A Prolog based Information Extraction System

Dr. Martin Emms

Dept of Computer Science, Trinity College,
Dublin, Ireland
Martin.Emms@tcd.ie

Abstract. This paper describes a Prolog-based information extraction system, which can be used to populate a database with relevant biogenetic facts extracted from journal articles. The central component is a chartparser written in Prolog, providing linguistic annotation of text. Templates for biological relations refer to the annotation elements and remain thereby concise. We argue that the Prolog base allows the ambition of the linguistic analyses to be easily raised and lowered, something useful for information extraction experimentation. Prolog also enables highly expressive template matching to be easily supported. Some Prolog-specific efficiency enhancing measures will also be described. This includes tail-recursive processing of a specialised *trie* data structure, and the use of a module implementing 'virtual' automata in Prolog in a preparsing segmentation process.

1 Introduction

Information Extraction (IE) is the name which has been given to the process of distilling natural language texts (newswire, journal abstracts) into database entries. An example is the extraction from biogenetic articles of the relations between genes reported therein, for example turning:

> all three coactivators ARA70 , steroid receptor coactivator 1 , and RAC3/ACTR can enhance androgen receptor (AR) transcriptional activity at 1 nM dihydrotestosterone

into

```
regulates([ARA70],[AR,transcriptional,activity])
regulates([steroid,receptor,coactivator,1], [AR,transcriptional,activity])
regulates([RAC3/ACTR],[AR,transcriptional,activity])
```

The extracted information can be useful in a number of ways: as the data fed to data-mining techniques ([1] [2]), as an aid to literature navigation, or as a content representation for anaphora resolution. This paper describes such a system, implemented mainly in Prolog. The system is outlined in section 2, highlighting parts where the use of Prolog has been advantageous. Further details of the design are given in section 3 and section 4. Performance is summarised in section 5.

2 System Outline

The system comprises a part-of-speech (POS) tagger, a parser, and database entry creator, acting in a pipeline. These components are outlined in this section.

TAGGING: an initial $\mathcal{LEX}$ tagging is made by lexical lookup in a comprehensive morphological dictionary. This assigns each word a set of tripartite lemmas, consisting of base form, POS-tags (from a 12-strong set), and morphosyntactic features ((see (i)). To disambiguate this $\mathcal{LEX}$ tagging an order-2 Hidden Markov Model[3] first generates a $\mathcal{STAT}$ tagging (see 1(ii)): a best POS sequence, whose POS-tags come from the 80-strong BNC tagset [9]. This is then *mapped* to a underspecified $\mathcal{LEX}$ tagging (see 1(iii)), and essentially *unified* with the initial $\mathcal{LEX}$ tagging (see 1(iv)). This 'map and unify' approach allows the statistical disambiguation to be changed as a module and is ideally suited to Prolog implementation.

(1)

	was	bound
(i)	V(**be**):P1sg:P3sg	V(**bind**):PART V(**bound**):PRESnot3sg N(**bound**):sg
(ii)	VBD	VVN
(iii)	V(**X**):P1-3sg:P1-3pl	V(**Y**):PART
(iv)	V(**be**):P1sg:P3sg	V(**bind**):PART

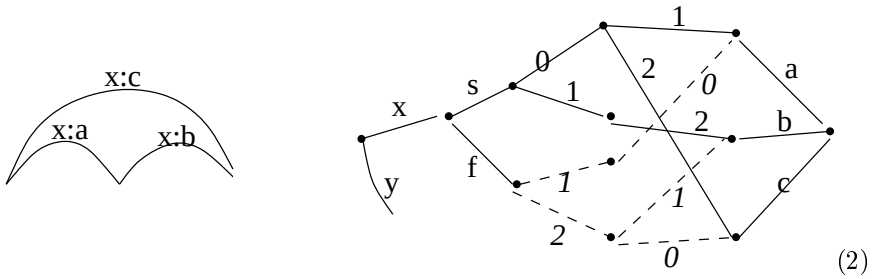
PARSING: The output of the tagger goes to a general purpose, bottom-up chart-parser. It is driven by both conventional grammar rules (eg.$np \Rightarrow det, n$) and categorial-grammar ([4] [5] [1]) type entries (eg. wants : $vp[fin]/vp[to]$), where individual words indicate the subcategorisation requirements. Logically complex categories are catered for (conjunction, disjunction,kleene-star) as well as features, sometimes variable. Variable feature are given a *universally quantified* semantics by a combination of variable refreshing and unification [1].

Rather than implement chart update using non-logical `assert/retract` features, the central predicate of the parser uses the technique of *tail-recursion* and an accumulator argument position to represent the chart.

A purpose-built data structure for edges is used, rather than relying on built-in fact indexing. The chart edges are stored in a kind of *trie* structure. An edge is an n-tuple, and so a set of these can be likened to a finite language. An acyclic finite state automaton can therefore compactly define this set, with arcs labelled with the elements of the n-tuples. In order to facilitate fast retrieval by both start and end points, edges are thought of as 6-tuples *(Cat, Dir, Crd1, Crd2, Feats, Rest)*. *Dir* is s or f according to whether *Crd1* is thought of as a *s*tart or

[1] thus if `swam` is universally quantified in its number feature, it may contribute in a single chart both to an edge spanning `fish swam`, with a plural subject noun-phrase, and to an edge spanning `my pet fish swam`, with a singular subject noun-phrase

*e*nd point. In (2) a chart with 3 `x` edges is shown, each with different features. To the right is the corresponding trie structure. We have used dashed lines and italic arc labels for the transitions in the f branch of the trie.



(2)

Notice that 2 paths with mirror image positional arc pairs always converge, so they share all other characteristics. Each trie branch has 5 arcs, and the time to search in this trie with a length 5 key depends on the branching factor, which grows linearly with the number of words. Also the sets $\{F \mid (c, x, F) \in \mathcal{C}\}$ (end-points given category and start) and $\{S \mid (c, S, y) \in \mathcal{C}\}$ (start-points given category and finish) can be obtained in linear time, in each case by searching with the appropriate 3 part key (either (c, x, s) or (c, y, f)).

Being able to search effectively with just start or end point is important because of the way the parser operates. Rather than pursuing a strict left to right strategy through the participants of a rule or subcategorisation, the parser widens a partial match to the left and right. Search to the left places requires fast retrieval on end points, while search to the right requires fast retrieval on start point. Particularly when rules and lexical entries mention more than 2 participants, this concentric widening strategy allows for early recognition of blind alleys.

We have compared this trie storage of edges with storage as a dynamic database, created with `assert` and deleted with `retract`. As an example, in a situation where 200 edges share the same end point, a search for an edge with a given end point can be solved by the trie system was 56% faster than the dynamic database system[2].

Beside the opportunity to customize the storage to suit the parser, there are other advantages to representing the chart as parameter-passed term, rather than using `assert`/`retract`. The final chart can be easily stored if the results of parsing are to be canned for later use. The implementation of *island-parsing*, described in section 4, is also facilitated. Finally, in debugging, it is desirable not to have to manage side-effects when execution takes an unintended path.

EXTRACTION: The output of parsing, a chart of linguistic information, is given to the final phase of IE. Here the chart is compared with *templates*. In the case of

[2] This has to do with the fact the Prolog indexes on predicate name and functor of first argument; the edge facts have category as first argument.

datamining from biogenetic source material, these are simple (ie. non-recursive) rules which spell out biologically relevant regions in terms of the categories which are in the chart. For example, this

```
reg_func_of(regulates(A,B)) +++>  np(A), vg(V), np(B),
  {member(V,[enhance,stimulate,inhibit, block,induce,inactivate]}.
```

will match[3]

$$\left\{\begin{array}{l}\text{ele1alpha}\\ \text{the ligand}\end{array}\right\}\left\{\begin{array}{l}\text{enhance}\\ \text{has enhanced}\\ \text{has been shown}\\ \text{to enhance}\end{array}\right\}\left\{\begin{array}{l}\text{ele1beta}\\ \text{the gene iso31}\end{array}\right\}$$

Matching is by a top-down, left-to-right parsing algorithm. Though top-down, left-to-right parsing can be computationally very expensive, here it is very fast as only the left-hand side template categories are really 'parsed': the building-block categories, those on the righthand side of a rule, are simply present in the chart.

Prolog is very suited to support the template matching process, allowing an expressive template syntax to be easily supported. Further details of this are given in the next section, but an obvious example is the ease with which one can match on any combination of the dimensions associated with words, such as the inflected form, base form, category, and features.

3 Recursive Structures

Extraction of noun-phrases is widely noted (eg. Bennet *et al*[6]) to be of value for IE purposes. Our parsing procedure delivers a chart, which records for sub-strings of the input, their syntactic category and optionally also their syntactic structure. This is more than *NP-tagging*, if by this is meant a *non-overlapping* set of NP substrings: the chart may indicate overlapping NP substrings. There is a method to extract an NP-tagging, currently a *grep*-style leftmost/longest selection of np-edges from the chart, to produce output like:

```
np([ELE1alpha]),
   has
np([no, intrinsic, transcription, activation, domain]),
   but
np([it]),
   does interact with
np([another, histone, acetyltransferase, p/CAF]),
```

One might feel that an 'NP-tagging', in the above noted sense, is really all one wants by way of mark-up to assist IE. Moreover it is the kind of output

[3] the `vg` part of this stands for *verb group* and is defined with a regular expression over tags

one can expect from a system relying exclusively on finite state automata. I think there are number of reasons why an 'NP-tagging', in the above noted sense, is not enough to support IE. Several phenomena covered by the system's comprehensive NP grammar lead to NP within NPs, such as coordination (3a), apposition (3b,c), and preposition phrases (3d):

a ARA70 , steroid receptor coactivator 1, and, RAC3/ACTR (3)
b the androgen coactivator, ARA70,
c ARA70, an androgen coactivator,
d the presence of 10 nM 17beta-estradiol

There is an IE-payoff to recording the NPs at different levels and their interrelation. Consider the sentence in (4a), from which the relation in (4b) should be extracted:

a ARA70 represents [$_{np}$ another important coactivator for [$_{np}$ ELE1alpha]] (4)
b `regulates(ARA70, ELE1alpha)`
c `NP (.. represent) np(.. coactivator for NP)`

Informally a template like (4c) is required, where `NP` and `np(..)` stand for a regions of text that are identified as noun-phrases, with material enclosed in the round brackets indicating further details[4]. Since the chart indicates nps at many levels, as well those which are 'top-level' nps, it is easy to support templates which specify such an 'np-within-np' pattern. Here is an example:

```
regulate_func_of(regulates(NP1,NP2)) +++>
    np(NP1), vg(V), np(_)/(star(anyword),
                           word(base:B),
                             words([for]),
                                np(NP2)),
   {member(V,[be,represent]), member(B,[coactivator,ligand])}.
```

which has a template item of the form $X/(Y)$, which indicates that the match for Y equal the match for X. However, without making reference to the nesting, it is problematic to skip intervening material without overshoot, extracting (5b) from (5a):

a ele1alpha is inert and ele1gamma represents another coactivator for ele1beta (5)
b `regulates(ele1alpha, ele1beta)`

Coordinated NPs are a further example of the utility of embedded structures. (6a) is an example, from which the 2 facts in (6b) should be extracted:

a [$_{np}$ [$_{np}$ ele1alpha] and [$_{np}$ ele1beta]] activate ele1gamma (6)
b `activates(ele1alpha,ele1gamma)` `activates(ele1beta,ele1gamma)`

[4] (4c) is an informal sketch of the template and is not the template syntax actually employed

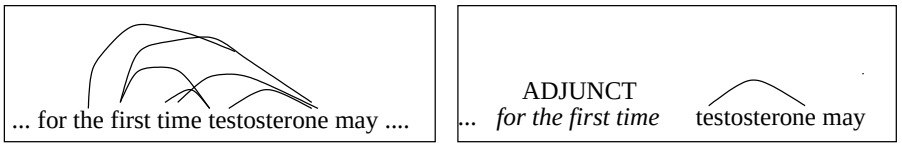
With multiple levels indicated it is very easy 'multiply out' a match to deliver the several database entries. Appositives provide a further example, as in (7a):

a [$_{np}$ the androgen coactivator, [$_{np}$ ARA70]] activates ELE1beta (7)
b `coactivate(ARA70,androgen)`
c `activate(ARA70,ELE1beta)`

One kind of template will look at the first np and extract with reference to its sub-np the fact in (7b). But another template will relate this whole NP to a second np, extracting the fact (7c). This requires multi-level NP-tagging.

4 Island Parsing and Prolog FSAs

The chartparser may construct np edges which are not necessary or which are misleading for extraction. Consider the picture to the left:



The words in the first time are here an *adjunct*[7], but the parser can combine them with succeeding nouns in many ways. To prevent edges leading into adjunct phrases, it is desirable to segment the input at adjunct phrases, parse the sublists separately, and to then recombine into a single chart. There is a huge number of such adjunct phrases (eg. *however*, *always*, *for example*, *for the first time*, *in addition*), and instead of constructing a list by hand, the adjunct phrases for segmentation (≈ 5000) were extracted from the Susanne[8] corpus of parsed English[5].

From this list, a Prolog-based finite-state automaton `adj` was defined, with arc transitions labelled with relevant *base* forms. The input to be segmented is actually in the form of lists of tags, each tripartite, containing a category, base-form and features (see section 2), and thus is not suitable for segmentation using `adj`. However, our Prolog implementation of FSA-driven leftmost/longest matching can handle not only arc labels which are atoms (ie. traditional FSAs), but also arc labels which are proper Prolog terms, including terms containing variables. This made it easy to define a *virtual* segmentation FSA `adj_tag` out of `adj`. A few rules suffice to make `adj_tag`-arcs be labelled by tags which have underspecified fields except for the base form part, with this required to label the corresponding `adj`-arc. It is with reference to `adj_tag` that the input is actually segmented.

It is also the case that NP-edges should not lead into verb-groups, so segmentation on verb-groups also make sense. Again an FSA with underspecified terms labelling its arcs was used for this, this time cyclic, due to the iterability

[5] This a corpus of general English, not focussed on any particular domain.

of verbs and adverbs. This time the arcs leave everything variable but for the part-of-speech.

The process of chartparsing each chunk and then combining the results into a single chart is made easier by the fact that the parser does not represent a chart as an `assert`-ed database. This would require two `assert`-ed databases to be managed, on the the currently parsed chunk and another for the accumulating overall parse.

5 Results

The system has been applied so far to MEDLINE abstracts on biogenetic research. Guided by biologists, a set of templates was drawn up to identify certain kinds of relations of interest. To give an impression of the kind of extraction task that the system performs, the following indicates some of the diverse expressions from which a `regulates` relation is extracted.

AR enhances ER the role of AR as a coactivator for ER AR, a coactivator for ER, the ER coactivator, AR,	`regulate(AR,ER)`

and other template targets are:

androgen receptor (AR)	`synonym(AR,androgen receptor)`
AR functions as a ligand	`is-a(AR,ligand)`
interaction between AR and ER	`interact(AR,ER)`

POS tagging accuracy is around 97%. Around 20% of words are unknown substance names and 90% of these are guessed right. NP extraction performed with 81% precision, 77 % recall. In the most recent experiments, 20-30 extraction templates were constructed, and performance on a 100 abstract set was 84% recall, 75% precision. Overall speed was 93 words/sec [6]. This relies on some further optimisations not described here, including statistical edge suppression and orthogonal parsing of parenthesised material.

6 Conclusions and Directions for futher work

There have been a number of proposals for IE based on the use of FSAs (eg. [10]). Whilst exploring methods of optimum IE it seems desirable to not tie into such a restrictive framework immediately. Within the system here described it is relatively easy to revise between successive development cycles the linguistic ambition of the analysis, both upwards and downwards.

The system's performance is promising, both in terms of speed and accuracy. The usefulness of Prolog has been noted at several points. It was of use in

[6] on 64 MB, 333 MHz PC running linux.

designing a modular process by means of which the output of a statistical tagger can disambiguate a lexical look-up. It is ideal for the programming of template matching against the structured information provided by the parsing process. It also facilitated generalised FSAs whose edges are labelled by more or less underspecified terms.

There are many outstanding challenges. Amongst these are the treatment of verb-phrase coordination and relative clauses. Syntactically, these are within the scope of the parser, but the challenge is to arrange for resort to be taken to the relevant mechanisms only when necessary. Another issue is the semi-automatic coverage of phrasal alternations such as active/passive and nominalisation, whereby the same information in expressed in different forms.

Acknowledgements

The work reported here made in part possible by the EUREKA Biopath project and the cooperation of The Centre for Information and Language, Munich University, who provided access to Medline abstracts and to the microbiological expertise needed for design and evaluation of templates. Their assistance is gratefully acknowledged.

References

1. Jong C. Park, Hyun Sook Kim, and Jung Jae Kim. Bidirectional incremental parsing for automatic pathway identification with combinatory categorial grammar. In *Proceedings of the Pacific Rim Symposium on Biocomputing 2001*, 2001.
2. Akane Yakushiji, Yuka Tateisi, and Yusuke Miyao. Event extraction from biomedical papers using a full parser. In *Proceedings of the Pacific Rim Symposium on Biocomputing 2001*, 2001.
3. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989.
4. Martin Emms. Parsing with polymorphism. In *Proceedings of the Sixth Annual Conference of EACL*, 1993.
5. Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*, chapter Categorial Type Logics. Elsevier, 1997.
6. Nuala A. Bennet, Qin He, Kevin Powell, and Bruce R. Schatz. Extracting noun phrases for all of medline. In *Proceedings of the AMIA 99*, 1999.
7. Rodney Huddleston. *Introduction to the Grammar of English*. Cambridge University Press, 1984.
8. Geoffrey Sampson. The Susanne Corpus `ftp://ftp.cogs.susx.ac.uk/pub/users/geoffs/SUSANNE.tgz`. 1995.
9. Lou Burnard et al. British National Corpus. 1995.
10. J.R. Hobbs, D. Appelt, D. Israel, and M. Kameyama. 1997. Fastus: A cascaded finite state transducer for information extraction from natural language text. In E. Roche and Y.Schabes, editors, *Finite State Devices for Natural Language Processing*. ILLC, University of Amsterdam.