Using ILP to Construct Features for Information Extraction from Semi-Structured Text

Ganesh Ramakrishnan[1], Sachindra Joshi[1], Sreeram Balakrishnan[1], and Ashwin Srinivasan[12]

[1] IBM India Research Laboratory, Block 1, Indian Institute of Technology, New Delhi 110016, India
{ganramkr, jsachind, srbalakr, ashwin.srinivasan}@in.ibm.com
[2] Dept. of CSE Centre for Health Informatics, University of New Kensington, Sydney, Australia

Abstract. Machine-generated documents containing semi-structured text are rapidly forming the bulk of data being stored in an organisation. Given a feature-based representation of such data, methods like SVMs are able to construct good models for information extraction (IE). But how are the feature-definitions to be obtained in the first place? (We are referring here to the representation problem: selecting good features from the ones defined comes later.) So far, features have been defined manually or by using special-purpose programs: neither approach scaling well to handle the heterogeneity of the data or new domain-specific information. We suggest that Inductive Logic Programming (ILP) could assist in this. Specifically, we demonstrate the use of ILP to define features for seven IE tasks using two disparate sources of information. Our findings are as follows: (1) the ILP system is able to identify efficiently large numbers of good features. Typically, the time taken to identify the features is comparable to the time taken to construct the predictive model; and (2) SVM models constructed with these ILP-features are better than the best reported to date that rely heavily on hand-crafted features. For the ILP practioneer, we also present evidence supporting the claim that, for IE tasks, using an ILP system to assist in constructing an extensional representation of text data (in the form of features and their values) is better than using it to construct intensional models for the tasks (in the form of rules for information extraction).

1 Introduction

The amount of text data available in a machine-readable format is already very large: Google alone indexes more than 10 billion pages, most of which are text. This is only expected to increase, as organisations increasingly employ machines capable of generating semi-structured (XML-like) text data (for example, projections by IBM in that corporation's Global Technology Outlook for 2003 suggests that by 2010, nearly 75% of the data stored in an organisation may of this type). This trend is accompanied by a substantial industrial impetus to develop automated methods for extracting information of potential commercial interest from such data. Information Extraction (IE), normally studied under the umbrella of "text mining" ([10, 6]), involves a number of tasks like: sentence segmentation [22], part of speech tagging [5], noun-phrase coreferencing [24] and named entity annotation [3]. The principal goal of IE is to

extract structured information from unstructured text documents. This structured information is normally in a form that can be stored in a database: although modern relational database implementations allow the direct storage and limited querying of XML-like data, a representation using attributes (features in the machine-learning sense) remains the most popular. This makes the data amenable to not just efficient querying, but to a variety of parametric and non-parametric techniques for mining and modelling (like association-rule mining [1], naive Bayes [17], hidden Markov models [10], maximum entropy models [3], support vector machine [4] and conditional random fields [13]). These methods construct models using some subset of the features identified to represent the text.

In this paper, we are concerned with the question of how an appropriate feature representation is to be arrived at the first place. This is a step before feature-subset selection: there, once a set of features have been defined, a subset of them is sought—usually, but not always, with a view of building a good predictive model. However, obtaining definitions of the features *ab initio* is a more complex business. The "feature engineer" has to construct these from some combination of general-purpose and problem-specific information, and, if available, knowledge of how the features would be used (for example, to build a model that can discriminate accurately amongst interesting and uninteresting corporate mergers). This has meant that the task of defining the features has been one that has largely been manual, or achieved through problem-specific programs that use the knowledge sources in a pre-defined manner. It is difficult to see how these approaches could scale-up to meet the demands imposed by the scale and heterogeneity of text-based data that are being generated, or to incorporate new sources of information that become electronically available. Our interests are therefore in automated methods that can assist in this by automatic identification of interesting features. We adopt the following positions: (1) Formal logic, at least with the power embodied within logic programming languages, is adequate for representing the different kinds of information that are needed for IE; and (2) Any automatic method for identifying feature definitions that uses a logical representation has to be at least at the level of first-order logic.

We will ask the reader to take (1) as axiomatic for this paper. Some justification for (2) on the other hand, follows from the observation that feature definitions in logic are essentially functions of the form $f : X \mapsto Y$, from the set of individuals X to some more or less arbitrary Y (Y could be the Booleans, for example). Thus, if these definitions are to be identified automatically, then the underlying program has to be able to construct functions. This implies any program for feature engineering must at least be in a position to construct definitions in first-order logic (or higher, if composition of functions are needed to construct complex features; or if the representation of individuals could in turn employ functions as in [16]).

Arguably the most well-developed and general-purpose programs for constructing first-order definitions have been in the area of Inductive Logic Programming (ILP). ILP programs are normally, but not exclusively, used to learn rules (called theories in the literature) in first-order logic to classify examples. The rules employ predicates provided as background knowledge encoded in some subset of first-order logic. In this paper, we propose that ILP could be employed to attain the IE goal of converting unstructured text data to a structured form: the process we envisage is summarised in Fig. 1.

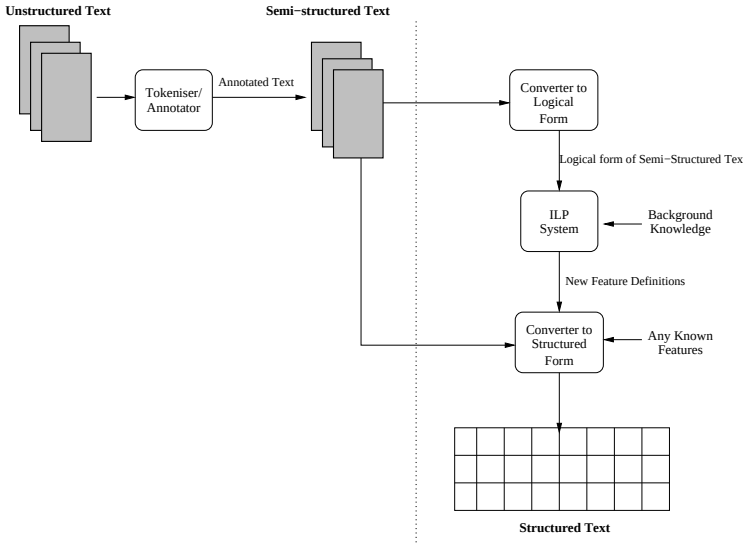

Fig. 1. A simplified view of a role for ILP in information extraction. The focus of this paper is on programs, inputs, and outputs to the right of the dotted line.

In the past, several authors have used ILP, or ILP-inspired systems for information extraction. Notable amongst these are: the work of Aitken [2] who uses ILP to construct theories for IE; Califf's work with with Rapier [7], which is inspired by bottom-up ILP systems; and the work of Roth and colleagues [23] who use restricted templates defined by "relation generating functions" to construct features for IE (their motivation for this is that general-purpose ILP methods are inflexible, making their use impossible in NLP-like domains). Our results here are intended to add these by providing evidence for the following: (1) a general-purpose ILP system can efficiently extract useful features for information extraction; and (2) feature extraction is a more effective way to use a general-purpose ILP system than the usual process of extracting rules.

Although the use of ILP as a general-purpose mechanism for identifing feature definitions has a long history (a process termed "propositionalisation": see [12] for a detailed survey), and has been shown to be one of the most effective ways to use ILP to address difficult problems (see for example: [11]), there has not been, to the best of our knowledge, any attempt to use them for this purpose in IE. We believe that there may be two reasons for their neglect in IE. First, ILP has been perceived as being too inefficient to identify a suitably large set of features that may be needed to represent documents adequately (for example, [23] use this as their primary motivation to develop the restricted approach) . Second, it is not apparent that an automatic feature construction method could match the kinds of performance achievable by good hand-crafted features. In this paper, it is our intention to demonstrate using an established test-bed that both these concern may be unfounded. Specifically, we examine a task in information extraction concerned with the problem of extracting instances of a structured target schema from an unstructured text data. The data we use concerns corporate mergers and acquisitions, from which we intend to extract the

names of the acquiring and acquired companies; and the deal amount (the actual task, described in Section 3.2 actually involves identifying seven different entities). For this task, we use a well-known ILP system to identify efficiently a large number—from a few thousand to ten thousand— good features using two quite different sources of information (Wordnet [18] and a dependancy parser). The features identified are used by a standard support vector machine (SVM) classifier to construct predictive models for the entities of interest. These models—a special kind of the type proposed by [20]—are compared against the best reported in the literature (these rely largely on hand-crafted features).

The rest of the paper is organised as follows. Section 2 describes the aspects of ILP to the extent that it relates to the feature-identification task undertaken here. Section 3 describes the empirical investigation undertaken in the paper. This includes a description of aims (Section 3.1), materials (Section 3.2), methods (Section 3.3) and results (Section 3.4). Section 4 concludes the paper.

2 Feature Definitions using Inductive Logic Programming

Given problem-specific data and general-purpose ("background") knowledge encoded in some logical form—normally a subset of first-order logic—an ILP system attempts to construct models, also in a logical form, for the data. Implementations have been dominated by two classes of programs, corresponding somewhat to the broader division into supervised and unsupervised learning. The first class—predictive ILP—is concerned with constructing "theories" (sets of rules; or first-order logic variants of classification or regression trees) for discriminating accurately amongst two sets of examples ("positive" and "negative"). The second—descriptive ILP—has is concerned with identifying relationships that hold amongst the data and the background knowledge a view of discrimination. More details of the requirements of programs in these two categories can be found in [19]:

The task of finding the definition of features using a first-order logic representation is one that is not easily characterised as either predictive or descriptive ILP. Solutions conceptually involve two steps: (1) a feature-construction step that identifies (within computational reason) all the features that are consistent with the constraints provided by the background knowledge. This is characteristic of a descriptive ILP program; and (2) a feature-selection step that retains some of the features based on their utility estimated using the problem-specific data. This is characteristic of a predictive ILP program. A partial specification for an ILP program that reflects this combination has been recently proposed in [26], which we follow here (we refer the reader to [21] for definitions of the logical terms used below):

- B (background knowledge) consists of a finite, possibly empty, set of clauses $= \{C_1, C_2, \ldots\}$
- E (data) consists of a finite set $E^+ \cup E^-$ where:
  - *Positive Examples*. $E^+ = \{e_1, e_2, \ldots\}$ is a non-empty set of definite clauses;
  - *Negative Examples*. $E^- = \{\overline{f_1}, \overline{f_2} \ldots\}$ is a set of Horn clauses (this may be empty)
- $\mathcal{H}$ is the set of definite clauses, constructable with predicates, functions and constants in $B \cup E$; $\mathcal{F}$ the set of features constructable using a set of individuals; and $\tau : \mathcal{H} \mapsto \mathcal{F}$ a function that maps a definite clause $h \in \mathcal{H}$ to a feature $f \in \mathcal{F}$.

– $F = \{f_1, f_2, \ldots\} \in \mathcal{F}$, the output of the algorithm given B and E is acceptable for any set $H = \{h_1, h_2, \ldots\} \in \mathcal{H}$ if the following conditions are met:
  - *Posterior Sufficiency.* $B \cup \{h_i\} \models e_1 \vee e_2 \vee \ldots$, where $\{e_1, e_2, \ldots\} \subseteq E^+$
  - $f_i = \tau(h_i)$

The reader would have noted that Posterior Sufficiency requires features constructed here use clauses that entail, with B, at least one positive example: this is not necessarily satisfied by programs that simply seek to identify all features constructable with predicates, functions and constants in $B \cup E$. The features returned by such programs will be a superset of the features F. Here, we seek instead to constrain the set F further to return a subset F' of F that accounts for a notion of utility using some predicate *Good* (effectively, being the same notion of "interestingness" used in the data-mining literature):

– *Utility.* $F' = \{f : f \in F,\ \text{and } \mathit{Good}(f, B, E) = TRUE\}$

For the rest, we follow [26] and refer the reader to that paper for details of $\mathcal{F}$, $\mathcal{H}$ and τ. We view the definition of *Good* as problem-specific, and details for the empirical study here are provided in Section 3.3. A program that satisfies these requirements constructs the definition of a feature in the following manner. First, a set of clauses H is identified using the examples and background knowledge. Each clause is of the form $\mathit{head} \leftarrow \mathit{body}$, where *head* is a literal and *body* a conjunction of literals; and entails at least one positive example, given the background knowledge B. Next, each clause h_i in H is converted into a boolean feature f_i that takes the value 1 (or 0) for any individual for which the body of the clause true (if the body is false).[3] Thus, the set of clauses H gives rise to a boolean vector for each individual in the set of examples. An example in the context of information extraction is shown in Fig. 2, in which the task is to assign "roles" to individuals. The problem concerns corporate mergers and acquisitions, and individuals are assigned roles like "purchaser", "purchased", "deal amount" and so on. In the example in Fig. 2, individuals are identified by the triple $\langle d, s, l \rangle$, where d denotes a document, s a sentence in d, and l the location of a segment in d.

3 Experimental Evaluation

3.1 Aims

We intend to investigate the use of an ILP system for automatic feature construction in information extraction. Specifically, using a benchmark dataset, our primary goals are to examine: (a) Whether the ILP system is able to identify features efficiently using multiple sources of background knowledge;[4] and (b) Whether the features identified by an ILP system are "good", measured by comparing the predictive models that are constructed using the features—we will call the approach "ILP-assisted"—against the best reported models (these use a combination of hand-crafted and simple automatically constructed features).

[3] The body forms the definition of a "context predicate" in the terminology of [22]

[4] In principle, it is evident that an ILP capable of using one source of background information should be able to use multiple sources as well. However, the sub-optimal nature of most implementations has meant that this is not necessarily the case in practice.

Clause:

$$\forall d, s, l(Has\_role(d, s, l, purchaser) \leftarrow \\ Has\_annotation(d, s, l, \text{organisation}) \wedge \\ After(l, l1) \wedge \\ Has\_hyp\_sense(d, s, l1, \text{02incrs0incrm0}))$$

Feature:

$$f(d, s, l) = \begin{cases} Has\_annotation(d, s, l, \text{organisation}) \wedge \\ 1 \; After(l, l1) \wedge \\ Has\_hyp\_sense(d, s, l1, \text{02incrs0incrm0}) \\ \quad = TRUE \\ \\ 0 \text{ otherwise} \end{cases}$$

Fig. 2. Example of a boolean feature constructed from a clause. The clause assigns the role 'purchaser' to any individual (denoted by specific values assigned to variables d, s and l) that satisfies the conditions in the body of the clause. The meanings of the predicate symbols $Has\_annotation$, $After$, and Has_hyp_sense are explained in Section 3.

3.2 Materials

Data We use data contained in the "Corporate Acquisition Events" corpus described in [14]. This is a collection of 600 news articles describing acquisition events taken from the Reuters dataset. News articles are tagged to identify fields related to acquisition events. These fields include 'purchaser' , 'acquired', and 'seller' companies along with their abbreviated names ('purchabr', 'acqabr' and 'sellerabr') Some news articles also mention the field 'deal amount'. Together, these seven fields define the set of target elements for information extraction task: we will refer to these fields as "roles" in the rest of the document. In Table 1, we summarize this information.

Role	Number of Examples
acquired	651
acqabr	1494
purchaser	594
purchabr	1347
seller	707
sellerabr	458
deal amount	206
Total	5457

Table 1. Examples in the Corporate Acquisitions Events corpus.

Each unstructured text document is first converted to a semi-structured form using a tokeniser, followed by an annotator. The output of each of these are then converted automatically into a logical form, which is then part of the data provided to the ILP system. The details are as follows.

Each unstructured text document contains one or more sentences, each of which can have several tokens. Tokens are individual words, or groups of words which have a

unique identifier within a sentence in a document. Tokens are then tagged using a high-recall named entity annotator. We use a rule-based named entity annotator developed in-house, that produces four different annotations: 'currency', 'date', 'location' and 'organisation'.

We convert the output of a tokeniser into a logical form that encodes the location of tokens. This is done using the predicate Has_token, resulting in facts of the form *Has_token (doc_id, sentence_id, token_id, t)*. This states that at there is a token t at location *token_id* within *sentence_id* in document *doc_id*. For the dataset here, there are approximately $70,000$ statements of this form. In a similar manner, the results of the annotator are encoded by facts of the form *Has_annotation(doc_id, sentence_id, token_id, a)*. There are approximately $10,000$ statements of this form.

Examples of roles identified in the text are encoded using the predicate Has_role. The result is a set of facts of the form *Has_role (doc_id, sentence_id, token_id, r)*. the role r. Corresponding to the entries in Table 1, there are 651 facts with role *acquired*, 1494 with *acqabr* and so on. The entire corpus used is thus represented in a logical form by approximately $85,000$ facts.

We further generate "negative examples" for tokens at a position (that is, at a location in a sentence within a document) in the following manner. First, roles not assigned to a token at a position are each taken to constitute a negative example for the token at that position. Second, a token at a position that has been annotated by the named entity annotator, but does not have a role assigned to it at that position is marked as having none of the possible seven roles at that position. This results in approximately in a further $71,000$ facts. The entire dataset is thus represented by about $156,000$ facts.

Background Knowledge Background knowledge for the ILP system consists of logical encodings of the following two sources of information:

Semantic Lexicon. Natural languages provide a rich set of expressions allowing several different ways of expressing the same fact. As an example, the expression 'company A purchased company B recently' and the expression 'company A recently acquired company B' mean the same thing. We use WordNet to help address some of this problem. WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called 'synsets' and records the various semantic relations between these synonym sets. We use the hypernym relations within Wordnet to compute the hypersense of tokens. This requires the following computation. First, the synset corresponding to a token is obtained. Hypernyms of this synset is a hyper sense for the token. Further hyper senses can obtained recursively from hypernyms of the synsets, their hypernyms and so on. We restrict ourselves to two levels of such recursion. The result is encoded using predicates of the form: *Has_hyp_sense(doc_id, sentence_id, token_id, s)*. While any modern ILP system can compute these facts "on-the-fly", we pre-compute them for efficiency. This results in approximately $531,000$ facts.

Dependency Parser. We use MINIPAR [15] to obtain dependency relationships in a sentence. A dependency relationship is an asymmetric relationship between a token called *head* or parent and another token called *modifier* or dependent. A token in a sentence may have several modifiers, however, it can modify only a single word. The root of a dependency tree does not modify any word and is

called the head of the sentence. A MINIPAR 'relationship' is a label assigned to a dependency relationship between a pair of tokens in a sentence. Some of the examples relationships used in MINIPAR are 'subj' (subject), 'adjn' (adjunct), 'cmpl' (complement), and 'obj' (object). We generate a set of facts of the following form using the output generated by MINIPAR: *Links(doc_id, sentence_id, link_type, token_id_1, token_id_2)*. The argument *Link_type* can take values from the set of relations used by MINIPAR. The arguments *token_id_1* and *token_id_2* correspond to the modifier and the header word respectively. Again, all these facts could be computed on-th-fly, but are precomputed here. There are approximately $76,000$ facts of this form.

In addition, we also provide utility predicates *After* and *Before* that allow the ILP program to access locations in the neighbourhood of any given location in a sentence. The background information is thus represented by about $600,000$ facts.

3.3 Method

Our method for estimating the performance of ILP-assisted approach is straightforward:

Repeat N times:

1. Randomly split the data into training (Tr) and test (Te) data;
2. Obtain a set of no more than k features using the ILP program using background knowledge B;
3. Construct a classificatory model for predicting the roles in Fig. 1 using a standard classification technique equipped wth the features identified in Step 2 and the data in Tr;
4. Record the performance of the model obtained in Step 3 on the data in Te;

Estimate the predictive performance of ILP-assisted approach by the mean values of accuracies obtained in Step 4.

Compare the performance of the models with ILP-assisted approach against performance of the best models reported in the literature.

The following details are relevant:

(a) We take N to be 5 for our experiments;
(b) k is restricted to $20,000$ for our experiments;
(c) We use the ILP system Aleph (Version 5) [27] for all experiments. This program has a procedure that constructs features using a non-greedy set covering approach (we refer the reader to the Aleph manual for details). The procedure ensures that features are constructed from clauses that satisfy the Posterior Sufficiency requirement in Section 2. The classificatory model in Step 3 is obtained using a linear SVM: the specific implementation used is the one provided in the WEKA toolbox called SMO.[5] It can be shown that the resulting models are a special case of the SVILP models proposed in [20]. We compare the performance of this model against those constructed by SRV [9], HMM [10] and Elie [8].

[5] `http://www.cs.waikato.ac.nz/ ml/weka/`

(d) Satisfying the Utility requirement requires the definition of the predicate *Good*. Here, *Good* is $TRUE$ for clauses (and, by implication, features found by the ILP system) that entail at least 5 positive examples and have a precision of at least 0.6. These numbers are arbitrary, but do not seem to affect the results greatly; The implementation we use is also able to ensure that this utility requirement is met during its search for features: this is more efficient than first finding all features and then removing those that fail to meet the criterion of minimum utility. All other settings for Aleph are at their default values for feature construction.

(e) Although the utility predicates $Before$ and $After$ allow access to any number of locations within a sentence, for experiments here we restrict the neighbourhood to no more than 5 locations on either side of a token.

(f) Performance will be measured using the following standard statistics: precision (P), recall (R), and F1. These are defined as follows: $P = TP/PredPos$, where TP is the true positives and $PredPos$ are the numbers of examples predicted as positive; $R = TP/ActPos$, where $ActPos$ are the numbers of examples that are actually positive; and $F1 = 2PR/(P + R)$.

(g) A quantitative comparison of the performance of the ILP-assisted approach are only possible against Elie (predictions for SRV and HMM are only available for 3 of the 7 roles, which makes any quantitative statement unreliable). For this, we use the Wilcoxon signed-rank test [25]. The test is a non-parametric test of the null hypothesis that there is no significant difference between the median F1 performance of a pair of algorithms.

3.4 Results

Figure 3 shows the performance of the classificatory model obtained using ILP-assisted approach; and Fig. 4 shows the comparative performance of this against the best reports available. On balance, the signed-rank test suggests that there is some evidence in favour of the ILP-assisted approach over Elie, although the difference is not significant (the sum of signed ranks is 14: the critical value for a significant difference is 22). Numbers are too small to make a reliable statement about other comparisons: the odds seem to favour the ILP-assisted approach over either SRV or HMM. These results suggest that the ILP-assisted approach is at least as good as the others tabulated.

Role	**Performance**		
	P	R	F1
acquired	51.7	35.2	41.8
acqabr	46.0	39.8	42.6
purchaser	54.7	39.0	45.4
purchabr	42.3	30.8	35.4
seller	52.2	52.1	51.5
sellerabr	25.4	19.5	21.7
dlramt	60.9	47.3	53.0

Fig. 3. Estimates of performance of the ILP-assisted classifier. P, R denote precision and recall. F1 is the harmonic mean of P and R.

Comparative assessments of this nature, while valuable, are unrepresentative of how the ILP-assisted approach is intended to be used in practice. As shown in Fig. 1, we envisage the features identified by the ILP algorithm to augment any existing

features. It is therefore of more interest to see how a methods like Elie perform when provided additionally, with the ILP features; or conversely, providing SMO (the SVM implementation used here to build models with the ILP features) additionally with the features used by methods like Elie, HMM and SRV. Unfortunately, we are not able to perform either of these experiments: executable implementations of Elie, SRV and HMM are not available, and we are unable to re-construct the definitions of their features from the descriptions provided in the literature. A surrogate experiment is however possible, that gives some insight into what may be achievable in practice. We are able to examine the construction of features by the ILP system without the semantic lexicon and the dependency parser. This amounts to the ILP system only having access to the tokeniser, annotator, and the utility predicates $Before$ and $After$. In such circumstances, the features constructed will be regular expressions, referring to the locations of tokens, annotations and gaps. Taking this as a kind of baseline for either hand-crafted or features that can be constructed (this does not need a general-purpose ILP system), we are in a position to examine the change in performance effected by the ILP system as it identifies features by incorporating background information. From Fig. 5, it is evident that models that use ILP features constructed with additional background knowledge do perform better than the baseline. This suggests that the ability of the ILP system to use background information does translate into finding features that can improve predictive performance. If the results in Fig. 5 are representative, then augmenting the baseline features with ILP features that use all the background information yields improvements in the F1 value of 5.5. This can be taken as some indication of the improvements that may be achievable by augmenting SMO with the features used by Elie, SRV or HMM.

Role	SRV	HMM	Elie	ILP-assisted
acquired	34.3	30.9	42.0	41.8
acqabr	35.1	40.1	40.0	**42.6**
purchaser	42.9	**48.1**	47.0	45.4
purchabr	−−	−−	29.0	**35.4**
seller	−−	−−	15.0	**51.5**
sellerabr	−−	−−	14.0	**21.7**
dlramt	−−	−−	**59.0**	53.0

Fig. 4. Comparative performance of the ILP-asssisted approach. Results for three roles are not reported by SRV and HMM.

Role	**B**	**B+L**	**B+P**	**B+L+P**
acquired	34.5	40.2	41.5	41.8
acqabr	34.5	39.8	42.6	42.6
purchaser	42.5	42.8	45.2	45.4
purchabr	25.0	30.7	29.4	35.4
seller	44.3	48.8	50.6	51.5
sellerabr	19.8	23.2	19.3	21.7
dlramt	49.2	50.3	47.1	53.0

Fig. 5. Comparative F1 values for models constructed by SMO. "B" refers to models constructed using baseline features that are regular expressions on tokens and annotations. "L" refers to semantic lexicon features that use information provided by Wordnet, and "P" refers to features that use information provided by the dependency parser MINIPAR.

It is also of some interest to consider for the specific IE tasks considered here whether it is better to use the ILP-assisted approach proposed; or to use ILP to construct models that directly predict the roles of individuals in documents (we will call this "ILP-models"). Evidence that we have suggests the former is better: see Fig. 6.

We have not commented so far on the efficiency of constructing features using ILP. Our estimates suggest that in all cases, the time for feature construction is comparable

Method	Performance
ILP-assisted	39.8 ±0.9
ILP-models	35.2 ±1.5

Fig. 6. Average F1 values for models using the ILP-assisted approach and ILP models that predict roles directly. The averages are weighted averages that account for the proportions of examples for each role.

to the time taken for model construction with the SVM. Some caveats are needed. Model construction with an SVM is faster than feature construction (but no more than about 5 times) when: (a) the number of features are small (say about 2000 or so); and (b) the number of classes are small. As either of these are increased, model construction time was observed to be significantly greater than the time taken to construct features.

4 Concluding Remarks

In this paper, we have investigated the use of ILP to assist in the task of identify features for converting text data into a structured form. Our results suggest that an ILP system can effectively amalgamate information from disparate sources to identify features that can then be used to build good predictive models for specific IE tasks. The promise of adopting this route rests on two points: (1) ILP provides a general-purpose setting for feature-construction that uses a substantially rich subset of first-order logic for representation, and makes explicit provision for incorporating domain-specific and general-purpose background information; and (2) features constructed in this manner augment, not replace, those already known to be effective. In principle, the output of any good model builder should not get any worse.

While the experimental results are sufficient to demonstrate the feasibility and the value obtained using an ILP-assisted approach, there are three ways in which they could be extended immediately. First, similar positive results on other IE tasks would clearly establish the utility of the approach further. Second, we have used a very general-purpose ILP system in our experiments. ILP implementations that are specifically designed for feature-identification now exist (these do not lose any of the other generality of ILP). These would provide more efficient identification of features (and possibly even better ones). Third, we intend using other knowledge sources such as VerbNet in future work.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 12–15 1994.
2. J.S. Aitken. Learning Information Extraction Rules: An Inductive Logic Programming approach. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 355–359, 2002. http://citeseer.ist.psu.edu/586553.html.
3. A. Borthwick. *A maximum entropy approach to named entity recognition.* PhD thesis, 1999.
4. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

5. Thorsten Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, 2000.
6. Razvan Bunescu and Raymond J. Mooney. Relational markov networks for collective information extraction. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, 2004.
7. 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.
8. Aidan Finn and Nicholas Kushmerick. Multi-level boundary classification for information extraction. In *ECML*, 2004.
9. Dayne Freitag. Toward general-purpose learning for information extraction. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics*, 1998.
10. Dayne Freitag and Andrew Kachites McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction*, 1999.
11. R. D. King, A. Srinivasan, and L. DeHaspe. WARMR: A Data Mining Tool for Chemical Data. *Computer Aided Molecular Design*, 15:173–181, 2001.
12. Stefan Kramer, Nada Lavra;, and Peter Flach. *Propositionalization approaches to relational data mining*. Springer-Verlag New York, Inc., 2000.
13. John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
14. David Dolan Lewis. *Representation and learning in information retrieval*. PhD thesis, 1992.
15. D. Lin. Dependency-based evaluation of minipar. In *In Workshop on the Evaluation of Parsing Systems*, 1998.
16. J.W. Lloyd. Logic for learning: Learning comprehensible theories from structured data. In *Springer, Cognitive Technologies Series*, 2003.
17. A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
18. George Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11), 1995.
19. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
20. S.H. Muggleton, H. Lodhi, A. Amini, and M.J.E. Sternberg. Support Vector Inductive Logic Programming. In *Proceedings of the 8th International Conference on Discovery Science*, LNAI 3735, pages 163–175. Springer-Verlag, 2005.
21. S. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. Springer, Berlin, 1997.
22. Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *ANLP*, pages 16–19, 1997.
23. Dan Roth and Wen tau Yih. Relational learning via propositional algorithms: An information extraction case study. In *IJCAI*, pages 1257–1263, 2001.
24. Erik F. Tjong Kim Sang, Walter Daelemans, Hervé Déjean, Rob Koeling, Yuval Krymolowski, Vasin Punyakanok, and Dan Roth. Applying system combination to base noun phrase identification. In *COLING*, pages 857–863, 2000.
25. Sidney Siegel and N. John Castellan Jr. *Nonparametric Statistics for The Behavioral Sciences*. McGraw-Hill, 1956.
26. Lucia Specia, Srinivasan A., Ramakrishnan G., and Nunes M.G.V. Word sense disambiguation using ilp. In *16th International Conference on Inductive Logic Programming*, 2006.
27. A. Srinivasan. The Aleph Manual. Available at http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/, 1999.