Learning for Biomedical Information Extraction with ILP

Margherita Berardi[1], Vincenzo Giuliano[2], and Donato Malerba[1]

[12]Dipartimento di Informatica
Università degli Studi di Bari, via Orabona, 4 - 70126 Bari - Italy
[1]{berardi,malerba}@di.uniba.it
[2]enzoju@hotmail.com

Abstract. Information in text form remains a greatly unexploited source of biological information. Information Extraction (IE) techniques are necessary to map this information into structured representations that allow facts relating domain-relevant entities to be automatically recognized. In biomedical IE tasks, extracting patterns that model implicit relations among entities is particularly important since biological systems intrinsically involve interactions among several entities. In this paper, we resort to an Inductive Logic Programming (ILP) approach for the discovery of mutual recursive theories from text. Mutual recursion allows dependencies among entities to be explored in data and extraction models to be applied in a context-sensitive mode. In particular, IE models are discovered in form of classification rules encoding the conditions to fill a pre-defined information template. An application to a real-world dataset composed by publications selected to support biologists in the task of automatic annotation of a genomic database is reported.

1 Introduction

The last decade has witnessed an unexampled expansion of biomedical data and related literature. Advances of genome sequencing techniques have mainly risen an overwhelming increase in the literature discussing discovered genes, proteins and their role in biological processes. The ability to survey this literature and extract relevant portions of information is crucial for researchers in biomedicine. However, finding explicit entities (e.g., a protein or a kinase) and facts (e.g., phosporylation and interaction relationships) in unstructured text is a time consuming and boring task because of the size of available resources, data sparseness and continuous updating of information. Information Extraction (IE) is the process of mapping unstructured text into structured form, such as knowledge bases or databases, by filling predefined templates of information describing objects of interest and facts about them. This motivates the interest of IE and text mining practitioners towards the biomedical field [12, 19, 13].

In a machine learning perspective, IE can be tackled as a classification task, where classification models composed by rules or patterns encoding the conditions to fill a given slot of a template of interest are learned from a set of

annotated texts (i.e., examples of filled templates) [17]. Although natural language research has widely made use of statistical techniques (e.g., hidden Markov models and probabilistic context-free grammars) because of their robustness and wide coverage peculiarities, logic-based approaches are able to overcome some intrinsic defects of statistical approaches due to their inability to cope with the level of semantic interpretation and the linguistically impoverished nature of discovered models that are difficult to interpret and extend [16]. Indeed, logical approaches, such as those that are developed in the Inductive Logic Programming (ILP) framework, allow to naturally encode natural language statements representing both data and background knowledge and to learn or revise these logical representations [8]. Moreover, IE tasks can be naturally framed in the ILP relational setting where data have a relational structure and examples can be related each other. In the literature, there are several works that take advantages of ILP principles to learn rule-based models from logical representations of texts [1, 9, 14, 10, 4], whereas few attempts have been conducted to solve IE problems from biomedical texts [5, 11], despite the fact that it promises to become a major application area where ILP may converge [7]. Difficulties are due to the complexity of the biomedical language which is characterized by inconsistent naming conventions, that is, ambiguities occurring when the same term is used to denote more than one semantic class (e.g., p53 is used to specify both a gene and a protein) or when many terms lead to the same semantic class (abbreviations, acronym variations), continuous creation of new biological terms or evolutions of the same biological object (e.g., genes are renamed once their function is known), non standard grammatical structures as well as domain-specific jargon combined with English. This makes the data processing phase in the learning process really difficult. On the other hand, it is available a large amount of controlled vocabularies, lexicons and ontologies that can be exploited both in the data processing and reasoning steps. This further motivates the use of an ILP approach since it allows to naturally handle explicitly expressed background knowledge.

In biomedical IE tasks, extracting patterns that model implicit relations among entities is particularly important since biological systems intrinsically involve interactions among several entities, e.g., genes and proteins interacting in regulation networks. Implicit relations can be captured in form of mutual recursive patterns, that is, patterns relating more than one entity of interest. Mechanisms supporting mutual recursion exploration in the space of candidate patterns allow hidden dependencies among entities to be discovered in data and extraction models to be applied in a context-sensitive mode.

In this paper, IE models are discovered in form of classification rules including mutual recursive definitions of classes of interest, where each class plays the role of a template slot while recursive patterns are searched among slots of the same template. In an ILP perspective, classification is tackled as a concept learning problem in a multiple predicate learning framework. In the following section, we describe how the annotation of a genomic databases can be supported by properly defining a biomedical information extraction problem. In Section 3, representation and algorithmic issues faced in the ATRE system to discover con-

cept dependencies are briefly described. Data preparation techniques adopted to process data and the representation model used to describe data and background knowledge are reported in Section 4. Results obtained on a real-world dataset composed by publications concerning studies on mitochondrial pathologies are reported in Section 5. Finally, some conclusions are drawn in Section 6.

2 The Information Extraction problem

The application we are addressing concerns the annotation in the HmtDB resource [1] of variability and clinical data associated to mitochondrial pathological phenotypes [2]. Currently, HmtDB stores data from healthy subjects while variability and clinical data are manually extracted from published literature. A peculiarity of this fragment of the scientific literature is that biomedical documents are organized according to a regular section structure (composed by Abstract, Introduction, Methods, Results and Discussion) and that often biologists already know which part of the documents may contain a certain kind of information. This suggests to conduct the IE process in a local way to pre-categorized sections of interest [3]. Indeed, selecting relevant portions of text is a prerequisite step to IE as the lack of robustness and data sparseness make IE methods inapplicable to large corpora and irrelevant documents. In this application, selected publications concern mitochondrial mutations and biologists are interested in automating the identification of occurrences of specific biological objects (i.e., mitochondrial mutations) and their features (i.e., type, position, involved nucleotides, expressing locus, related pathology) as well as the particular method and experimental setting adopted in the scientific study (i.e., dimension, age, sex, nationality of the sample) discussed in the publication. Each of these information to be annotated can be considered as a single entity of interest, but more effective and expressive "extractors" might be mined when instances of relations among objects are modelled. Implicit relations among relating entities can be mined in data when observations include some relational knowledge, e.g., by simply describing word neighbourhood in text or when a domain-specific taxonomy is available, distances among textual objects can be described on the basis of the path in the hierarchy linking categories to which pairs of textual objects belong. More complex relations can be modelled by using ontologies of is_a or part_of relations that allow to recognize sentences containing specializations of concepts expressed by previous sentences.

Considering the following example of a text fragment of the collection:

```
Cytoplasts from two unrelated patients with MELAS (mitochondrial
myopathy, encephalopathy, lactic acidosis, and strokelike episodes)
harboring an A-*G transition at nucleotide position 3243 in the
tRNALeU(UUR) gene of the mitochondrial genome were fused with human
cells lacking endogenous mitochondrial DNA (mtDNA)
```

[1] http://www.hmdb.uniba.it/

MELAS is an instance of the *pathology* associated to the mutation under study, $A-*G$ is an instance of the *substitution* that causes the mutation, *transition* is the *type* of the mutation, 3243 is the *position* in the DNA where the mutation occurs, and $tRNALeU(UUR)$ gene is the instance of the *gene* correlated to the mutation.

By modelling the sentence structure, annotation rules that take into account the dependence between these entities can be discovered, as the following logical clauses show:

```
substitution(X) ← follows(Y,X), type(Y).

type(X) ← distance(X,Y,3), position(Y),
   word_between(X,Y,‘‘nucleotide position’’).
```

In the above example, it is noteworthy that *type*, *substitution* and *position* are classes of entities of interest, and that learning their classification models independently could not lead to the expected result. Most of the studies on inductive learning make the implicit assumption that concepts are independent (independence assumption). In many real applications, like in this one, this is not always true since capturing dependence among entities may lead to more accurate extraction models.

3 Learning Concept Dependencies

Rules for automated entity extraction are automatically learned in form of mutually recursive patterns by means of ATRE [2] [15]. ATRE is an inductive learning system that supports multiple concept learning. In multiple concept learning, the aim is to learn for each concept a set of interacting predicate definitions or properties that hold among various predicates. In this application, each concept plays the role of an annotation class (i.e., template slot) and the learning problem is formulated as a single-class problem, namely a textual object can not be multi-annotated. ATRE solves the following learning problem:

Given

- a set of concepts $C_1, C_2, \ldots, C_r$ to be learned,
- a set of observations O described in a language $\mathcal{L}_O$,
- a background knowledge BK described in a language $\mathcal{L}_{BK}$,
- a language of hypotheses $\mathcal{L}_H$ that defines the space of hypotheses S_H
- a user's preference criterion PC,

Find a (possibly recursive) logical theory $T \in S_H$, defining the concepts $C_1, C_2, \ldots, C_r$, such that T is complete and consistent with respect to the set of observations and satisfies the preference criterion PC.

[2] http://www.di.uniba.it/ malerba/software/atre

The language of hypotheses $\mathcal{L}_H$ and the language of background knowledge $\mathcal{L}_{BK}$ is that of linked, range-restricted definite clauses. The language of observations is *object-centered*, that is, observations are represented as ground multiple-head clauses, called *objects*, which have a conjunction of simple literals in the head that are examples of the concepts $C_1, C_2, \ldots, C_r$. They can be considered either positive or negative according to the problem definition. In this application domain, the set of concepts to be learned is defined by means of a set of predicates `annotation(X)=annotation_class`. We are interested in finding rules which predict the class label of a textual object, namely a predicate definition for each class. No rule is generated for the case `annotation(X)=no_tag` that corresponds to negative examples. The main assumption made in ATRE is that each object contains examples explained by some base clauses of the underlying recursive theory. Therefore, by choosing as seeds *all* examples of different concepts represented in the head of one training object, it is possible to induce some of the correct base clauses. A parallel exploration of all candidate seeds is feasible and mutually recursive concept definitions will be generated only after some base clauses have been added to the theory.

Therefore, the search space is a forest of as many search-trees as the number of chosen seeds. Each search-tree is rooted with a unit clause and ordered by generalized implication. The forest can be processed in parallel by as many concurrent tasks as the number of search-trees (parallel-conquer search). Each task traverses the specialization hierarchy top-down through a separate-and-conquer strategy, but synchronizes traversal with the other tasks at each level. Search proceeds towards deeper and deeper levels of the specialization hierarchies until at least a user-defined number of consistent clauses is found. Task synchronization is performed after that all "relevant" clauses at the same depth have been examined. A supervisor task decides whether the search should carry on or not on the basis of the results returned by the concurrent tasks. When the search is stopped, the supervisor selects the "best" consistent clause according to the user's preference criterion. In this work, short rules, which explain a high number of positive examples and a low number of negative examples, are preferred.

This separate-and-parallel-conquer search strategy provides us with a solution to the problem of *interleaving* the induction of distinct concept definitions. Mined models reflecting dependencies among annotations may enable a context-sensitive recognition of them when the annotation is automatically performed.

4 Data preparation

The dataset is composed by a set of manually annotated pre-categorized texts. In the observation language adopted by ATRE, the body of a clause contains descriptions of texts reporting information obtained by a preprocessing phase, while related positive and negative examples are reported in the head of the clause on the basis of expert users' annotations. Counterexamples for all the classes of annotations to be learned are all the unlabelled tokens. Therefore, each text generates as many training examples as the number of described tokens.

Annotated texts are preprocessed by means of natural language facilities provided in the GATE (General Architecture for Text Engineering) infrastructure [6]. In particular, we exploit the ANNIE (A Nearly-New IE system) component to perform tokenisation, sentence splitting, part-of-speech tagging, named-entity recognition (e.g., persons, locations, organizations), mapping into dictionaries. We use both predefined dictionaries available with ANNIE (e.g., organization names, job title, geographical locations, dates, etc.) and domain-specific dictionaries that categorize biological entities such as diseases, enzymes, genes, etc. General domain dictionaries are used to disambiguate some terms (e.g. places and geographical locations are useful to recognize terms about the ethnic origin of the sample). Biology dictionaries are flat dictionaries of entities that are peculiar of the mitochondrial genome, they include lists of names about diseases, genes, methods of analysis, nucleic acids, enzymes, etc. Domain-specific dictionaries are also useful to perform a rough resolution of acronyms that in this domain is one of the sources of redundancy and ambiguity in data. This text engineering framework allows also to define user-specific components to integrate in a pipeline of text processing. For instance, it includes a finite-state transduction engine to recognize regular expressions over processed texts. We define regular expressions to identify appositions occurring in texts and some particular numeric and alphanumeric strings that are really frequent in this domain. Stopwords are removed, such as articles, adverbs, prepositions etc. (taken from Glimpse-glimpse.cs.arizona.edu), stemming is performed by means of Porter's algorithm for English texts [18].

4.1 Data Representation

The relational representation used to describe data considers a text fragment as the composition of smaller passages described in terms of properties of occurring tokens and relations among them. Properties express statistical (e.g., token frequency), lexical (e.g., alphanumeric, capitalized token), structural (e.g., structure of complex tokens such as alphanumeric strings, abbreviations, acronyms, hyphenated tokens), syntactical (e.g., singular/plural proper/not proper nouns, base/conjugated verbs) and domain-specific knowledge (e.g., an entity belonging to a dictionary). Relations describe structural properties, such as the composition of sentences in passages of text and tokens in chunks or directly in sentences. Properties or attributes of a token are represented by unary descriptors, while relations between two tokens are represented by binary descriptors. Each token and sentence is given a unique identifier based on its ordering within the given text. The predicate `word_to_string` maps an identifier to the corresponding stemmed token, `word_frequency` expresses the relative frequency of a token in the given text, `type_of` refers to morphological features as `allcaps`, `mixedcaps`, `upperinitial`, `numeric`, `percentage`, `alphanumeric`, `real number`. Part-of-speech are encoded by the predicate `type_pos`. Semantics is added by the `word_category` predicate. Structural knowledge is expressed by means of the binary predicate `follows` that states the relation of successor among tokens, while complex tokens are described by using `s_part_of` relations

on component tokens, `first` and `last` predicates that state strings corresponding to the first and second part of an hyphenated token, `lenght` predicate defining the lenght of component tokens, some predicates (e.g., `first_is_numeric`, `middle_is_char`) stating the lexical nature of tokens composing an alphanumeric string. Relational knowledge is also asserted with respect to domain dictionaries by expressing the distance among categorized tokens in the context of a sentence (`distance_word_category`). Numeric knowledge is handled by ATRE by means of on-line discretization algorithm.

In this application, we have basically defined two templates, one for the entity *mutation* composed by the following slots: *position* (i.e., position in the DNA where the mutation occurs), *type* (i.e., type of the mutation: insertion, deletion, translation, substitution, etc.), *type_position* (i.e., pieces of the DNA involved in the mutation and at which position in the DNA), *locus* (gene involved in the mutation), *substitution* (i.e., type of substitution: which nucleotide is substituted by which other). The other template concerns the *subjects* under study that is described by the following slots: *method* (method of analysis of mutations), *nationality* (geographic origin of the sample population that is under study), *category* (category of the sample population, e.g., families, single individuals, patients, etc.), *number* (dimension of the sample population), *pathology* (pathologies affecting the population).

Concerning the unit of observation for the learning step, namely the dimension of the context of example annotations, we restrict observations only to sentences containing at least a positive example (target sentences). Moreover, no relation among sentences is currently used. Hence, the extraction is local to sentences. This is to prevent the generation of unbalanced data sets from which IE systems typically suffer since only a very small number of phrases contains examples.

Following ATRE's language of observations, the body of an object describes the tokens composing a sentence while the head describes annotations associated to tokens. All literals in the head of the clause are examples (either positive or negative) of the concept `annotation(_)`. The complete list of concepts of interest for this domain is obtained by varying the label associated to the `annotation(_)` predicate in the set of annotation class values corresponding to template slots. It is noteworthy that models for automatic entity extraction can be learned by looking for dependencies among slots intra-template as well as slots inter-template. An instance of training observation is reported in the following:

```
annotation(22)=no_tag, ..., annotation(27)=pathology,...,
annotation(35)=no_tag, section(1)=abstract ←
sentence(21)=true, part_of(21,22)=true, ..., part_of(21,35)=true,
word_to_string(22)=preval, word_to_string(23)=trans-membran, ...,
word_to_string(34)=t14634c, word_to_string(35)=famili,
type_of(25)=upperinitial, ..., type_of(34)=alphanumeric,
s_part_of(23,36)=true, s_part_of(23,37)=true, first(23)=trans,
last(23)=membran, lenght(36)=5, lenght(37)=7,
```

```
first(24)=t, last(24)=c, lenght(24)=7, first_is_char(24)=true,
middle_is_numeric(24)=true, last_is_char(24)=true, ...
type_POS(22)=nn, ..., type_POS(35)=nns, word_frequency(22)=1,
..., word_frequency(35)=3, word_category(27)=ethnic_group, ...,
word_category(33)=disease, distance_word_category(27,29)=2, ...,
distance_word_category(32,33)=1, follows(22,23)=true,
follows(23,24)=true, ..., follows(34,35)=true
```

The constant 1 denotes the whole text, which belongs to an abstract of the collection, the constant 21 denotes the sentence described in this clause, while the constants 22, 23, ..., 35 denote identifiers of tokens that are described in the body of the clause.

4.2 Background Knowledge

The specification of the following domain specific knowledge:

```
follows(X,Z)=true ← follows(X,Y)=true, follows(Y,Z)=true
```

permits to define in a transitive way the relation of "successor" among tokens, while some domain knowledge can be specified to reduce redundancy in token values by expressing a sort of synonymy relation among biological terms, such as by means of the following clauses:

```
word_to_string(X)=transition ← word_to_string(X)=transversion
word_to_string(X)=substitution ← word_to_string(X)=replacement
```

that allow ATRE to generalize over tokens with the same meaning.

Moreover, in ATRE, it is possible to prevent the use of some predicates that are involved in the body of training objects and, rather, to use some other predicates that are intensionally defined in the background knowledge. This allows to support a form of abstraction in the inference strategy, that is, to perform a shift of representational language in order to eliminate superfluous details from the representation language. For instance, the following statements allow to compact some patterns into a unique predicate that unburdens the comprehensibility of the mined models.

```
char_number_char(X)=true ← first_is_char(X)=true,
      middle_is_numeric(X)=true, last_is_char(X)=true

number_char_char(X)=true ← first_is_numeric(X)=true
      middle_is_char(X)=true, last_is_char(X)=true

char_char_number(X)=true ← first_is_char(X)=true,
      middle_is_char(X)=true, last_is_numeric(X)=true
```

5 Experiments

We considered a data set composed by seventy-one papers concerning mithocondrial mutations selected for the annotation of HmtDb. We considered the abstract of each paper and we present results obtained for annotation classes related to the *mutation* template. The total number of annotated tokens is 355, that is, 1.68 tokens per target sentence and 8.65 per abstract. They correspond to about 10.3% of the total number of tokens that are described in the data set. The remaining tokens are considered as *no tagged* tokens (negative examples).

Table 1. Distribution of examples per folds.

Fold	**# abstract**	**# sentences**	**# locus**	**# position**	**# substitution**	**# type**	**# type_position**	**# no_tag**	**# literals in body**
F1	9	36	16	12	4	8	12	424	2424
F2	9	40	27	13	13	5	4	474	2552
F3	13	40	22	14	6	17	0	550	3098
F4	13	34	16	5	5	17	24	553	3260
F5	15	39	24	15	8	8	18	524	3083
F6	12	27	14	6	2	6	31	531	3199
Total	41	211	119	38	55	61	89	3085	17793

Performances are evaluated by means of a 6-fold cross-validation, that is, the set of seventy-one papers is firstly divided into six blocks (or folds) (Table 1), and then, for every block, ATRE is trained on the remaining blocks and tested on the hold-out block. Results have been evaluated on the basis of different perspectives on accuracy of a classifier. In particular, we computed for each concept, the number of omission and commission errors and the value of precision and recall. *Omission* errors occur when annotations of tokens are missed, while *commission* errors occur when wrong annotations are "recommended" by a rule. The omission measure is reported as the ratio of the number of omission errors and the number of positive examples, while the commission measure as the ratio of the number of commissions and the total of examples. The *recall* measure is computed as the ratio of positive examples correctly annotated (i.e., true positives) and the sum of true positives and the omissions (i.e., false negatives). The *precision* measure is computed as the ratio of true positives and the sum of true positives and the commissions (i.e., false positives). Experimental results are reported in Table 2 for each trial and average values on errors are also given.

We can observe that there is a high variability among trials. This is mainly due to an heterogeneous distribution of examples that leads to different degrees of data sparseness. However, the percentage of commission errors is very low with respect to the percentage of omission errors (the system misses annotations rather than suggesting wrong annotations) independently on the trail. This means that learned rules are quite specific. Some explanations can be drawn

Table 2. Experimental results: percentage values are reported.

Fold	**locus**		**position**		**substitution**		**type**		**type_position**	
	omiss.	**comm.**	**omiss.**	**comm.**	**omiss.**	**comm.**	**omiss.**	**comm.**	**omiss.**	**comm.**
F1	75	0.18	1	0	25	0	1	0.37	0	0
F2	40.7	0.31	46.15	1.54	0	3.08	80	0	0	3.08
F3	68.2	0.65	1	0	33.3	0	1	0	–	0.33
F4	68.75	0.32	1	0	40	0.32	88.23	0	0	0
F5	62.5	0.33	53.3	0.16	0	0	50	0.16	0	0%
F6	64.3	0.34	1	0	0	0.34	50	0.169	19.35	0.34
Avg	*63.24*	*0.35*	*83.25*	*0.28*	*16.39*	*0.16*	*78.04*	*0.117*	*3.87*	*0.16*
St.D.	*11.84*	*0.157*	*26.05*	*0.619*	*18.57*	*0.177*	*22.99*	*0.148*	*8.65*	*0.178*
Prec.	**76.2**		–		**82.7**		–		**76.5**	
Rec.	**36.75**		**16.75**		**83.61**		**21.9**		–	

by considering the complexity of learned theories described in Table 3 that reports coverage rates of discovered clauses as the ratio of the number of clauses and the number of training examples per trial. Indeed, we find that coverage rate is low for annotation classes affected by a more evident difference among the number of commissions and omissions (*locus*, *type*, *position*). Low coverage rate of learned theories explains also some low recall values. This can be due to the preprocessing module that is not completely apt to manage the variety of morpho-syntactic variations on the same term that affect this type of domain. By scanning the learned theories, we discover that for some annotation classes, many rules take into account the information on the specific occurrence of a token (predicate *word_to_string*) rather than involving information on the context of the example. This might also explain the nature of some commission errors such as in the case of *locus*. Specificities of learned theories is also due to the low percentage of positive examples with respect to negatives. Best performances are obtained on the *substitution* class for which the system learns more general and accurate theory. Indeed, examples of this class are the most homogeneous and the preprocessing module is able to produce discriminative descriptions.

Table 3. Experimental results. Complexity of the learned theory.

Fold	**locus**	**position**	**substitution**	**type**	**type_position**
	# clauses/ # pos. ex.	**# clauses/ # pos. ex.**	**# clauses/ # pos. ex.**	**# clauses/ # pos. ex.**	**# clauses/ # pos. ex.**
F1	37/103	31/53	3/34	25/53	6/77
F2	40/82	20/52	3/19	21/56	6/108
F3	34/97	18/51	2/32	21/44	5/89
F4	41/103	20/60	3/92	19/34	6/55
F5	38/95	17/50	3/30	24/53	6/49
F6	47/105	22/59	3/36	23/55	2/58
Avg	40.72	39.47	8.74	45.8	7.59

For the sake of completeness, some clauses learned by ATRE have been analyzed. Some of them follow:

```
annotation(X1)=type_position ←
   char_number_char(X1)=true

annotation(X1)=type_position ←
   type_of(X1)=alphanumeric, length(X1) ∈ [5..6]

annotation(X1)=position ← follows(X2,X1)=true,
   type_of(X1)=numeric, follows(X1,X3)=true,
   word_category(X3)=gene, word_to_string(X2)=position
```

The first rule states that $X1$ is annotated as *type_position* if it is an alphanumeric token composed by a char, a number and another char. This is one of the first rule that ATRE adds to the theory, that generally are the most general rules that cover many examples. Actually, information on *type_position* of a mutation are tokens such as $A1262G$, that means that A is substituted by G at position 1262 of the DNA. The second rule concerns the same concept and it states that $X1$ is annotated as *type_position* if it is an alphanumeric token which is about 5 long. The third rule states that $X1$ is annotated as *position* if it is a numeric token that is preceded by the token "position" and followed by a token of the "gene" category. This rule captures patterns like "an A-to-G mutation at *position 3426 (tRNALeu)*".

It is noteworthy that ATRE is able to discover meaningful dependencies as the following rules show:

```
annotation(X1)=position ← follows(X2,X1)=true
   annotation(X2)=substitution, follows(X3,X1)=true,
   follows(X1,X4)=true, word_frequency(X4) ∈ [6..6],
   annotation(X3)=type, follows(X1,X5)=true,
   annotation(X5)=locus, word_frequency(X1) ∈ [1..2]
```

This rule states that $X1$ is annotated as *position* if it is preceded by two tokens that have been annotated as *type* and *substitution*, respectively. Moreover it is followed by a token occurring about 6 times in the abstract and that is followed by a *locus* annotation. Finally, $X1$ is quite infrequent in the abstract.

```
annotation(X1)=position ← follows(X2,X1)=true
   annotation(X2)=substitution, follows(X1,X3)=true,
   annotation(X3)=locus, follows(X4,X1)=true, word_to_string(X4)=np
```

This rule states that $X1$ is annotated as *position* if it is preceded by two tokens, the first is the string "np" and the second has been annotated as *substitution*. Moreover, it is followed by a *locus* annotation.

Through this first-order logic formalism, the automatic entity extraction task can be reformulated as a matching test between a logic formula that describes a model and another logic formula that represents the properties of the text to be annotated. The antecedent of a rule describes properties that should hold between some tokens of the sentence, while the consequent specifies the annotation class of some token involved in the antecedent part.

6 Conclusions

Biomedical information extraction is an appealing task for ILP thanks to its ability to reason in presence of logically described examples and abundant domain knowledge. In this paper, we have proposed an application of recursive logical theories learning to a real-world IE task on the biomedical literature. Recursive rules are discovered by inducing mutually dependent definitions of predicates by means of the ILP system ATRE. This system allows us to discover meaningful dependencies among biomedical entities of interest that reflect implicit relations among entities. These can also subsume relations at the semantic level, such as the association of a mutation type with the responsible/involved gene. "Implicit" refers to the fact that the kind of relation is not found out. Moreover, results show that ATRE performs well when descriptions of examples are safe from inconsistencies and training observations are sufficiently homogeneous; anyway, it leads to low recall values when the inaccurate data preprocessing causes specificity of learned theories.

As future work, we plan to investigate the use of additional domain dictionaries such as taxonomical dictionaries to reduce redundancy in data as well as domain-specific tools for acronym and abbreviations resolution. Evaluation of the system on some recently made available biomedical datasets for ILP [7] will also be conducted to test performances on noisy free data. Combination of logical theories induction with probabilistic measures to handle noisy data is also worth to be investigated. Finally, we plan to explore the application of association rule mining from text to discover implicit relations among entities in form of strong co-occurrences of entities as alternative way to mine data.

References

1. J. S. Aitken. Learning information extraction rules: An inductive logic programming approach. In *ECAI*, pages 355–359, 2002.
2. M. Attimonelli, M. Accetturo, M. Santamaria, D. Lascaro, G. Scioscia, G. Pappada, M. Tommaseo-Ponzetta, and A. Torroni. Hmtdb, a human mitochondrial genomic resource based on variability studies supporting population genetics and biomedical research. *BMC Bioinformatics*, 1(6), 2005.
3. M. Berardi, M. Ceci, and D. Malerba. A hybrid strategy for knowledge extraction from biomedical documents. In *ICDAR workshop on "Neural Networks and Learning in Document Analysis and Recognition"*, Seoul, Korea, 2005.

4. M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *AAAI '99/IAAI '99*, pages 328–334. American Association for Artificial Intelligence, 1999.
5. M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, pages 77–86, 1999.
6. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and application. In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, USA, 2002.
7. J. Cussens and C. Nedellec, editors. *Proceedings of the 4th ICML Workshop on Learning Language in Logic (LLL05)*, Bonn, Germany, 2005.
8. S. Dzeroski, J. Cussens, and S. Manandhar. An introduction to inductive logic programming and learning language in logic. In J. Cussens and S. Dzeroski, editors, *Learning Language in Logic*, volume 1925, pages 3–35. 2000.
9. S. Ferilli, N. Fanizzi, and G. Semeraro. Learning logic models for automated text categorization. In *AI*IA 01: Proceedings of the 7th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, pages 81–86, London, UK, 2001. Springer-Verlag.
10. D. Freitag. Toward general-purpose learning for information extraction. In *Proceedings. of the 17th int. conf. on Computational linguistics*, pages 404–408, Morristown, NJ, USA, 1998. Association for Computational Linguistics.
11. M. Goadrich, L. Oliphant, and J. W. Shavlik. Learning ensembles of first-order clauses for recall-precision curves: A case study in biomedical information extraction. In *ILP*, pages 98–115, 2004.
12. L. Hirschman, J. C. Park, J. Tsujii, and L. Wong. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18:1553–61, 2002.
13. L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. Overview of biocreative: critical assessment of information extraction for biology. *Bioinformatics*, 6, 2005.
14. M. Junker, M. Sintek, and M. Rink. Learning for text categorization and information extraction with ilp. In *Learning Language in Logic*, pages 247–258, 1999.
15. D. Malerba. Learning recursive theories in the normal ilp setting. *Fundamenta Informaticae*, 57(1):39–77, 2003.
16. R. Mooney. Learning for semantic interpretation: Scaling up without dumbing down. In J. Cussens, editor, *Proc. of the 1st Workshop on Learning Language in Logic*, pages 7–15, Bled, Slovenia, 1999.
17. C. Nedellec, editor. *Machine Learning for Information Extraction in Genomics - State of the art and perspectives*, volume 138 of *Studies in Fuzziness and Soft Computing*. Springer Verlag, 2004.
18. M. F. Porter. *Readings in information retrieval*, chapter An algorithm for suffix stripping, pages 313–316. 1997.
19. H. Shatkay and R. Feldman. Mining the biomedical literature in the genomic era: an overview. *Journal of Computational Biology*, 10:821–855, 2003.