Information Extraction using Non-consecutive Word Sequences

Sachindra Joshi, Ganesh Ramakrishnan, Sreeram Balakrishnan, Ashwin Srinivasan

IBM India Research Labs, IIT Delhi, Hauz Khas, New Delhi, India
{jsachind, ganramkr, srbalakr, ashwin.srinivasan}@in.ibm.com

Abstract. We address an important deficiency in existing machine learning approaches for information extraction from natural language texts. Existing techniques for information extraction employ rules that exploit properties of consecutive word sequences. We argue that sequences of non-consecutive words capturing long range contextual correlations are vital features for information extraction from natural language text. We propose an efficient method that extends the a-priori algorithm to mine frequently occurring non-consecutive word sequences from a given corpus. We also perform a simplistic aggregation of feature information across multiple mentions of an entity in a document to avoid independent classification of the multiple occurrences of the entity. Experiments on some standard data sets show substantial improvements over previously reported results.

1 Introduction

The task of information extraction aims at distilling out facts from documents. Formally, we can consider this as a problem of extracting all instances of a structured target schema E from an unstructured source S. Typically, the target schema is tied to a specific domain and an information extraction task. For the domain of financial news about mergers and acquisition, if we want to extract the names of the acquiring and acquired companies and the deal amount, we would define the set {*acquiring company, deal amount, acquired company*} as the target schema.

Following [2], we propose populating this schema by decomposing the task into two sub-tasks based on the amount of contextual knowledge required at each stage. The first sub-task, called *entity annotation* identifies all the entities in the document that belong to a generic type such as 'person name', 'company', 'place' or 'date'. In general, this sub-task can be accomplished using information obtained from intrinsic features of the entity or features of the immediate neighboring tokens. The second sub-task assigns a role to each entity of a particular type. This usually requires detailed knowledge of the surrounding context in which the entity occurs. We refer to this sub-task as *role labeling*. In our financial example, this corresponds to assigning the role of 'acquiring' or 'acquired' to an entity of type 'company'. The focus of this paper is on the second sub-task. We use a simple rule-based annotator for the first sub-task although any one of many alternative rule and machine learning approaches from the literature could be applied. We use very general rules to achieve high recall for entity identification.

Rule-based approaches have been widely employed for information extraction [11, 7, 3]. An information extraction rule consists of a pattern and an action. A pattern typically uses words or sequences of consecutive words, to impose contextual constraints. When a group of words satisfies these constraints, the corresponding action is taken, to assign a *role* to the group of words. The rules can be either manually developed or can be learned automatically.

In this paper, we argue that patterns that capture non-consecutive word sequences are important for natural language text. Natural language expressions allow inclusion of complex modifiers. The

COMP1 purchased COMP2 for an amount of $2 billion.
COMP1 recently **purchased** COMP2 for a cost of $2 billion.
COMP1 said that it **purchased** COMP2 for $2 billion.
COMP1 announced that it **purchased** COMP2 for $2 billion.

Fig. 1. Example Sentences for Acquiring Company

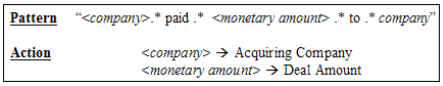

Fig. 2. An Example Rule R

intervening modifiers improve the richness of expressions and are thus important in natural languages. However, these variations make certain highly co-related words non-consecutive. Figure 1, presents a set of example sentences. All the sentences convey the same information, namely, a company acquiring another company. However, they slightly differ on the surface due to the use of modifiers, *etc.* The rule $\{$"$<ALLCAPS>$ *purchased*" $\Longrightarrow$ $<ALLCAPS>$= *Acquiring Company*$\}$ will extract information only from the first example sentence in Figure 1 and will fail on the rest. Patterns involving non-consecutive word sequences are required to enable information extraction from all the sentences given in Figure 1.

In the past, several methods that identify contextual constraints in the form of patterns, have been proposed for the information extraction task. The approach proposed in [11] requires the contextual constraints to be identified manually. Finkel et al [5] incorporate non-local structure in CRFs while preserving tractable inference to augment an existing CRF-based information extraction system with long-distance dependency models. However, they require the non-local features to be manually engineered. Finn *et. al.* [6] formulate IE as a task of classifying each token as either the start of a field to extract, as an end of a field or neither. They use neighbouring tokens as features in the classification task and do not consider the engineering of patterns of non-consecutive tokens as features. The techniques presented in [7, 3] capture contextual patterns of consecutive tokens only. Further, several of these methods [3, 12] employ greedy covering techniques that yield sub-optimal patterns. On the contrary, the method proposed in this paper generates all patterns that satisfy pre-specified *support* and *confidence* thresholds. Yunbo Cao *et. al.* [4] report a technique for information extraction that identifies patterns with non-consecutive token sequences. However, their method can be used only when the schema has a single target element. Some methods [12] attempt to capture long-range dependencies through the syntactic parsing of sentences. Unfortunately, syntactic parsing can be time-intensive and error-prone, especially on grammatically poor texts.

Several methods that identify consecutive word sequences or n-grams, have been proposed in the past [14]. Some of these methods identify only significant n-grams. The total number of consecutive word sequences of length n in a word sequence of length m is $(m - n + 1)$. Let δ be an upper bound on the number of intervening words between every pair of successive words in a non-consecutive word sequence. The total number of non-consecutive word sequences of length n in a word sequence of length m is at least $((m - n\delta + \delta) * \delta^{(n-1)})$, which is of order $O(m\delta^n)$. This is a huge search space that warrants more efficient methods for identifying *significant* non-consecutive word sequences. We propose a method based on the a-priori algorithm [1], to mine *significant* non-consecutive word sequences. The significance of word sequences are captured through the *support* measure. The *support* measure which we introduce in this paper for word sequences, can be efficiently measured [10] using operations on the inverted index of the document collection.

In this paper, we also use a simple method to aggregate contextual information available across the different occurrences of an entity in a document, before assigning it a role. We make an assumption that all the occurrences of an entity in a document have the same role. Entities of a particular type are grouped together such that all the entities belonging to a group refer to the same entity. Grouping is done on the basis of exact string match and substring relationships to capture abbreviated forms. More sophisticated techniques for co-reference resolution [13] can also be adopted. The set of rules discovered by the above mentioned algorithm are applied to each entity in the group and the output is aggregated into a single feature vector for the group. Standard classification techniques (SVMs, majority voting) are then used to obtain the final role assignment for all the entities in the cluster. Our experimental results show that non-consecutive word sequences significantly improve the IE performance for natural language text. Our results show that even the simplistic method for aggregation, substantially improves the performance. Techniques for collective information extraction [2] can be employed to perform aggregation in a more principled manner.

The organization of the rest of the paper is as follows. In Section 2 we describe the details of the two sub-tasks of our approach: firstly the entity annotator and secondly the rules for determining the role of the entity together with the algorithm for aggregating the output of these rules over multiple entities. Subsequently, in Section 3, we present details of the method for learning rules that determine roles. In Section 4, we present experimental evaluation of the proposed method. We finally conclude and describe future work.

2 Entity Extraction and Role labeling

2.1 Identification of entities

There are many alternative rule-based and machine learning-based methods [9] for building entity annotators capable of identifying token sequences in a document which are instances of *entity types*, such as, 'company name', 'currency measure', 'place', *etc.*. For this paper, we used a rule-based annotator that has rules in first order logic and that involve predicates over orthographic features and dictionary containment properties of tokens [10]. We use very generic entity annotation rules to achieve high recall. Low precision of our named entity annotation does not negatively effect the overall performance, since the surrounding context information is used to assign a role to an entity.

2.2 Representation of rules for role assignment

In this subsection, we describe the structure of the rules that we use for assigning roles to entities. Each rule has a *pattern* p and an *action* a. Figure 2 shows an example rule R. The *pattern* part of the rule is a regular expression over tokens, each token being either a word or an entity type. To enable the reader distinguish between words and entity types, instances of the latter are shown in italized font. A role is assigned to an entity based on its type and is independent of the actual entity itself. If the type of an entity should be associated with a role, the corresponding instance of the type is called an *actionable type instance*. Actionable type instances are denoted by prefixing the type instance with a '$<$' and suffixing with a '$>$'. '$<$*company*$>$' is an example actionable type instance. The *pattern* specifies which type instances are actionable.

The *pattern* in R matches occurrences of the following sequence of tokens, while allowing a bounded number of intervening words between successive tokens: (1) an entity of type 'company'

(2) the word 'paid' (3) an entity of type 'monetary amount' (4) the word 'to' (5) an entity of type 'company'. Moreover, the pattern specifies that a role has to be associated with entities at positions (1) and (3).

The *action* part specifies the role that should be assigned to each actionable type instance, in the order of its position in the pattern. The action part of R assigns the role of 'Acquiring Company' to the first token and 'Deal Amount' to the third token. Since roles are associated only with entity types, the actual entities are not required in the training data. Each entity in the training data is therefore replaced with its entity type.

2.3 Information extraction from a test document

We obtain a set of rules for information extraction using a training data set and the algorithms described in the next two sections. In the rest of this section, we describe the method that we employ for assigning a role to an entity using the set of rules. The number of extracted rules will typically be in several thousands. Further, multiple rules may assign different roles to the same entity in a document. Therefore, we generate a feature set for each entity using the rules and then use a classifier to assign a unique role to it based on the feature set.

For each rule r, we define a feature F_r, such that, given an entity, F_r assumes the value of the role assigned by r to the entity and assumes the value 'NONE' if no role is assigned. Consider the annotated document D in Figure 3. The rule R when applied to D, assigns the role of 'Acquiring Company' to 'XYZ Ltd.' and no role to 'ABC Inc.' The feature F_R therefore takes the value 'Acquiring Company' for the first entity and 'NONE' in the case of the second.

In general, there could be a set of rules $\mathcal{R}$ that assign a set of roles to different entities in a document. The IE task boils down to aggregating the decisions of the different rules on each entity and making a consolidated decision on the role to be assigned to the entity. Further, an entity may be referred to using multiple ways in a document. Aggregating contextual information across different references for the same entity yields higher evidences for role assignment. More precisely, we perform information extraction from a test document using the following four steps:

1. Entities and their types are identified from the test document.

2. For each entity type, co-referring entities are grouped. An entity A is said to refer to an entity B *iff* either $A = B$ or A is a substring of B. The longest member string of a group is treated as the canonical form for that group. *E.g:* For the document D, there are three groups for the entity type 'company': ($\mathcal{C}_1$) group of 'XYZ Ltd.' and 'XYZ', with the former as the canonical form, ($\mathcal{C}_2$) the single entity 'ABC Inc.' and ($\mathcal{C}_3$) the single entity 'PQR News'.

3. We next obtain a set of feature values for each group. Each rule is applied on the document to determine feature values for each entity. A feature F_r may assume different values for different members that belong to the same group $\mathcal{C}$. We pool all these values into a single set and apply an aggregation function to this set that returns a single value. Currently the aggregation function we use returns the most frequent value other than 'NONE'. A value of 'NONE' is returned only if the set consists exclusively of 'NONE' values. Thus, F_R takes the following values for different 'company' groups in D: 'Acquiring Company' for ($\mathcal{C}_1$) and 'NONE' for ($\mathcal{C}_2$) and ($\mathcal{C}_3$).

4. The set of values taken by a set of features $\{F_r\}$ for a group $\mathcal{C}$, is used to determine a role $\mathcal{Z}$ to be finally assigned to each member of $\mathcal{C}$. We determine $\mathcal{Z}$ by performing a majority voting over the values assigned by the features in $\{F_r\}$, weighed by the support of rule r. As an example, given an entity and each rule with a unit support, if 4 features take the value 'NONE', 6 take the value 'Acquiring Company' and 2 take the value 'Acquired Company', the voting–based technique will assign the role

<company>*XYZ Ltd.*</company> paid an advance of <currency>*10 mln dlrs*</currency> to <company>*ABC Inc.*</company> toward purchase of the latter's drilling unit. <company>*PQR News*</company> had announced <company>*XYZ*</company>'s acquisition plan in <date>*June 2005*</date>.

Fig. 3. A sample annotated document D

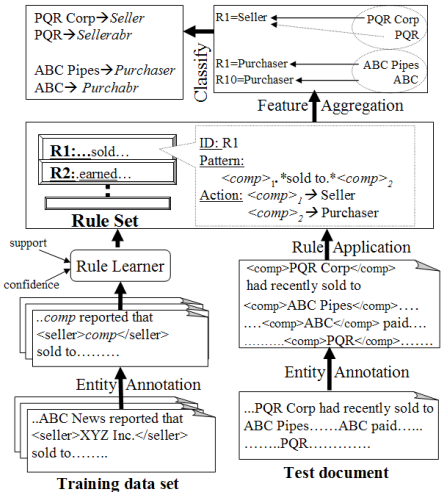

Fig. 4. The Complete Process for Information Extraction

'Acquiring Company' to the entity. Alternatively, the role assignment can be done using some classifier trained on the values of $\{F_r\}$ assumed over training instances.

Figure 4 outlines the complete process that we employ for information extraction.

3 Generating Rules for Role Assignment

3.1 Generating Patterns

The pattern of a rule, in general, can be any regular expression over tokens, each token being a word or an entity type. Further, the entity type may be an actionable type instance. Thus there are three categories of tokens that appear in a pattern, *viz.*, (1) words, (2) entity type instances and (3) actionable entity type instances. Each occurrence of (3) is also considered to be an occurrence of (2).

The search space for learning arbitrary regular expressions over these tokens is very large. Hence, we restrict the search space, to regular expression patterns that are sequences of tokens, with a bounded number of words allowed between two successive tokens. We observe that words influence the role assignment of only closest type instances and therefore do not allow intervening tokens to be instances of entity types.

Each such pattern can be equivalently represented as a *token-sequence*. For example, the pattern in Figure 2 can be equivalently written as the following token sequence: '[<*company*>, paid, <*monetary amount*>, to, *company*]'. We will denote a token-sequence comprising n tokens by s_n.

Definition 1. *A document* d *is said to contain an instance of token-sequence* s_n*;* $d \sqsupseteq s_n$ *;iff all tokens in* s_n *appear in* d *in the same order, with a fixed upper bound,* δ*, on the number of intervening words between every pair of successive tokens in* s_n*.*

Let $\delta = 4$. The document D in Figure 3 has an instance of '[<company>, paid, <currency>]'. On the other hand, D does not have any instance of '[toward, unit]', because there are more than δ intervening

words. Similarly, D does not have any instance of '[<currency>, toward]', since there is an intervening occurrence of 'company' type.

Let $\mathcal{D}$ be a set of training documents of size $|\mathcal{D}|$. $freq(s_n) = |\{d|d \in \mathcal{D}, d \sqsupseteq s_n\}|$ is the number of documents in $\mathcal{D}$ that contain instances of s_n. $freq(s_n)$ can also be counted at a granularity finer than document, such as at the sentence level. However, $freq(s_n)$ considers overlapping occurrences of s_n to be a single occurrence[1]. Note that this definition of $freq(s_n)$ is well-suited for the information extraction task.

We define support $sup(s_n)$ as $sup(s_n) = \frac{freq(s_n)}{|\mathcal{D}|}$. Let $minSup$ be a threshold on support. We define $\mathcal{S}_n$ as the set of all token sequence of length n that have support greater than or equal to the minimum support $minSup$, *i.e.*, $\mathcal{S}_n = \{s_n \,|sup(s_n) \geq minSup\}$. Let $\mathcal{S}_*$ be the set of token-sequences s_* such that each s_* has length between 1 and a threshold N and $sup(s_*)$ is above the threshold $minSup$. $\mathcal{S}_*$ is identified using the algorithm described in Figure 5. The algorithm is inspired by a-priori [1]. The a-priori algorithm optimally and efficiently yields all item-sets or item-sequences, that have their support value above a given threshold value. We next prove that the algorithm given in Figure 5 optimally discovers all non-consecutive token sequences.

Definition 2. *Let s_i and s_j be two token sequences where $j < i$. We say $s_i \supset s_j$; iff s_j is a contiguous subsequence of s_i.*

E.g. Token sequences $<t_1, t_2>$ and $<t_2, t_3, t_4>$ are contiguous subsequences of $s_4 = <t_1, t_2, t_3, t_4>$. On the other hand, $<t_1, t_3, t_4>$ is not a contiguous subsequence of s_4. Note that every document that has an instance of a sequence also has an instance of each of its contiguous subsequences, *i.e.* $\forall d \in \mathcal{D}, d \sqsupseteq s_n \Rightarrow \forall s_n \supset s_i, d \sqsupseteq s_i$. This implies $\forall s_n \supset s_i, sup(s_n) \leq sup(s_i)$.

Theorem 1. *If $s_n \in \mathcal{S}_*$, then $\forall s_i | s_n \supset s_i, s_i \in \mathcal{S}_*$.*

Proof: We prove the theorem by contradiction. Let there be an $s_i, i < n$, *s.t.* $s_n \supset s_i$ and $s_i \notin \mathcal{S}_*$. This implies $sup(s_i) < minSup$. However, $sup(s_n) \leq sup(s_i)$. Therefore, $sup(s_n) < minSup$ which implies $s_n \notin \mathcal{S}_*$. This is a contradition.

Using the result of Theorem 1, we iteratively build token-sequences of length $n + 1$ from token sequences of length n. The following corollary is important, as it suggests that a set of patterns obtained with particular values of δ and N subsumes all sets obtained using lower values of the respective parameters.

Corollary 1. *Let $S_*(\delta, N)$ be the set of all non consecutive token sequences upto a length of N such that a maximum of δ intervening words are permitted between any two successive tokens. Then, $\forall N' \leq N, \delta' \leq \delta, S_*(\delta', N') \subseteq S_*(\delta, N)$.*

3.2 Generating Rules

We generate a set of rules, such that each rule has a *pattern* and an *action* as defined earlier. In order to generate an *action* part for a *pattern*, the *patten* should have at least one actionable type instance. To ensure this, all token-sequences that have no instances of actionable entity types are removed from $\mathcal{S}_*$. Given a confidence threshold $minConf$, a set of confident rules $\mathcal{R}_*$ is generated in the following manner (the algorithm is outlined in Figure 6):

[1] If overlapping occurrences of a token sequence are considered as multiple occurrences, the monotonicity property of $freq(s_n)$ will not hold [15].

Find $\mathcal{S}_1$, the set of all 1-item-sequences;
$n = 1$
$\mathcal{S}_* = \{\}$
while $n \leq N$ **do**
  $\mathcal{S}_{n+1} = \{\}$
  for Each $s_n, {s'}_n \in \mathcal{S}_n$ **do**
    if s_n and ${s'}_n$ have a subsequence of length $n-1$ in common **then**
      Merge s_n and ${s'}_n$ to obtain s_{n+1}
      if $sup(s_{n+1}) \geq minSup$ **then**
        $\mathcal{S}_{n+1} = \mathcal{S}_{n+1} \cup \{s_{n+1}\}$
      end if
    end if
  end for
  for Each $s_n \in \mathcal{S}_n$ **do**
    if $\neg(\exists s_{n+1} \in \mathcal{S}_{n+1} | s_{n+1} \supset s_n)$ **then**
      $\mathcal{S}_* = \mathcal{S}_* \cup \{s_n\}$
    end if
  end for
  $n = n + 1$
end while

Fig. 5. The algorithm for generating the set $\mathcal{S}_*$ of token-sequences with high support

$\mathcal{R}_* = \{\}$
for Each $s_* \in \mathcal{S}_*$ **do**
  Generate the set of all possible actions $\mathcal{A}_*$
  for Each $a_* \in \mathcal{A}_*$ **do**
    if $conf(a_*, s_*) \geq minConf$ **then**
      Add the rule $R = [a_*, s_*]$ to the set $\mathcal{R}_*$
    end if
  end for
end for
Output the set of rules $\mathcal{R}_*$

Fig. 6. The algorithm for generating the set $\mathcal{R}_*$ of confident rules

1. For each pattern $s_* \in \mathcal{S}_*$, a set $\mathcal{A}_*$ of all possible actions is generated. Each action $a_* \in \mathcal{A}_*$ specifies a unique assignment of roles to the actionable entities in s_*. *E.g:* Given the pattern in Figure 2, '<*company*>⇒Purchaser, <*monetary amount*>⇒Deal Amount' is an action, different from the action specified in the same figure.

2. The confidence $conf(a_*, s_*)$ of each action a_* for the sequence s_* is computed. We define $conf(a, s)$ as $conf(a, s) = \frac{freq(s,a)}{freq(s)}$, where $freq(s, a)$ is the number of instances of s (in $\mathcal{D}$) that satisfy the role assignment specified by a.

3. A rule R is constructed for each pair of pattern and action $[s_*, a_*]$ that has confidence above the $minConf$ threshold.

4 Experimental Results

We present experimental results on nine information extraction problems from two corpora. The first data set is a collection of 600 news articles describing *corporate acquisition events* taken from the Reuters data set. Each news article has a 'purchaser' , 'acquired', and a 'seller' company along with their abbreviated forms, *viz.*, 'purchabr', 'acqabr' and 'sellerabr' respectively. We focus only on the extraction task for these IE target elements, since all of them correspond to company entity type and differ only in their roles. In the rest of this section, we refer to 'purchaser', 'acquired' and 'seller' as *complete roles* and 'purchabr', 'acqabr', and 'sellerabr' as *abbreviated roles*. The second data set is a collection of *seminar announcements* posted to local newsgroups at a large university. Both of these

data sets as well as the IE problems defined for them, are described in detail in previously published work [8].

Each result reported in this section is an average over 10 random 50-50 train-test splits. The rules are learnt from a training data set. By default, all the reported results use rules generated with a minimum support of 0.01 and minimum confidence value of 0.9. In our experiments, we used the values $\delta = 4$ and $N = 4$. Table 1 shows some example rules generated by our algorithm on the two data sets. A

Pattern	Action
<company>$_1$.*to.*acquire.*<company>$_2$	<company>$_1 \rightarrow purchaser$ <company>$_2 \rightarrow acquired$
<company>.*sell.*its	<company> $\rightarrow seller$
who.*:.*<person>	<person> $\rightarrow speaker$

Table 1. Examples of Extracted Rules

feature set is generated for each entity using the extracted rules. We used a majority voting classifier weighed by the support of each rule for the role assignment. Experiments were also conducted with an SVM classifier which yielded similar results. We performed experiments with and without aggregation referred to as 'with aggn.' and 'w/o aggn.' respectively, in the result tables. While classifying without aggregation, the classifier directly assigns a role to each entity. In case of aggregation, all the company names in a document are first grouped using the substring relationship. This groups together company names along with their abbreviated forms. An aggregated feature set is generated for each group using the feature sets of its members. The classifier is then used to assign a complete role to each group. Once a complete role is assigned to a group, the longest company name in that group is assigned the complete role and all other members of it are assigned the abbreviated role corresponding to the complete role.

As a first-cut approach, we trained an SVM classifier to assign roles to entities based on their (1) types and (2) neighboring words along with information of whether they appear on the left or right of the entity. We obtained the following F1 using this approach: 'acquired'= 38.1, 'acqabr'=36.1, 'purchaser'=42.4 , 'purchabr'=34.0, 'seller'=41.3 and 'sellerabr'=21.11. Next, we report results on the *acquisition* data set with longer contextual patterns. Table 2, presents the results with and without the aggregation of information across company names using weighted majority voting. The results with aggregation outperform results without aggregation. Note that while aggregation improves F1 measure for complete roles slightly, it improves F1 for abbreviated roles substantially. This is true as entities associated with abbreviated roles are mentioned at multiple places in a document with different contexts. Context associated with some of the instances is not sufficient for the correct assignment of roles and aggregating information improves the performance.

In Table 3, we compare the best results obtained by our approach for a subset of roles on the *acquisition* data sets for which previously published results are known, *viz.*, SRV [7], HMM [8] and Elie [6]. We achieve significantly better results for all the roles even without aggregation. This result underscores the importance of long-range contextual constraints specified by the rules.

In Table 4, we present the results obtained on the *seminar announcement* data set. Note that the gains achieved on *acquisition* data set is significantly more than the gains achieved on the *seminar* data set. The reason is that the *acquisition* data set contains plenty of natural language text (compared to the *seminar* data set) and patterns of non-consecutive word sequences prove to be vital contextual features,

Entity type	W/o aggn.			With aggn.		
	P	R	F1	P	R	F1
acquired	54.4	44.8	49.1	62.8	54.6	58.4
acqabr	44.0	47.9	45.8	65.2	56.7	60.6
purchaser	57.6	50.8	53.9	64.1	52.3	57.6
purchabr	56.8	31.3	40.3	71.3	49.6	58.4
seller	62.7	43.8	51.4	68.7	41.7	51.7
sellerabr	68.9	10.1	17.1	70.3	35.7	47.1

Table 2. Performance of Weighted Majority Voting

Entity	SRV	HMM	Elie	w/o aggn.	with aggn.
acquired	34.3	30.9	42.0	49.1	**58.4**
acqabr	35.1	40.1	40.0	45.8	**60.6**
purchaser	42.9	48.1	47.0	53.9	**57.6**
purchabr	−−	−−	29.0	40.3	**58.4**
seller	−−	−−	15.0	51.4	**51.7**
sellerabr	−−	−−	14.0	17.1	**47.1**
Macro F1 for 3 rows	37.4	39.7	43.0	49.6	**58.9**
Macro F1	-	-	31.16	42.9	**55.6**

Table 3. Comparison of F1 against SRV, HMM Shrinkage and Elie on *acquisition* data set

Entity	SRV	HMM	Elie	w/o aggn.	with aggn.
speaker	70.3	71.1	**88.0**	69.3	69.3
location	72.3	83.9	86.0	**87.8**	**87.8**
stime	98.8	**99.1**	98.0	94.2	94.2
etime	83.9	59.5	95.0	**97.6**	**97.6**
Macro F1	81.3	78.4	**91.7**	87.2	87.2

Table 4. Comparison of F1 on *seminar announcement* data set against SRV, HMM and Elie

Entity type	W/o context.			With context		
	P	R	F1	P	R	F1
speaker	15.0	98.0	26.0	80.7	60.8	69.3
location	27.6	99.2	43.2	90.7	85.1	87.8

Table 5. Improvements achieved using contextual information

producing significant gains of over 10% in many cases. Also, aggregation of information proves to be helpful for such data sets.

In order to evaluate the importance of contextual information for role assignment, we conducted a small experiment on the *seminar announcement* data. All the identified 'person' name entities were assigned a role of 'speaker' and all the identified 'place' names were assigned a 'location' role. In Table 5, we show the precision and recall numbers achieved. Using the contextual information, we get significant improvement in precision. The recall decreases but the overall F1 improves substantially. This experiment illustrates that our technique of assigning roles to entities can be effectively used in conjunction with high recall entity annotators. Thus, the proposed technique can be easily customized to a new domain, provided that a set of relevant high recall annotators can be built.

The Figures 7 and 8 describe the effect on performance with an increase in the value of minimum support and minimum confidence respectively for the 'acquired' role. As the minimum support increases, the recall decreases, affecting the F1 measure negatively. This is justified, since a fewer number of rules qualify with a higher value of the minimum support. The precision improves with an increase in the value of minimum confidence, since rules with a higher confidence value are more likely to be correct.

5 Conclusion

In this paper, we addressed an important deficiency in existing automated approaches for information extraction - the lack of features that capture sequences of non-consecutive contextual words. We devel-

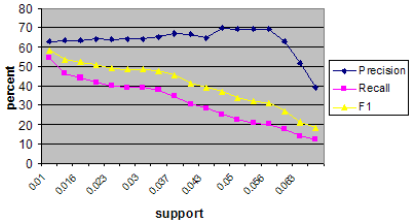

Fig. 7. Variations in F1, precision and recall, plotted against support for the 'acquired' role

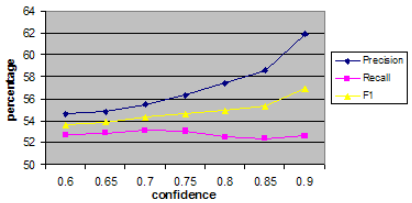

Fig. 8. Variations in F1, precision and recall, plotted against confidence for the 'acquired' role

oped an efficient method for optimally determining frequent sequences of non-consecutive words by extending the a-priori algorithm. Our experimental results show significant performance improvement over the existing techniques for a data set that exhibits richness of natural language, and comparable results for another dataset. The use of simple methods for aggregation further improves the performance.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
2. R. Bunescu and R. J. Mooney. Relational markov networks for collective information extraction. In *ICML Workshop on Statistical Relational Learning and its Connections to Other Fields*, 2004.
3. M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 1998.
4. Y. Cao, H. Li, and S. Li. Learning and exploiting non-consecutive string patterns for information extraction. In *MSR-TR-2003-33*, 2003.
5. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.
6. A. Finn and N. Kushmerick. Multi-level boundary classification for information extraction. In *ECML*, 2004.
7. D. Freitag. Toward general-purpose learning for information extraction. In *ACL*, 1998.
8. D. Freitag and A. K. McCallum. Information extraction with hmms and shrinkage. In *AAAI Workshop on Machine Learning for Information Extraction*, 1999.
9. D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. Named entity recognition from diverse text types. In *RANLP*, 2001.
10. G. Ramakrishnan, S. Balakrishnan, and S. Joshi. Entity annotation based on inverse index operations. In *EMNLP*, 2006.
11. E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *National Conference on Artificial Intelligence*, 1993.
12. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3), 1999.
13. W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 2001.
14. C.-M. Tan, Y.-F. Wang, and C.-D. Lee. The use of bigrams to enhance text categorization. *Information Process Management*, 2002.
15. M. Zhang, B. Kao, D. W. Cheung, and K. Y. Yip. Mining periodic patterns with gap requirement from sequences. In *SIGMOD*. ACM Press, 2005.