshold λ_{word} or a part-of-speech tag in the RRE with frequency in the true set higher than a threshold λ_{tag}.

Root: The first element discovered by our algorithm in a RRE.

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

N: An input parameter to our algorithm that fixes the maximum number of elements in a given RRE. N is an integer greater than or equal to one.

S: S is the set of word tokens in the lexicon employed in our approach combined with the part-of-speech tag tokens in the Penn tag set (Manning & Schütze, 2000). |S| is the total number of tokens in S.

4. A Semi-supervised learning Approach

In this section we present our semi-supervised approach to the discovery of RREs from a small set of labeled training segments. The process begins with pre-processing. Following this we apply a greedy algorithm to discover RREs. We detail these steps in what follows.

4.1. Pre-Processing

Our semi-supervised learning algorithm requires three pre-processing steps: segmentation, segment labeling, and part-of-speech tagging.

4.1.1. Segmentation

Each incident report is split into segments at this stage. Each segment becomes an instance in our system. The first step splits the input into sentences using the technique presented in (Reynar & Ratnaparkhi, 1997) to detect sentence boundaries. The police incident report input is further split on commas. As mentioned in section 3, there are a small number of cases where commas are not segment boundaries. We also assume that no features cross segment boundaries. This assumption is practical for a number of important attributes, including those discussed in section 6.

4.1.2. Segment Labeling

Prior to the start of the labeling stage, domain experts must identify the attributes that will be extracted. For instance, if the high-level goal is to extract physical descriptions of suspects, the list should include *Height*, *Weight*, *Eye Color*, etc. During training set development, each segment is evaluated manually and assigned one or more labels. For example, if a segment includes *Height* and *Weight* information, the domain expert assigns both of these labels to the segment. After labeling, each attribute has its own true set and false set.

4.1.3. 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 (1995) part-of-speech tagger to tag our training sets. Brill's tagger uses the Penn tag set (Manning & Schütze, 2000) (Table 2 contains some examples of Penn tags). We have enhanced our lexicon to include extra tags for feature extraction from police incident reports. For example, {CDS} is used for plural numbers such as "twenties".

TABLE 2. Example tags from Penn tag set

Tag	Category	Example
CD	Cardinal number	3, fifteen
IN	Preposition	in, for
PRP	Pronoun	they, he
PRP$	Determiner, possessive	their, your
DT	Determiner, article	the, both
CC	Conjunction	and, or
RB	Adverb	ago, very
JJ	Adjective	happy, bad
NN	Noun, singular	aircraft, data
NNP	Proper Noun	London, Reston
NNS	Plural Noun	books, years

VBG	Verb, present participle	taking, living
VBP	Verb, base present	are, take
VBD	Verb, auxiliary *be*, past	was, were
VBZ	Verb, auxiliary *be*, present	is

4.2. 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 the true set, while having low frequency outside the true set. The algorithm first discovers the most common element of a RRE, termed the root of the RRE. The algorithm then extends the RRE using an "AND" operator, after which it discovers the gaps between elements of the RRE. Finally, the start and the end of the RRE are learned. Figure 1 depicts the entire learning process. Figure 2 portrays the same process in algorithmic form. An example (in section 4.3) explains how the algorithm works at each stage of the learning process.

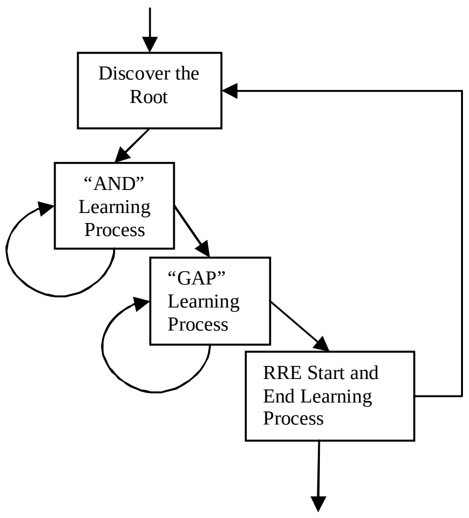


FIG. 1. RRE Discovery

```
Do {
    Find the ROOT
    "AND" learning
    "GAP" learning
    "RRE Start/End" learning
    Prune true set
} While True Positives ≥ δ
```

FIG. 2. RRE Discovery in Algorithmic Form

Our approach employs a covering algorithm. After a 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 δ. We use this threshold because overfitting results if too few segments are used to discover a RRE. δ is a parameter in our system, and must be an integer. By default δ is set to two

Our approach is a semi-supervised learning method. 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. Nonetheless, the RRE learned by the algorithm precisely matches the feature of interest – no more and no less. We depict the details of each step of the algorithm in what follows.

4.2.1. Discovering the Root of a RRE

In this step, each word and/or part-of-speech tag (specified as a simple RRE) in each true set segment is matched against all segments. The performance of each such RRE in terms of F_β from Van Rijsbergen (1979) (Equation 1 below) 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 β is the ratio of recall to precision and enables one to place greater or lesser emphasis on recall versus precision depending on the needs of the application.

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \qquad (1)$$

The element with maximum F_β 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 containing the desired feature. It must also have low frequency in segments that do not contain the desired feature. For example, the root discovered for the attribute *Age* in the example given in section 4.3 is the part-of-speech tag "{IN}". Because each element in each and every segment in the true set must be tested during root discovery, the time complexity of root discovery is equal to the number of elements in the true set.

Our approach places less emphasis on precision and more on recall during the root discovery process. We use the parameter β_{root} (by default set to six) 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", "GAP", and "Start/End" learning phases all prune false positives from the set of segments that match the root RRE. As will become evident, the result is both high precision and high recall.

4.2.2. "AND" Learning Process

After the root is discovered, the algorithm tests 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. The RRE with the highest F_β replaces the previous RRE. For example, starting with the root "{IN}" for the attribute *Age*, the RRE "years <GAP> {IN}" may be discovered after the first pass of the "AND" learning process, where <GAP> is the gap discovered between these two elements[1]. Intuitively, element adjacency implies the use of an "AND" operator: thus the name 'AND learning process'.

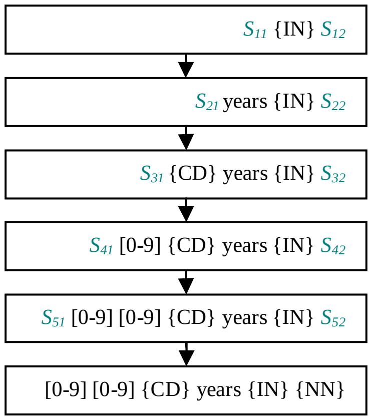
The new RRE is then extended in the same way. As before, candidate elements are inserted before and after the current RRE. The algorithm measures the performance of each extended RRE, and the RRE with the maximum F_β is selected. In this sense our algorithm is greedy. Continuing the previous example, the RRE after the second pass of the "AND" learning process may be "{CD} <GAP> years <GAP> {IN}", where "{CD}", "years", and "{IN}" are elements of the RRE and <GAP> represents the gaps that have been learned.

Candidate elements consist not only of words and part-of-speech tags, but also can be 'numeric' tokens composed of the digits [0-9]. The lengths of such tokens provide clues useful in RRE discovery. For example, two digit tokens such as "23" are often a person's age, whereas four digit tokens such as "2003" are likely to be a year. Similarly, a person's height always has one digit for feet and one or two digits for inches, and a person's weight usually has three digits.

The overall complexity of the "AND" learning process depends on both the number of candidate elements and the maximum number of elements in R_{and}. 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 2. Although S_{ij} can contain many elements, there are only two positions to test in each pass of the "AND" learning process. One is the position before the current RRE and the other is the position after the current RRE. An example of how S_{ij} changes during the "AND" learning process is depicted in Figure 3. The actual size of S_{ij} depends on the number of candidate elements for each position j=1 and j=2 in a segment. The algorithm is data-driven in this regard.

Generalizing from this example, we can see that in the "AND" learning process there are at most N-1 passes required to test the N-1 positions between a maximum of N elements. Thus the "AND" learning process ends when at least one of the two following conditions is met: either the number of elements in the RRE reaches the maximum N, or all candidate elements in the segment have been tested in all allowable positions.

[1] The details of gap discovery will be discussed in section 4.2.3.



S_{11} {IN} S_{12}
S_{21} years {IN} S_{22}
S_{31} {CD} years {IN} S_{32}
S_{41} [0-9] {CD} years {IN} S_{42}
S_{51} [0-9] [0-9] {CD} years {IN} S_{52}
[0-9] [0-9] {CD} years {IN} {NN}

FIG. 3. "AND" learning example for the *Age* attribute

Given that $0 \le |S_{ij}| \le |S|$, we have $0 \le C_{AND} \le 2(N-1)|S|$. Therefore, the complexity C_{AND} is bounded by $O(0) \le C_{AND} \le O(N|S|)$.

$$0 \le C_{AND} \le \sum_{i=1}^{N-1} \sum_{j=1}^{2} |S_{ij}| \qquad (2)$$

The fact that the algorithm is data-driven improves its performance – this can be deduced by comparing the actual complexity in Equation 2 with the upper bound 2(N-1)|S|. In other words, due to the fact that a given segment does not contain many words, in practice $|S_{ij}| << |S|$. In fact we have found a value of 10 to work well for the maximum number of elements N allowed in an RRE.

4.2.3. "GAP" Learning Process

In the "GAP" learning process our algorithm learns the length of the gap between elements of an RRE. For example, suppose a RRE learned for the *Date* attribute is "{CD} {MONTH}", where {MONTH} is the part-of-speech tag for January through December. Because our algorithm is designed to learn gaps between adjacent elements automatically, it can for example learn that a gap of zero to two elements between {CD} and the following {MONTH} is optimal. An example of this type of RRE is "{CD} {token}{0,3} {MONTH}", where {token} is a word or part-of-speech tag in Σ. This RRE allows at most three elements between a number and a following month, e.g., "the{DT} 5th{CD} of{IN} January{MONTH}". In this case, the word "of" followed by its part-of-speech tag followed by the word "January" form a gap that is three elements in size.

In this simple *Date* attribute example we assumed that gaps are measured in terms of elements – in fact, the execution time performance of our algorithm is significantly improved if gaps are measured in terms of characters. As a result, we measure gaps in terms of characters and use an input parameter ψ to control the maximum gap allowed between any two adjacent elements in a RRE. Our system initializes $\psi = 100$ based on the observation that no adjacent elements are separated by more than 100 characters in our training data. For each gap, our algorithm first determines $\phi = \min(\psi, \omega)$, where ω is the longest gap in any segment in the true set between the two elements under consideration. For example, in the segment "the{DT} 5th{CD} of{IN} January{MONTH}", including spaces there are 15 characters between '{CD}' and '{MONTH}', and $\phi = \min(100,15)$, which is 15.

Once ϕ has been determined, the algorithm tests ".{0, ϕ}" as the gap. In this expression, the '.' represents any single character member of Σ. The syntax ".{0, ϕ}" means that any single character member of Σ can occur between 0 and ϕ times. The algorithm in turn tests gaps of ".{0, ϕ-1}", ".{0, ϕ-2}", …, "{0,0}". The smallest gap that does not decrease the current RRE's performance is used as the final gap between two elements of the RRE. Gaps between different elements can differ in size.

The time complexity for the "GAP" learning process is depicted in Equation 3, where G_i is the number of comparisons to identify the optimal gap between RRE element i and element i+1. $0 \le G_i \le \psi$. Given N, the maximum number of elements in a RRE, the complexity of gap discovery C_{GAP} is thus $0 \le C_{GAP} \le (N-1)\psi$. Therefore, $O(0) \le C_{GAP} \le O(N\psi)$.

$$C_{GAP} = \sum_{i=1}^{N-1} G_i \qquad (3)$$

4.2.4. Start and End Learning

The start symbol "^" and end symbol "$" of a segment are also useful in RRE discovery. As a result, our algorithm tests whether the current RRE should include the start symbol and/or the end symbol. We first insert the start symbol at the beginning of the RRE to form a new RRE. If it has equal or better performance compared to the previous RRE, then the new RRE with the start symbol replaces the previous one. We deal with the end symbol in a similar manner. In addition, if the initial element of a RRE is a part-of-speech tag, our algorithm ensures that the RRE also covers the word before the tag.

The time complexity of this learning process is O(1) since only two candidate RREs are tested.

4.3. Example

In this section we use the *Age* attribute to illustrate our semi-supervised learning algorithm in detail. Tables 3 and 4 contain the initial true and false sets respectively. The characters between "{" and "}" are part-of-speech tags. {MALE} is a special tag for words of male gender, and {FEMALE} is the tag for words of female gender.

There are two distinct *Age* patterns in the true set. One is illustrated by the feature "in her twenties", the other by "25 years of age". Our semi-supervised algorithm discovers two distinct RREs, one based on each *Age* pattern. In other words, in this example our covering algorithm completes two iterations of Figure 2 during RRE discovery.

TABLE 3. True set for example

Segment Number	True segments
1	six{CD} feet{NNS} tall{JJ} and{CC} 25{CD} years{NNS} of{IN} age{NN}
2	both{DT} 30{CD} years{NNS} of{IN} age{NN} with{IN} cornrows{NNS}
3	in{IN} his{MALE} twenties{CDS} standing{VBG} six{CD} feet{NNS} tall{JJ}
4	they{PRP} are{VBP} in{IN} their{PRP$} thirties{CDS}
5	she{FEMALE} was{VBD} in{IN} her{FEMALE} twenties{CDS}
6	Tom{NNP} is{VBZ} in{IN} his{MALE} early{JJ} teens{CDS}

TABLE 4. False set for example

Segment Number	False segments
1	the{DT} first{JJ} man{MALE} is{VBZ} in{IN} his{MALE} car{NN}
2	in{IN} the{DT} roaring{VBG} twenties{CDS}
3	in{IN} the{DT} 111{CD} block{NN}
4	25{CD} years{NNS} ago{RB}
5	5{CD} feet{NNS} 8{CD} inches{NNS} tall{JJ} with{IN} a{DT} tall{JJ} thin{JJ} build{NN}
6	weighing{VBG} 180{CD} pounds{NNS}

In the first iteration of the algorithm, the root "{IN}" of the first RRE is discovered with $F_{\beta=6.0}$ of 0.98. Next, the "AND" learning process extends this root to the five RREs portrayed in Figure 4 in sequence. In order to prune the false positives covered by the root, β is set to 0.5 in this example (by default it is set to two in our system). As noted previously, greater emphasis is placed on precision during "AND" learning in this way. Each of these five RREs has $F_{\beta=0.5}$ of 0.71. Thus, in this example R_{and} is learned after five passes in the "AND" learning process.

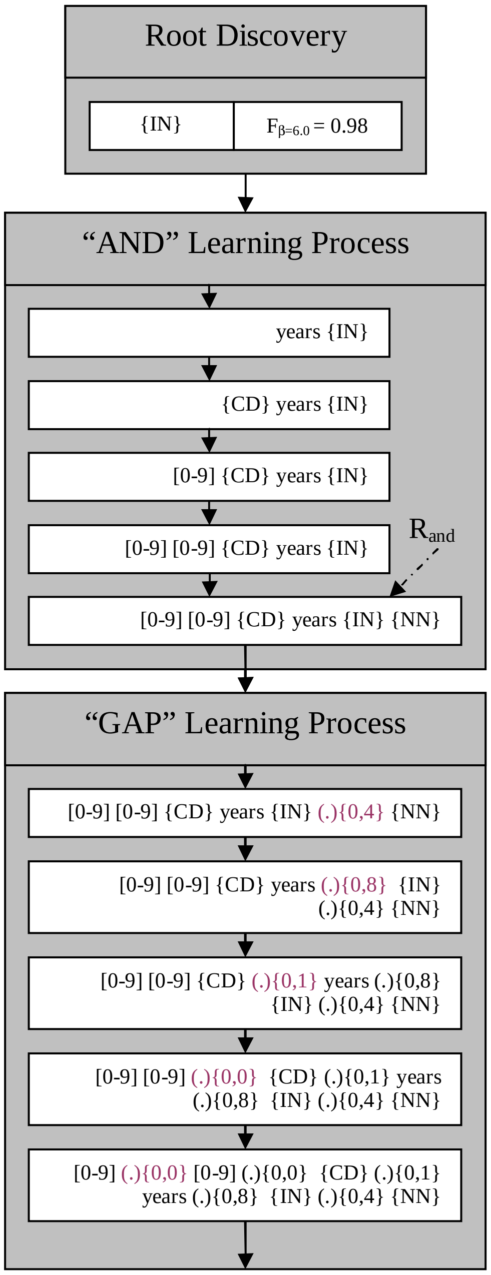

Root Discovery
{IN}
F_β=6.0 = 0.98
"AND" Learning Process
years {IN}
{CD} years {IN}
[0-9] {CD} years {IN}
[0-9] [0-9] {CD} years {IN}
R_and
[0-9] [0-9] {CD} years {IN} {NN}
"GAP" Learning Process
[0-9] [0-9] {CD} years {IN} (.){0,4} {NN}
[0-9] [0-9] {CD} years (.){0,8} {IN} (.){0,4} {NN}
[0-9] [0-9] {CD} (.){0,1} years (.){0,8} {IN} (.){0,4} {NN}
[0-9] [0-9] (.){0,0} {CD} (.){0,1} years (.){0,8} {IN} (.){0,4} {NN}
[0-9] (.){0,0} [0-9] (.){0,0} {CD} (.){0,1} years (.){0,8} {IN} (.){0,4} {NN}

FIG. 4. The first iteration of RRE discovery

After R_{and} is discovered, the "GAP" learning process takes place and tailors R_{and} to further improve precision. The process is depicted for our example in the "GAP" learning process in Figure 4. This process involves five steps, one for each of the five gaps between the six elements in R_{and}. Each intermediate RRE produced during "GAP" learning in this example has $F_{\beta=0.5}$ of 0.71.

In this case inclusion of either the start or the end symbol in the RRE decreases $F_{\beta=0.5}$, so the RRE discovered after "GAP" learning remains unchanged. This RRE is depicted at the bottom of Figure 4 and covers segments 1 and 2 in the true set. To prepare for the second iteration of the algorithm, segments 1 and 2 are thus removed from the true set. Therefore, the second iteration begins with segments 3, 4, 5, and 6 in the true set, and segments 1, 2, 3, 4, 5, and 6 in the false set. The steps to learn the second RRE are portrayed in Figure 5.

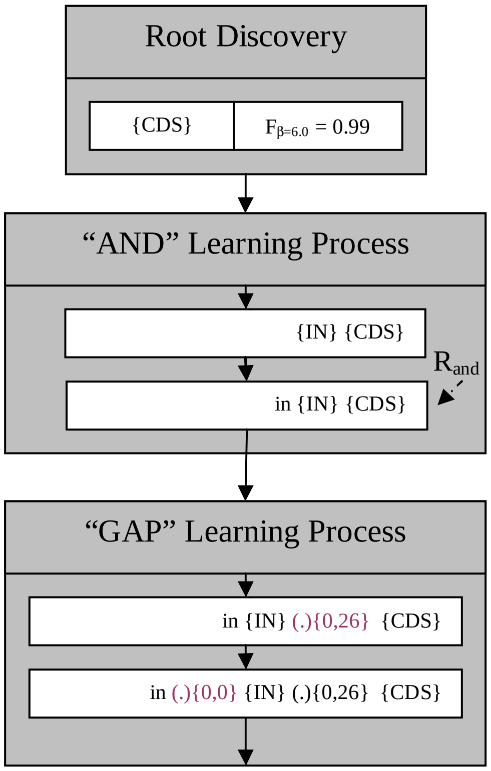


FIG. 5. The second iteration of RRE discovery

As before, the addition of either the start or the end symbol decreases $F_{\beta=0.5}$. As a result, the final RRE for the second iteration is the RRE depicted at the bottom of Figure 5. The covering algorithm terminates after the second iteration because there are no true segments left in the true set – all have been covered by the two RREs discovered.

This example highlights a distinguishing characteristic of our algorithm – features of a given attribute can be extracted precisely from a training set in which the features are imprecisely labeled. For instance, the RRE "in (.){0,0}{IN} (.){0,26} {CDS}" precisely matches only the *Age* feature "in{IN} his{MALE} twenties{CDS}" from the segment "in{IN} his{MALE} twenties{CDS} standing{VBG} six{CD} feet{NNS} tall{JJ}". Although this segment contains both *Age* and *Height* information and was labeled as such during training set development, our algorithm discovers an expression that precisely matches and extracts only *Age* features. Although we have not exemplified the discovery of a RRE for extracting *Height*, the same holds true. I.e., features for *Height* are also precisely matched and extracted despite the fact that the source segments are imprecisely labeled. To summarize, precise labeling of features is tedious and time-consuming, and this technique reduces training set development time without impacting performance.

4.4. Biases

In this section we discuss various biases inherent in the use of RREs in information extraction. This includes language biases, search biases, and overfitting biases inherent in our RRE discovery algorithm.

Our semi-supervised learning algorithm is *a top-down search algorithm*. It starts with a general description, the root of a RRE, and then extends it to a more specific RRE. This is a general-to-specific search bias. A more crucial search bias is that the algorithm attempts to identify words or part-of-speech tags that occur most often in the true set while having low frequency in the false set. Another search bias is that the algorithm attempts to discover the first and last elements of an RRE rather than every detail of a given attribute's features. When the first and last elements of an RRE are discovered, the extension process terminates.

Our semi-supervised learning algorithm employs *forward pruning*[2]. The algorithm begins with the simplest search, i.e., the search to find the root of the RRE. Subsequently only segments in the true and false sets containing the root element are used in the next step to extend the RRE. Therefore, certain complex descriptions are pruned without being considered. In this way, some overfitting problems can be avoided (Witten & Frank, 1999).

Our approach is clearly *a greedy search algorithm*. In each step to extend the RRE during the "AND" learning process, the algorithm chooses a word or part-of-speech tag such that F_β of the extended RRE is greater than or equal to F_β for the previous RRE. In other words, $F_i \le F_{i+1}$, where F_i is F_β of the RRE after the ith extension, and F_{i+1} is F_β of the RRE after the (i+1)th extension (we refer to this relationship below as Equation 4).

There are many such greedy algorithms used in information extraction that are not globally optimal (e.g., decision tree algorithms, transformation-based learning, etc.). Indeed, our approach is no different in this respect, yet greedy algorithms often achieve excellent results. The results portrayed in section 6 provide evidence that our approach indeed achieves near-optimal performance.

Another bias of our algorithm is that *the recall of a RRE monotonically decreases during extension of the RRE in the "AND" learning process*. The proof of this fact is contained in Figure 6. As a result, our approach initially places less emphasis on precision and more on recall during the root discovery process. In this way we avoid overfitting during the "AND" learning process.

[2] Forward pruning uses simplest-first search and stops when a sufficiently complex concept description is found (please refer to section 1.5 of (Witten & Frank, 1999).

There is also a precision bias in our algorithm: *during the "AND" learning process, the precision of a RRE monotonically increases*. We simplify the proof of this fact in Figure 7 by substituting the symbol 'F' for F_β.

❖ Suppose RRE_i is the RRE after the ith extension, and RRE_{i+1} is the RRE after the (i+1)th extension for a given attribute in the "AND" learning process.

❖ We first show that all segments covered by RRE_{i+1} must also be covered by RRE_i.

In the "AND" learning process, $RRE_{i+1} = E_{i+1} RRE_i$ or $RRE_{i+1} = RRE_i E_{i+1}$ where E_{i+1} is the element (a simple RRE) discovered in the (i+1)th extension. Therefore, any string accepted by RRE_{i+1} will either take the form $S_{i+1} S'_{i+1}$ or $S'_{i+1} S_{i+1}$ where E_{i+1} accepts S_{i+1}, RRE_i accepts S'_{i+1}, and S_{i+1} and S'_{i+1} are substrings of the strings accepted by RRE_{i+1}. This implies that any segment containing a substring accepted by RRE_{i+1} must also include substrings accepted by RRE_i. Since the discovery of the start and end of a RRE is a special form of the "AND" learning process, the same conclusion holds.

In the "GAP" learning process, the gap learned is the smallest one. Thus, strings accepted by the RRE with the smallest gap discovered must also be accepted by a RRE containing the same elements but having larger gaps between elements.

Based on the above, we conclude that all segments covered (i.e., containing strings accepted) by RRE_{i+1} must also be covered by RRE_i.

❖ Obviously, all true-set segments covered by RRE_{i+1} must thus be covered by RRE_i. This implies that $TP_i \ge TP_{i+1}$. Furthermore, we have $TP_i + FN_i = TP_{i+1} + FN_{i+1}$.

❖ Thus $\dfrac{TP_i}{(TP_i + FN_i)} \ge \dfrac{TP_{i+1}}{(TP_{i+1} + FN_{i+1})}$

❖ Or $R_i \ge R_{i+1}$ (5)

FIG. 6. The proof of monotonically decreasing recall

It is important that each RRE discovered have high precision. This is due to the fact that the overall precision of an expression is always greater than or equal to the lowest precision of its constituent RREs. In other words, the overall precision will be high if all constituent RREs have high precision. Thus, this bias aids in ensuring that, overall, precision is high.

Our target is to discover RREs that overall have both high precision and high recall. From the above analysis, it is clear that there is an inverse relationship between precision and recall (Cleverdon, 1972) during RRE discovery.

❖ Suppose P_i and P_{i+1} are the precisions of RRE_i and RRE_{i+1} in the ith and (i+1)th extensions respectively of the "AND" learning process.

❖ We wish to prove $P_i \le P_{i+1}$

Assuming $P_i > P_{i+1}$ we have the following:

$$\frac{F_i R_i}{(\beta^2+1)R_i - \beta^2 F_i} > \frac{F_{i+1}R_{i+1}}{(\beta^2+1)R_{i+1} - \beta^2 F_{i+1}}$$

➔

$$F_iR_i((\beta^2+1)R_{i+1} - \beta^2F_{i+1}) > F_{i+1}R_{i+1}((\beta^2+1)R_i - \beta^2F_i)$$

➔

$$F_iR_i(\beta^2+1)R_{i+1} - F_iR_i\beta^2F_{i+1} > F_{i+1}R_{i+1}(\beta^2+1)R_i - F_{i+1}R_{i+1}\beta^2F_i$$

➔

$$(\beta^2+1)R_iR_{i+1}(F_i - F_{i+1}) > \beta^2F_iF_{i+1}(R_i - R_{i+1})$$

From Equations 4 and 5, we have

$F_i \le F_{i+1}$ ➔ $0 \ge (\beta^2+1)R_iR_{i+1}(F_i - F_{i+1})$

$R_i \ge R_{i+1}$ ➔ $\beta^2F_iF_{i+1}(R_i - R_{i+1}) \ge 0$

➔

$$0 \ge (\beta^2+1)R_iR_{i+1}(F_i - F_{i+1}) > \beta^2F_iF_{i+1}(R_i - R_{i+1}) \ge 0$$

➔

$0>0$

Since $0=0$, the assumption is incorrect.

❖ Thus $P_i \le P_{i+1}$.

FIG. 7. The proof of monotonically increasing precision

Our semi-supervised learning algorithm is however a covering algorithm. The RREs discovered are used in sequence to extract features in previously unseen data. If any single constituent RRE matches a feature, recall improves overall without sacrificing precision. This is due to the fact that for a given attribute every RRE covers at least δ true positives, where δ>0. As a result, when the RREs are used in sequence, true positives increase, and false negatives decrease. In general, the precision and recall biases in our algorithm enable the algorithm to discover expressions with both high precision and high recall as measured by F_β.

5. An Active Learning Approach

Although the training-set developer does not need to label the exact location of features in segments, as with all supervised machine learning algorithms our approach still involves manual labeling of segments. In order to further reduce training-set development overhead, we have extended our semi-supervised learning algorithm with an active learning method.

Our active learning algorithm is seeded with an often-used description of an attribute. For example, "six feet tall" may be a seed for *Height*. Based on such choices for seeds, our semi-supervised learning algorithm discovers RREs representing the contexts surrounding the seeds. All segments with similar contexts become candidates for inclusion in the true set for a given attribute.

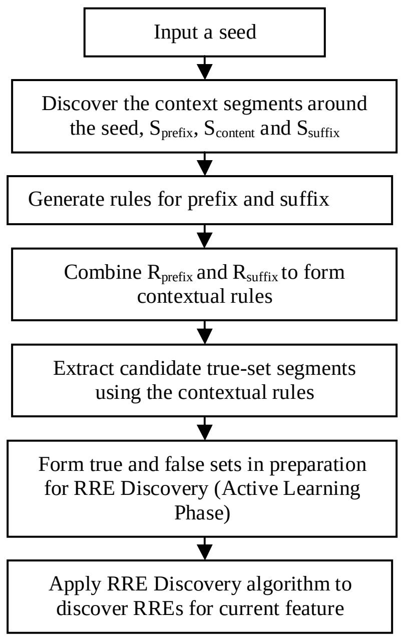

Input a seed
Discover the context segments around the seed, S_{prefix}, $S_{content}$ and S_{suffix}
Generate rules for prefix and suffix
Combine R_{prefix} and R_{suffix} to form contextual rules
Extract candidate true-set segments using the contextual rules
Form true and false sets in preparation for RRE Discovery (Active Learning Phase)
Apply RRE Discovery algorithm to discover RREs for current feature

FIG. 8. Active learning flow chart

The training-set developer need only select those candidates that actually contain a description of the given attribute. After this active learning phase, our semi-supervised learning algorithm is once again applied to discover RREs for each attribute per the process described in section 4 above. Figure 8 portrays a flowchart of the active learning process.

This approach is based on the observation that there is local context that can be leveraged to aid in the discovery process. For example, a person's *Age* may often follow *Race* and precede *Height* in a suspect's physical description. Therefore, if a segment occurs between *Race* and *Height*, it is likely an *Age* segment. In what follows we provide the details of our approach.

5.1. Context Pattern Discovery

There are several steps involved in the discovery of the context of the seed. First, all segments containing the seed in the training dataset are extracted. Let the set $S_{content}$ = {x | x is a segment containing the seed}. Second, let all segments immediately preceding each member of $S_{content}$ be defined as S_{prefix} = {y | y is a segment and the segment immediately following y contains the seed}. Third, let all segments immediately following each member of $S_{content}$ be defined as S_{suffix} = {z | z is a segment and the segment immediately preceding z contains the seed}.

After forming S_{prefix}, $S_{content}$, and S_{suffix}, we employ our semi-supervised learning algorithm to discover RREs for S_{prefix} and S_{suffix}. We then combine the prefix pattern with the suffix pattern to form a complete contextual pattern using the "AND" operator.

To discover the RREs for the prefix, we use S_{prefix} as the true set, and $S_{content} \cup S_{suffix}$ as the false set. The RREs discovered by our semi-supervised learning algorithm form the set R_{prefix} = {p | p is a prefix RRE}.

To discover the suffix RREs, we use S_{suffix} as the true set, and $S_{content} \cup S_{prefix}$ as the false set. The RREs discovered by our semi-supervised learning algorithm form the set R_{suffix} = {s | s is a suffix RRE}.

Combinations of members of R_{prefix} and R_{suffix} form contextual patterns that are used to discover candidates for the true set for the attribute under consideration. Since not all possible combinations of such members occur in the training data, our algorithm employs a data-driven approach that ensures that only combinations that actually occur in the input text are used. As noted, the final expression for extracting candidate true-set segments consists of the logical "AND" of members of S_{prefix} and S_{suffix} with a one-segment gap between. We refer to this final expression as a *contextual expression* in what follows.

5.2. Active selection of true segments

In this step, our active learning algorithm identifies all candidates for a given attribute using the contextual expression discovered as described above in section 5.1. Of course not all such candidates are actually true positive segments, so at this point the training set developer selects the segments that form the final true set. We define the true set T_a = {t | t is a segment accepted by the contextual expression for attribute *a* in which *a* actually occurs}. The remaining segments that are not selected become the false set F_a = {f | f is a segment accepted by the contextual expression that does not contain attribute *a*}.

For a given attribute, the ratio of true to false segments in these two sets is generally greater than the ratio of true to false segments in the training data overall. In other words, substantially less effort is required to develop these two sets using our semi-supervised active learning approach than would be required if these sets were manually generated by labeling every single segment in the training data. In fact, as will be seen in section 6, the reduction in effort is as great as 99%. As will also be seen in section 6, the performance of the RREs discovered with these sets is competitive with the performance of RRE discovery using all of the (manually labeled) training data.

The final true set for a given attribute *a* is $T_a \cup S_{content}$ and the final false set is $S_{prefix} \cup S_{suffix} \cup F_a$. As noted, once these true and false sets have been generated with our active learning algorithm for a given attribute, our semi-supervised learning algorithm is once again applied, this time to discover an expression that extracts features for the attribute of interest.

In summary, in this section, we described how an active learning algorithm is combined with our semi-supervised learning algorithm. In our experimental results reported in section 6 following we provide evidence that this approach significantly reduces training set development effort while at the same time preserving performance in terms of F_β.

6. Experimental Results

In this section, we first depict the results of the application of our semi-supervised learning algorithm using 10-fold cross validation based on 100 police incident reports consisting of 1404 segments. The average number of training examples is 1264 and the average number of test examples is 140 for the attributes evaluated in the first experimental results reported below. Following this, we present the results of the application of our combined semi-supervised active algorithm based on a training dataset (404 items with 2983 segments) and an independent test dataset (149 items with 1462 segments). Finally, we compare the results obtained using these two approaches.

Table 5 summarizes the results of our semi-supervised learning algorithm without active learning. As noted, we employed 10-fold cross-validation. There are ten different attributes evaluated in Table 5 (first column). *Eye Color. Gender* and *Weekday* have perfect performance (100% F_β) in part because we have modified the lexicon as noted in section 4.1.3. The performance of *Age, Date, Hair Color, Height,* and *Race* are also excellent ($F_{\beta=1.0} \geq 90\%$). Even though the performance of *Time* and *Weight* is not as good as other attributes, they still achieve over 85% F_β. As a result, we conclude that the RREs discovered for these ten features are overall of very high quality.

TABLE 5. Results of semi-supervised non-active learning

Attribute	Average Precision	Average Recall	Average F_β (β=1)	Average true positives
Age	97%	89%	93%	12.6

Date	100%	95%	97%	8.9
Time	88%	83%	85%	7.7
Eye Color	100%	100%	100%	1.00
Gender	100%	100%	100%	33.6
Hair Color	90%	90%	90%	0.9
Height	100%	94%	96%	2.2
Race	97%	90%	91%	3
Week day	100%	100%	100%	9.8
Weight	90%	85%	87%	1.7

Table 6 depicts the reduction in training set development effort gained by the use of our active learning algorithm. These results show that our active learning algorithm significantly reduces labeling effort for nine of the ten attributes. The second column of the table is the ratio of true segments to false segments that are generated by the active learning algorithm described in section 5. The third column is the ratio of true segments to false segments overall in the training dataset. The fourth column is the reduction in labeling effort and is calculated as C_4=1.0–C_3/C_2, where C_4 is the value in the fourth column, C_3 is the value in the third column, and C_2 is the value in the second column.

TABLE 6. Labeling effort saved

Attribute	Ratio of $\|T_a\|/\|F_a\|$	Ratio of true/false segments overall	Reduction in labeling effort	Training examples in active learning
Age	85 : 340	539 : 2444	11.78%	492
Date	9 : 1	222 : 2761	99.11%	22
Time	38 : 41	419 : 2564	82.37%	97
Eye Color	6 : 6	28 : 2955	99.05%	60
Gender	599:2168	1003:1980	-83.34%	2969
Hair Color	10 : 17	51 : 2932	97.04%	74
Height	27 : 36	93 : 2890	95.71%	117
Race	34 : 60	127 : 2856	92.15%	224
Week day	127 : 211	401 : 2582	74.20%	497
Weight	10 : 15	66 : 2917	96.61%	54

The results for *Age* and *Gender* in Table 6 are due to the fact that these two attributes occur in many different contexts in the training data. In particular *Gender* occurs in almost every possible context. In other words, our context-based active learning approach is not efficient when the attribute is widely distributed among several contexts in the training data. Note that this does not necessarily imply that the performance of extraction will be degraded – as noted in Table 7, *Gender* maintains F_β of 100%.

TABLE **7**. Results of semi-supervised active learning

Attribute	Precision	Recall	F_β (β=1)	Number of true positives
Age	76%	60%	67%	124
Date	100%	93%	96%	94
Time	98%	97%	98%	127
Eye Color	100%	100%	100%	5
Gender	100%	100%	100%	467
Hair Color	77%	83%	80%	10
Height	89%	89%	89%	16
Race	73%	84%	78%	16
Week day	100%	100%	100%	139
Weight	73%	73%	73%	11

One disadvantage of our active learning approach is that the size of the true set is generally smaller than the true set would be if all of the training data were used. For the *Time* attribute, for example, the true set generated by the active learning process contains only 44 segments (38 in T_a, and 6 in $S_{content}$) while there are 419 true segments in the training set overall. In this particular case, however, our semi-supervised active learning algorithm exceeds the performance of the semi-supervised learning algorithm alone (98% vs. 85%). Nonetheless, in several other cases the performance of our semi-supervised active learning algorithm is degraded compared to the semi-supervised learning algorithm alone. This result is due in part to the aforementioned bias of our active learning algorithm – the number of segments in the true set generated by the active learning process tend to be fewer than the actual number of true segments in the training data overall.

Table 7 summarizes the test-set results for the ten attributes of interest using our semi-supervised active learning algorithm. As before, *Eye Color*, *Gender*, and *Weekday* have 100% $F_{\beta=1.0}$. This implies that the RREs for these three attributes are stable. As noted, the *Time* attribute has a better test result than for the semi-supervised learning algorithm alone. The performance of *Date* in Table 7 is nearly identical to its performance in Table 5. Thus for *Eye Color*, *Time*, *Date*, and *Weekday*, our semi-supervised active learning algorithm not only achieves a significant reduction of labeling effort ($\geq$74%), but also maintains F_β performance equivalent to the performance of our semi-supervised algorithm alone.

The test performances of *Hair Color*, *Height*, *Race*, and *Weight* in Table 7 are not as good as the performance achieved using our semi-supervised algorithm alone (Table 5). Nonetheless, the labeling effort was reduced dramatically for all four of these attributes (≥90%). Given that the reduction in F_β for these same attributes is less than 14%, we see that a tradeoff exists between training set development cost and performance of extraction.

Our semi-supervised active learning approach does not work as well for the *Age* attribute because there is an important sub-pattern of *Age* that is not discovered during the active learning phase. The seed that we used for this sub-pattern only covers a few segments. Therefore, the contextual patterns generated are not general enough to be used to find other similar patterns. As a result, the test performance of *Age* is degraded compared to our semi-supervised algorithm alone.

We also evaluated our semi-supervised active learning algorithm on a collection of narrative text university crime reports. The training set consisted of 74 reports with 160 segments, and the test set contained 48 reports with 117 segments. Four attributes of interest were considered in our experiments: *Date*, *Time*, *Item Value*, and *Gender*. The fourth column of Table 8 portrays the reduction of labeling effort for these four attributes. The reduction is dramatic for the *Date*, *Item Value*, and *Gender* attributes (≥70%). The test performance of these four attributes is also quite good as depicted in Table 9. All have 100% precision. *Date* and *Gender* also have 100% recall. *Time* has almost perfect recall at 99%. In general, our semi-supervised active learning algorithm performed well on these attributes extracted from online university crime reports.

TABLE 8. Labeling effort saved

Attribute	Ratio of $\|T_a\| / \|F_a\|$	Ratio of true/false segments overall	Reduction in labeling effort
Date	46 :1	92 : 68	97.06%
Time	27 : 28	72 : 88	15.15%
Item Value	2 : 3	21 :139	77.34%
Gender	75 : 64	40 : 120	71.56%

TABLE 9. Results of semi-supervised active learning

Attribute	Precision	Recall	F_β (β=1)	Number of true positives
Date	100%	100%	100%	67
Time	100%	98%	99%	47
Item Value	100%	82%	90%	9
Gender	100%	100%	100%	37

In summary, in this section we have presented experimental results for our semi-supervised and semi-supervised active learning algorithms. Our semi-supervised algorithm achieved $F_\beta \geq 85\%$ (Table 5) for all ten attributes of interest. Our combined semi-supervised active learning algorithm achieved significant reductions in training set development effort for all of the attributes considered across both collections of training data[3]. There is however a trade-off for certain attributes between the reduction in training set development effort and extraction performance. Nonetheless, for six of the attributes of interest our algorithm achieved $F_\beta \geq 89\%$ (Table 7), three of which achieved $F_\beta = 100\%$.

In the following section we deal with related information extraction efforts in both academic research projects and commercial products.

7. 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 relevance statistics and make a comparison with three existing commercial products.

Stephen Soderland (1999) developed a supervised learning algorithm, WHISK, which uses regular expressions as patterns to extract features from semi-structured and narrative text. WHISK is an active learning system. In each iteration of the learning process, WHISK requires that a human expert label specific features in instances and then the system 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 the selection of a small number of seeds. As noted this represents a significant reduction in the effort required to develop a training set.

Eric Brill (2000) 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 two major differences between Brill's approach and ours. First, like the aforementioned work by Soderland, Brill's approach requires intensive feature-specific labeling to create the ground truth used in TBL. Secondly, 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 (Chau, Xu, & Chen, 2002). 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

[3] Fairfax County, Virginia police incident reports and online university crime reports.

entities. Although not readily apparent in (Chau, Xu, & Chen, 2002), they evidently employ a similar approach as 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. For instance, an example of a feature extracted for the *Height* attribute that is not a noun phrase is "from five feet eight inches to six feet tall".

Ellen Riloff (1996) developed AutoSlog-TS, a system that discovers patterns based on relevance statistics without exact labeling of attributes. Her approach first extracts all noun-phrases from the training data. Next it generates all possible patterns based on a set of pre-defined templates. It then tests a given pattern's performance according to its relevance statistics. Finally, human experts review and select the most important patterns.

The active learning component of our semi-supervised active learning algorithm can also be thought of as using relevance statistics. On the other hand, our approach does not require pre-defined templates, and as noted the attributes extracted are not limited to noun phrases. Furthermore, our approach does not generate a superfluity of patterns. It is quite possible for AutoSlog-TS to generate tens of thousands of patterns. Our approach usually has less than 50 patterns per attribute.

There are several commercial tools that support information extraction. The more advanced of these systems are (NetOwl), (ClearForest), (AeroText™), and (IBM Intelligent Miner for Text). At the time of this writing, the deficiency in IBM Intelligent Miner for Text is that the system does not support user-defined feature extraction. This severely limits the types of attributes that can be extracted – for example, it is not possible to use the NCIC (2000) codes as a basis for information extraction with these tools despite the fact that these codes form the basis for one of the nation's most advanced Criminal Justice Information Systems. NetOwl, ClearForest and AeroText™, on the other hand, only support the manual creation of user-defined attribute extraction rules. This results in a very high knowledge engineering (training set or rule development) cost for users of such tools. In contrast our approach supports the automatic discovery of user-defined attributes based on our semi-supervised active learning algorithm. This significantly reduces the knowledge engineering (training set development) cost of using our system.

In addition, our segmentation algorithm is more flexible than that provided, for example, in AeroText™. AeroText™ supports only the use of sentences and paragraphs as segments, but in our research we have determined that sub-sentence (i.e., segment) segmentation is required to precisely extract certain attributes. As a result, our segmentation algorithm enables users to define their own segmentation methods.

Another key capability that distinguishes our approach from other tools such as AeroText™ is the use of reduced regular expressions to represent patterns. AeroText™, for example, uses a manual pattern formation method based on assigning words in the input text to bins in the pattern. Theoretically, the rules created using AeroText™ are not as powerful as the reduced regular expressions defined in section 3, and although we have not conducted experiments to empirically verify this fact, our algorithm is as a result theoretically able to represent a wider range of patterns.

There are few if any published results relating to criminal justice that measure the performance of currently available commercial systems. Of the commercial systems we surveyed, results have been published for NetOwl, ClearForest and AeroText™. Based on the widely employed F_β metric (Van Rijsbergen, 1979), NetOwl (Isoquest, Inc, 1998) achieved a performance of 93.99% on the template-element extraction task of MUC-7 (1998). MUC-7, however, is a collection of newspaper articles. Results have been published for ClearForest as the winner of the 2002 KDD Challenge Cup competition in the biomedical domain (Regev, Finkelstein-Landau, & Feldman, 2002). In addition, AeroText™ results were presented in a technical report at the 2002 NIH Biomedical Computing Interest Group (BCIG) seminar (Childs & Weir, 2002). Based on our search of the publicly available information, however, we found no published information extraction performance results in terms of precision, recall, or F-measure related to the criminal justice domain for any of the commercial products we surveyed. This is not to say that these products are not being used effectively in the criminal justice domain – only that we were unable to identify publicly available materials that report performance in terms of these metrics.

8. Future Work

One of the tasks ahead is to further leverage contextual information surrounding a segment to improve the performance of RRE discovery. Currently, the RREs discovered cannot cross segments. Nearby segments, however, contain information helpful in determining whether a feature accepted by an RRE is valid. For example, the RRE discovered for a person's *Race* is "described as {JJ}". This RRE, however, also covers the description of an animal's color. One of false positives accepted by the RRE is "described as a large red" from the segment "described as a large red Boxer" in the sentence "The dog, described as a large red Boxer, bit the boy on the right arm." We hypothesize that if our algorithm can discover and make use of contextual rules such as "if the previous segment contains an animal, then the attribute identified as *Race* is not a *Race*", the false positive rate will be reduced. In fact, we have developed an approach to the automatic discovery of rules of this nature termed Error-Driven Boolean-Logic-Rule-Based Learning (BLogRBL) (Wu, Khan, Fisher, Shuler, & Pottenger, 2002). In future work we plan to use BLogRBL to discover contextual rules that will improve the overall performance of RREs discovered by our semi-supervised active learning algorithm.

We have also applied our semi-supervised algorithm to the extraction of the ‘problem solved’ in US patents (Wu & Pottenger, 2003b). The ‘problem solved’ in a patent identifies the particular solution to an insufficiency in prior art that the patent addresses. Our current results are still preliminary in nature. We have achieved 56% average precision, 38% average recall, and 45% average F_β (β=1) in 10-fold cross-validation. Considering the complexity of natural language expressions used in patents, we consider this result promising. In addition, based on feedback from the corporate sponsor of this work, we have developed a metric that measures the number of patents for which we discover at least one true positive ‘problem solved’. Our performance using this metric is 100%. This research in the patent domain is ongoing.

A drawback of the semi-supervised approach to learning is the difficulty in splitting sub-rules when they overlap. For example, “under the age of 17” and “approximately 25 years of age” are two different expressions of a person’s *Age*. If the word “age” is discovered by the algorithm as the root of a RRE in a situation where both of these representations occur with equal frequency, the algorithm must make an uninformed choice as to which direction to extend the RRE during the “AND” learning phase. As a result, the final expression may contain only the single word “age” with poor overall performance in terms of F_β. The solution to this problem is an open question that must be addressed in the course of future work.

9. Conclusion

We have presented a semi-supervised active learning algorithm as an extension to our semi-supervised algorithm for information extraction from police incident reports and United States patents. Our experiments demonstrate that reduced regular expressions extract information useful in law enforcement applications with high degrees of precision and recall. Furthermore, our algorithm significantly reduces the knowledge engineering (training set or rule development) cost of performing information extraction.

Acknowledgements

This work was funded in part by the Pennsylvania State Police (PSP) under a subcontract with the Lockheed Martin Corporation. We gratefully acknowledge Dr. Biju Kalathil and the Lockheed Martin/PSP team for their assistance in completing this work. Our work was also supported in part by the Eastman Kodak Company, and we gratefully acknowledge Dr. Dan Phelps at Kodak for his contributions to our research efforts. This project was also supported by grant number 2003-IJ-CX-K003 awarded by the National Institute of Justice, Office of Justice Programs, US Department of Justice. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of the US Department of Justice, the Pennsylvania State Police, the Lockheed Martin Corporation or Eastman Kodak Corporation.

We are also grateful for the help of Faisal M. Khan, Lars E. Holzman, Shenzhi Li, other co-workers, family members and friends. We also gratefully acknowledge the continuing help of our Lord and Savior, Yeshua the Messiah (Jesus the Christ) in our lives and work. Amen.

References

AeroText™. Available: http://www.lockheedmartin.com/factsheets/product589.html/

BCIG. Available: http://www.altum.com/bcig/index.htm

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

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

Chau, M., Xu, J., & Chen, H. (2002). Extracting Meaningful Entities from Police Narrative Reports. Proceedings of the National Conference for Digital Government Research, Los Angeles, California.

Childs, L.C., and Weir, C.E. (2002). Drug Discovery through Information Extraction Technology. 2002 NIH Biomedical Computing Interest Group (BCIG) seminar.

ClearForest. Available: http://www.clearforest.com/

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

Hopcroft, J., & Ullman, J. (1979). Introduction to Automata Theory. Languages and Computation, Addison-Wesley.

IBM Intelligent Miner for Text. Available: http://www-3.ibm.com/software/data/iminer/fortext

Isoquest, Inc (1998). Description of the NetOwl™ extractor system as used for MUC-7. Proceedings of the Seventh Message Understanding Conference (MUC-7), Fairfax, Virginia, 29 April – 1 May, 1998.

Manning, C.D., & Schütze, H. (2000) Foundations of Statistical Natural Language Processing. MIT Press.

MUC-7 (1998). The Proceedings of the Seventh Message Understanding Conference. Morgan Kaufmann.

NCIC (2000) code manual. Available: http://www.leds.state.or.us/resources/ncic_2000/ncic_2000_code_manual.pdf

NetOwl. Available: http://www.NetOwl.com/index.html/

Regev, Y., Finkelstein-Landau, M., and Feldman R. (2002). Rule-based Extraction of Experimental Evidence in the

Biomedical Domain – the KDD Cup 2002 (Task 1). SIGKDD Explorations 4(2):90-92.

Reynar, J.C. & Ratnaparkhi, A. (1997). A Maximum Entropy Approach to Identifying Sentence Boundaries. Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, D.C.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 1044-1049.

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

Van Rijsbergen, C.J. (1979). Information Retrieval, 2nd Edition. London: Butterworths.

Witten, I., & Frank, E. (1999). Data Mining Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 1999.

Wu, T. & Pottenger, W.M. (2003a). 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).

Wu, T. & Pottenger, W.M. (2003b). A Supervised Learning Algorithm for Information Extraction from Textual Data. Proceedings of the workshop on Text Mining, Third SIAM International Conference on Data Mining, San Francisco.

Wu, T., Khan, F.M., Fisher, T.A., Shuler, L.A., & Pottenger, W.M. (2002). Error-Driven Boolean-Logic-Rule-Based Learning for Mining Chat-room Conversation. Lehigh University CSE department technical reports. LU-CSE-02-008.