Automatic Acquisition of Domain Knowledge for Information Extraction

Roman Yangarber, Ralph Grishman
Courant Institute of
Mathematical Sciences
New York University
{roman|grishman}@cs.nyu.edu

Pasi Tapanainen
Conexor oy
Helsinki, Finland
Pasi.Tapanainen@conexor.fi

Silja Huttunen
University of Helsinki
Finland
sihuttun@ling.helsinki.fi

Abstract

In developing an Information Extraction (IE) system for a new class of events or relations, one of the major tasks is identifying the many ways in which these events or relations may be expressed in text. This has generally involved the manual analysis and, in some cases, the annotation of large quantities of text involving these events. This paper presents an alternative approach, based on an automatic discovery procedure, ExDisco, which identifies a set of relevant documents and a set of event patterns from *un-annotated* text, starting from a small set of "seed patterns." We evaluate ExDisco by comparing the performance of discovered patterns against that of manually constructed systems on actual extraction tasks.

0 Introduction

Information Extraction is the selective extraction of specified types of information from natural language text. The information to be extracted may consist of particular semantic classes of objects (*entities*), relationships among these entities, and events in which these entities participate. The extraction system places this information into a data base for retrieval and subsequent processing.

In this paper we shall be concerned primarily with the extraction of information about events. In the terminology which has evolved from the Message Understanding Conferences (muc, 1995; muc, 1993), we shall use the term *subject domain* to refer to a broad class of texts, such as *business news*, and the term *scenario* to refer to the specification of the particular events to be extracted. For example, the "Management Succession" scenario for MUC-6, which we shall refer to throughout this paper, involves information about corporate executives starting and leaving positions.

The fundamental problem we face in porting an extraction system to a new scenario is to identify the many ways in which information about a type of event may be expressed in the text. Typically, there will be a few common forms of expression which will quickly come to mind when a system is being developed. However, the beauty of natural language (and the challenge for computational linguists) is that there are many variants which an imaginative writer can use, and which the system needs to capture. Finding these variants may involve studying very large amounts of text in the subject domain. This has been a major impediment to the portability and performance of event extraction systems.

We present in this paper a new approach to finding these variants automatically from a large corpus, without the need to read or annotate the corpus. This approach has been evaluated on actual event extraction scenarios.

In the next section we outline the structure of our extraction system, and describe the discovery task in the context of this system. Sections 2 and 3 describe our algorithm for pattern discovery; section 4 describes our experimental results. This is followed by comparison with prior work and discussion in section 5.

1 The Extraction System

In the simplest terms, an extraction system identifies patterns within the text, and then maps some constituents of these patterns into data base entries. (This very simple description ignores the problems of anaphora and intersentential inference, which must be addressed by any general event extraction system.) Although these patterns could in principle be stated in terms of individual words, it is much easier to state them in terms of larger syntactic constituents, such as noun phrases and verb groups. Consequently, extraction normally consists of an analysis of the text in terms of general linguistic structures and domain-specific constructs, followed by a search for the scenario-specific patterns.

It is possible to build these constituent structures through a full syntactic analysis of the text, and the discovery procedure we describe below would be applicable to such an architecture. However, for reasons of speed, coverage, and system robustness, the more common approach at present is to perform a partial syntactic analysis using a cascade of finite-state transducers. This is the approach used by our extraction system (Grishman, 1995; Yangarber and Grishman, 1998).

At the heart of our system is a regular expression pattern matcher which is capable of matching a set of regular expressions against a partially-analyzed text and producing additional annotations on the text. This core draws on a set of knowledge bases of varying degrees of domain- and task-specificity. The lexicon includes both a general English dictionary and definitions of domain and scenario terms. The concept base arranges the domain terms into a semantic hierarchy. The predicate base describes the logical structure of the events to be extracted. The pattern base consists of sets of patterns (with associated actions), which make reference to information from the other knowledge bases. Some pattern sets, such as those for noun and verb groups, are broadly applicable, while other sets are specific to the scenario.

We have previously (Yangarber and Grishman, 1997) described a user interface which supports the rapid customization of the extraction system to a new scenario. This interface allows the user to provide examples of relevant events, which are automatically converted into the appropriate patterns and generalized to cover syntactic variants (passive, relative clause, etc.). Through this interface, the user can also generalize the pattern semantically (to cover a broader class of words) and modify the concept base and lexicon as needed. Given an appropriate set of examples, therefore, it has become possible to adapt the extraction system quite rapidly.

However, the burden is still on the user to find the appropriate set of examples, which may require a painstaking and expensive search of a large corpus. Reducing this cost is essential for enhanced system portability; this is the problem addressed by the current research.

How can we automatically discover a suitable set of candidate patterns or examples (patterns which at least have a high likelihood of being relevant to the scenario)? The basic idea is to look for linguistic patterns which appear with relatively high frequency in relevant documents. While there has been prior research on identifying the primary lexical patterns of a sublanguage or corpus (Grishman et al, 1986; Riloff, 1996), the task here is more complex, since we are typically not provided in advance with a sub-corpus of relevant passages; these passages must themselves be found as part of the discovery procedure. The difficulty is that one of the best indications of the relevance of the passages is precisely the presence of these constructs. Because of this circularity, we propose to acquire the constructs and passages in tandem.

2 ExDisco: the Discovery Procedure

We first outline ExDisco, our procedure for discovery of extraction patterns; details of some of the steps are presented in the section which follows, and an earlier paper on our approach (Yangarber et al., 2000). ExDisco is an unsupervised procedure: the training corpus does not need to be annotated with the specific event information to be extracted, or even with information as to which documents in the corpus are relevant to the scenario. The only information the user must provide, as described below, is a small set of seed patterns regarding the scenario.

Starting with this seed, the system automatically performs a repeated, automatic expansion of the pattern set. This is analogous to the process of automatic term expansion used in some information retrieval systems, where the terms from the most relevant documents are added to the user query and then a new retrieval is performed. However, by expanding in terms of patterns rather than individual terms, a more precise expansion is possible. This process proceeds as follows:

0. We start with a large corpus of documents in the domain (which have not been anno-

tated or classified in any way) and an initial "seed" of scenario patterns selected by the user — a small set of patterns whose presence reliably indicates that the document is relevant to the scenario.

1. The pattern set is used to divide the corpus U into a set of relevant documents, R (which contain at least one instance of one of the patterns), and a set of non-relevant documents $\overline{R} = U - R$.
2. Search for new candidate patterns:
   - automatically convert each document in the corpus into a set of candidate patterns, one for each clause
   - rank patterns by the degree to which their distribution is correlated with document relevance (i.e., appears with higher frequency in relevant documents than in non-relevant ones).
3. Add the highest ranking pattern to the pattern set. (Optionally, at this point, we may present the pattern to the user for review.)
4. Use the new pattern set to induce a new split of the corpus into relevant and non-relevant documents. More precisely, documents will now be given a relevance confidence measure; documents containing one of the initial seed patterns will be given a score of 1, while documents which are added to the relevant corpus through newly discovered patterns will be given a lower score. Repeat the procedure (from step 1) until some iteration limit is reached, or no more patterns can be added.

3 Methodology

3.1 Pre-processing: Syntactic Analysis

Before applying ExDisco, we pre-processed the corpus using a general-purpose dependency parser of English. The parser is based on the FDG formalism (Tapanainen and Järvinen, 1997) and developed by the Research Unit for Multilingual Language Technology at the University of Helsinki, and Conexor Oy. The parser is used for reducing each clause or noun phrase to a tuple, consisting of the central arguments, as described in detail in (Yangarber et al., 2000). We used a corpus of 9,224 articles from the Wall Street Journal. The parsed articles yielded a total of 440,000 clausal tuples, of which 215,000 were distinct.

3.2 Normalization

We applied a name recognition module prior to parsing, and replaced each name with a token describing its class, e.g. *C-Person*, *C-Company*, etc. We collapsed together all numeric expressions, currency values, dates, etc., using a single token to designate each of these classes. Lastly, the parser performed syntactic normalization to transform such variants as the various passive and relative clauses into a common form.

3.3 Generalization and Concept Classes

Because tuples may not repeat with sufficient frequency to obtain reliable statistics, each tuple is reduced to a set of pairs: e.g., a verb-object pair, a subject-object pair, etc. Each pair is used as a generalized pattern during the candidate selection stage. Once we have identified pairs which are relevant to the scenario, we use them to gather the set of words for the missing role(s) (for example, a class of verbs which occur with a relevant subject-object pair: "*company* {hire/fire/expel...} *person*").

3.4 Pattern Discovery

We conducted experiments in several scenarios within news domains such as changes in corporate ownership, and natural disasters. Here we present results on the "Management Succession" and "Mergers/Acquisitions" scenarios. ExDisco was seeded with minimal pattern sets, namely:

Subject	*Verb*	*Direct Object*
C-Company	C-Appoint	C-Person
C-Person	C-Resign	—

for the Management task, and

Subject	*Verb*	*Direct Object*
*	C-Buy	C-Company
C-Company	merge	*

for Acquisitions. Here *C-Company* and *C-Person* denote semantic classes containing named entities of the corresponding types. *C-Appoint* denotes the list of verbs { *appoint, elect, promote, name, nominate*}, *C-Resign* = { *resign, depart, quit* }, and *C-Buy* = { *buy* , *purchase* }.

During a single iteration, we compute the score, *Score*(p), for each candidate pattern p, using the formula[1]:

$$Score(p) = \frac{|H \cap R|}{|H|} \cdot \log |H \cap R| \qquad (1)$$

where R denotes the relevant subset of documents, and $H = H(p)$ the documents matching p, as above; the first term accounts for the conditional probability of relevance of p, and the second for its support. We further impose two *support* criteria: we distrust such frequent patterns where $|H \cap U| > \alpha|U|$, as uninformative, and rare patterns for which $|H \cap R| < \beta$ as noise.[2] At the end of each iteration, the system selects the pattern with the highest *Score*(p), and adds it to the seed set. The documents which the winning pattern hits are added to the relevant set. The pattern search is then restarted.

3.5 Document Re-ranking

The above is a simplification of the actual procedure, in several respects.

Only generalized patterns are considered for candidacy, with one or more slots filled with wild-cards. In computing the score of the generalized pattern, we do not take into consideration *all* possible values of the wild-card role. We instead constrain the wild-card to those values which themselves in turn have high scores. These values then become members of a new class, which is produced in tandem with the winning pattern.

Documents relevance is scored on a scale between 0 and 1. The seed patterns are accepted as truth; the documents they match have relevance 1. On iteration $i + 1$, each pattern p is assigned a precision measure, based on the relevance of the documents it matches:

$$Prec^{i+1}(p) = \frac{1}{|H(p)|} \cdot \sum_{d \in H(p)} Rel^i(d) \qquad (2)$$

where $Rel^i(d)$ is the relevance of the document from the previous iteration, and $H(p)$ is the set of documents where p matched. In general, if K is a classifier consisting of a *set* of patterns, we define $H(K)$ as the set of documents where *all* of patterns $p \in K$ match, and the "cumulative" precision of K as

$$Prec^{i+1}(K) = \frac{1}{|H(K)|} \cdot \sum_{d \in H(K)} Rel^i(d) \qquad (3)$$

Once the winning pattern is accepted, the relevance of the documents is re-adjusted. For each document d which is matched by some subset of the currently accepted patterns, we can view that subset of patterns as a classifier $K_d = \{p_j\}$. These patterns determine the new relevance score of the document as

$$Rel^{i+1}(d) = \max\left(Rel^i(d), Prec^{i+1}(K_d)\right) \qquad (4)$$

This ensures that the relevance score grows monotonically, and only when there is sufficient positive evidence, as the patterns in effect vote "conjunctively" on the documents.

We also tried an alternative, "disjunctive" voting scheme, with weights which accounts for variation in *support* of the patterns,

$$Rel^i(d) = 1 - \sqrt[\bar{w}]{\prod_{p \in K(d)} (1 - Prec^i(p))^{w_p}} \qquad (5)$$

where the weights w_p are defined using the relevance of the documents, as the total support which the pattern p receives:

$$w_p = \log \sum_{d \in H(p)} Rel(d)$$

and $\bar{w}$ is the largest weight. The recursive formulas capture the mutual dependency of patterns and documents; this re-computation and growing of precision and relevance ranks is the core of the procedure.[3]

4 Results

4.1 Event Extraction

The most natural measure of effectiveness of our discovery procedure is the performance of an extraction system using the discovered patterns. However, it is not possible to apply this metric directly because the discovered patterns lack some of the information required for entries in

[1]similar to that used in (Riloff, 1996)

[2]We used $\alpha = 0.1$ and $\beta = 2$.

[3]We did not observe a significant difference in performance between the two formulas 4 and 5 in our experiments; the results which follow use 5.

the pattern base: information about the event type (predicate) associated with the pattern, and the mapping from pattern elements to predicate arguments. We have evaluated ExDisco by *manually* incorporating the discovered patterns into the Proteus knowledge bases and running a full MUC-style evaluation.

We started with our extraction system, *Proteus*, which was used in MUC-6 in 1995, and has undergone continual improvements since the MUC evaluation. We removed all the scenario-specific clause and nominalization patterns.[4] We then reviewed all the patterns which were generated by the ExDisco, deleting those which were not relevant to the task, or which did not correspond directly to a predicate already implemented for this task.[5] The remaining patterns were augmented with information about the corresponding predicate, and the relation between the pattern and the predicate arguments.[6] The resulting variants of Proteus were applied to the formal training corpus and the (hidden) formal test corpus for MUC-6, and the output evaluated with the MUC scorer.

The results on the training corpus are:

Pattern Base	*Recall*	*Precision*	*F*
Seed	38	83	52.60
ExDisco	62	80	69.94
Union	69	79	73.50
Manual-MUC	54	71	61.93
Manual-NOW	69	79	73.91

and on the test corpus:

Pattern Base	*Recall*	*Precision*	*F*
Seed	27	74	39.58
ExDisco	52	72	60.16
Union	57	73	63.56
Manual-MUC	47	70	56.40
Manual-NOW	56	75	64.04

The tables show the recall and precision measures for the patterns, with F-measure being the harmonic mean of the two. The *Seed* pattern base consists of just the initial pattern set, given in the table on the previous page. To this we added the patterns which the system discovered automatically after about 100 iterations, producing the pattern set called ExDisco. For comparison, *Manual-MUC* is the pattern base manually developed on the MUC-6 training corpus—prepared over the course of 1 month of full-time work by at least one computational linguist (during which the 100-document training corpus was studied in detail). The last row, *Manual-now*, shows the current performance of the Proteus system. The base called *Union* contains the union of ExDisco and *Manual-Now*.

We find these results very encouraging: Proteus performs better with the patterns discovered by ExDisco than it did after one month of manual tuning and development; in fact, this performance is close to current levels, which are the result of substantial additional development. These results must be interpreted, however, with several caveats. First, Proteus performance depends on many factors besides the event patterns, such as the quality of name recognition, syntactic analysis, anaphora resolution, inferencing, etc. Several of these were improved since the MUC formal evaluation, so some of the gain over the MUC formal evaluation score is attributable to these factors. However, all of the other scores are comparable in these regards. Second, as we noted above, the patterns were reviewed and augmented manually, so the overall procedure is not entirely automatic. However, the review and augmentation process took little time, as compared to the manual corpus analysis and development of the pattern base.

4.2 Text filtering

We can obtain a second measure of performance by noting that, in addition to growing the pattern set, ExDisco also grows the rele-

[4]There are also a few *noun phrase* patterns which can give rise to scenario events. For example, "Mr Smith, former president of IBM", may produce an event record where Fred Smith left IBM. These patterns were left in Proteus for all the runs, and they make some contribution to the relatively high baseline scores obtained using just the seed event patterns.

[5]ExDisco found patterns which were relevant to the task but could not be easily accomodated in Proteus. For instance "X remained as president" could be relevant, particularly in the case of a merger creating a new corporate entity, but Proteus was not equipped to handle such information, and has not yet been extended to incorporate such patterns.

[6]As with all clause-level patterns in Proteus, these patterns are automatically generalized to handle syntactic variants such as passive, relative clause, etc.

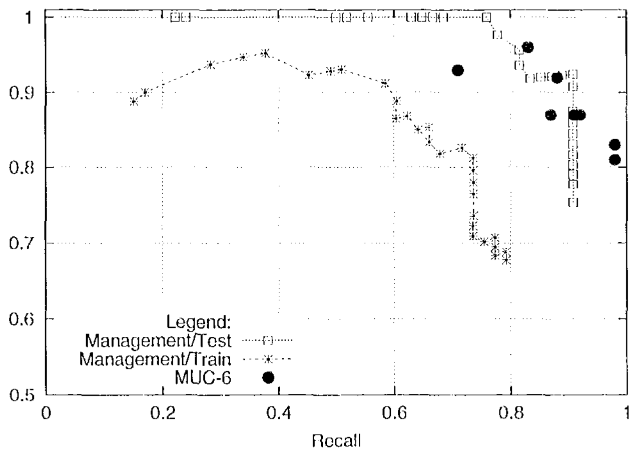

Figure 1: Management Succession

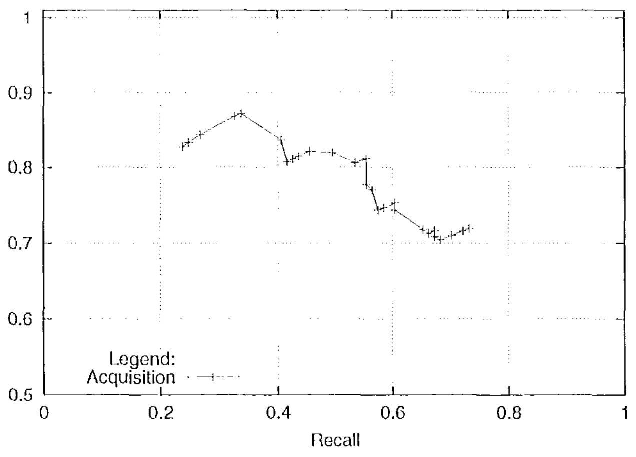

Figure 2: Mergers/Acquisitions

vance rankings of documents. The latter can be evaluated directly, without human intervention. We tested ExDisco against two corpora: the 100 documents from MUC-6 formal training, and the 100 documents from the MUC-6 formal test (both are contained among the 10,000 ExDisco training set)[7]. Figure 1 shows recall plotted against precision on the two corpora, over 100 iterations, starting with the seed patterns in section 3.4. This view on the discovery procedure is closely related to the MUC "text-filtering" task, in which the systems are judged at the level of *documents* rather than event slots. It is interesting to compare ExDisco's results with how other MUC-6 participants performed on the *MUC-6 test* corpus, shown anonymously. ExDisco attains values within the range of the MUC participants, all of which were either heavily-supervised or manually coded systems. It is important to bear in mind that ExDisco had no benefit of training material, or any information beyond the seed pattern set.

Figure 2 shows the performance of text filtering on the Acquisition task, again, given the seed in section 3.4. ExDisco was trained on the same WSJ corpus, and tested against a set of 200 documents. We retrieved this set using keyword-based IR search, and judged their relevance by hand.

[7]These judgements constituted the truth which was used only for evaluation, *not* visible to ExDisco

5 Discussion

The development of a variety of information extraction systems over the last decade has demonstrated their feasibility but also the limitations on their portability and performance. Preparing good patterns for these systems requires considerable skill, and achieving good coverage requires the analysis of a large amount of text. These problems have been impediments to the wider use of extraction systems.

These difficulties have stimulated research on pattern acquisition. Some of this work has emphasized interactive tools to convert examples to extraction patterns (Yangarber and Grishman, 1997); much of the research has focused on methods for automatically converting a corpus annotated with extraction examples into patterns (Lehnert et al., 1992; Fisher et al., 1995; Miller et al., 1998). These techniques may reduce the level of system expertise required to develop a new extraction application, but they do not lessen the burden of studying a large corpus in order to *find* relevant candidates.

The prior work most closely related to our own is that of (Riloff, 1996), who also seeks to build patterns automatically without the need to annotate a corpus with the information to be extracted. However, her work differs from our own in several important respects. First, her patterns identify phrases that fill individual slots in the template, without specifying how these slots may be combined at a later stage into complete templates. In contrast, our procedure discovers complete, multi-slot event patterns. Second, her procedure relies on a corpus in which the documents have been classified for relevance by hand (it was applied to the MUC-3 task, for which over 1500 classified documents are available), whereas ExDisco requires no manual relevance judgements. While classifying documents for relevance is much easier than annotating documents with the information to be extracted, it is still a significant task, and places a limit on the size of the training corpus that can be effectively used.

Our research has demonstrated that for the studied scenarios automatic pattern discovery can yield extraction performance comparable to that obtained through extensive corpus analysis. There are many directions in which the work reported here needs to be extended:

- using larger training corpora, in order to find less frequent examples, and in that way hopefully exceeding the performance of our best hand-trained system
- capturing the word classes which are generated as a by-product of our pattern discovery procedure (in a manner similar to (Riloff and Jones, 1999)) and using them to discover less frequent patterns in subsequent iterations
- evaluating the effectiveness of the discovery procedure on other scenarios. In particular, we need to be able to identify topics which can be most effectively characterized by clause-level patterns (as was the case for the business domain), and topics which can be better characterized by other means. We would also like to understand how the topic clusters (of documents and patterns) which are developed by our procedure line up with pre-specified scenarios.

References

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November. Morgan Kaufmann.

Ralph Grishman. 1995. The NYU system for MUC-6, or where's the syntax? In *Proc. Sixth Message Understanding Conf. (MUC-6)*, pages 167–176, Columbia, MD, November. Morgan Kaufmann.

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. 1992. University of massachusetts: MUC-4 test results and analysis. In *Proc. Fourth Message Understanding Conf.*, McLean, VA, June. Morgan Kaufmann.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and the Annotation Group. 1998. Algorithms that learn to extract information; BBN: Description of the SIFT system as used for MUC-7. In *Proc. 7th Message Understanding Conf.*, Fairfax, VA.

1993. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, MD, August. Morgan Kaufmann.

1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November. Morgan Kaufmann.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. 16th Nat'l Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proc. 13th Nat'l Conf. on Artificial Intelligence (AAAI-96)*. The AAAI Press/MIT Press.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proc. 5th Conf. on Applied Natural Language Processing*, pages 64–71, Washington, D.C. ACL.

Roman Yangarber and Ralph Grishman. 1997. Customization of information extraction systems. In Paola Velardi, editor, *Int'l Workshop on Lexically Driven Information Extraction*, Frascati, Italy. Università di Roma.

Roman Yangarber and Ralph Grishman. 1998. NYU: Description of the Proteus/PET system as used for MUC-7 ST. In *7th Message Understanding Conference*, Columbia, MD.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proc. Conf. on Applied Natural Language Processing (ANLP-NAACL)*, Seattle, WA.