



# Distributed modules for text annotation and IE applied to the biomedical domain

Harald Kirsch\*, Sylvain Gaudan, Dietrich Rebholz-Schuhmann

EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK

Received 15 April 2005; accepted 30 June 2005

## KEYWORDS

Text mining;  
Information extraction;  
Online tool;  
Protein–protein  
interaction;  
UniProt;  
Gene ontology

**Summary** Biological databases contain facts from scientific literature that have been curated by hand to ensure high quality. Curation is time-consuming and can be supported by information extraction methods. We present a server software infrastructure which allows to easily plug in modules to identify biologically interesting pieces of text to be then presented in a web interface to the curator. There are modules which identify UniProt, UMLS and GO terminology, gene and protein names, mutations and protein–protein interactions. UniProt, UMLS and GO concepts are automatically linked to the original source. The module for mutations is based on syntax patterns and the one for protein–protein interactions relies on chunk parsing. All modules work as separate servers possibly distributed on different machines and can be combined into processing pipelines as necessary. Communication is based on XML annotated text streams, each server processing the XML elements it is designed for, and possibly adding more information in the form of XML annotation. The server and the underlying software are available to the public.

© 2005 Elsevier Ireland Ltd. All rights reserved.

## 1. Introduction

Biologists rely on facts from public databases like GenBank,<sup>1</sup> LocusLink<sup>2</sup> and UniProt<sup>3</sup> and increasingly integrate facts from scientific literature into

OMIM,<sup>4</sup> FlyBase,<sup>5</sup> Gene Ontology (GO)<sup>6</sup> or COSMIC.<sup>7</sup> Curation of such facts is time-consuming and costly. It can be supported by text mining methods [13], but information extraction is not yet able to fully replace curators, since these databases target 100% precision.

Tools to support the curation process extract facts and context [7]. Results are then presented to curators for evaluation and eventually are added

---

\* Corresponding author. Tel.: +44 1223492593.

E-mail addresses: kirsch@ebi.ac.uk (H. Kirsch), gaudan@ebi.ac.uk (S. Gaudan), rebholz@ebi.ac.uk (D. Rebholz-Schuhmann).

<sup>1</sup> <http://www.ncbi.nlm.nih.gov/Genbank>.

<sup>2</sup> <http://www.ncbi.nlm.nih.gov/LocusLink>.

<sup>3</sup> <http://www.ebi.ac.uk/uniprot>.

<sup>4</sup> <http://www.ncbi.nlm.nih.gov/omim>.

<sup>5</sup> <http://www.flybase.org>.

<sup>6</sup> <http://www.geneontology.org>.

<sup>7</sup> <http://www.sanger.ac.uk/perl/CGP/cosmic>.

to the database. A supervisor may resolve conflicts [1].

Such information extraction (IE) tools have to meet a number of demands. First, protein and gene names (PGNs) have to be identified in the text. Although protein and gene names can be gathered from public resources like Human Genome Nomenclature (Hugo)<sup>8</sup> and UniProt, the sets are not complete and do not cover the full morphological variability encountered in the literature. Promising automatic extraction methods have been reported [4], but the BioCreative contest revealed lower performance,<sup>9</sup> thus leaving the problem unsolved. Second, relevant facts associated to PGNs have to be extracted, like disease, tissue type, species, indication of function and mutations of sequence. Again controlled vocabularies are available, e.g. UMLS<sup>10</sup> and GO, but are also not complete. In the case of mutations, syntax patterns support proper identification [8]. Finally, identification of relationships between PGNs, e.g. extraction of protein–protein interactions [12,6,9], is relevant to determine protein function.

Current IE tools like PASTA<sup>11</sup> are geared towards special tasks [3]. Others make only use of terminological resources [7]. No IE tool exists that fulfills all above-mentioned demands for the following reasons. The complexity of data makes it difficult to provide fully annotated data sets to train IE techniques based on machine learning, since annotation is time-consuming and costly. In addition, growing demands from curation teams and diversity of their needs require a flexible solution, which can incrementally be extended with new components. To meet this challenge and to provide an appropriate service, we developed a modular software environment which tackles basic tasks for curators.

We implemented a server software solution which annotates biological text with protein–protein interactions and mutations on top of named entities (e.g. proteins) that are linked to biomedical databases where possible. The server is typically accessed via a web browser. Its modular design allows integration of controlled vocabularies (GO, UniProt, UMLS), of syntax pattern sets, e.g. for abbreviation definitions, and of natural language processing like the identification of protein–protein interactions. The server is available through EBI's Web server and accepts text via cut and paste.

## 2. Available modules

The available modules belong to three categories: (1) basic NLP modules which mainly identify syntactical information, (2) modules which match controlled vocabularies, (3) modules which match a set of syntax patterns, and (4) modules for shallow parsing based on cascaded patterns. The categories are not independent, since named entity (NE) recognition relies on controlled vocabularies as well as on patterns for the identification of yet unknown NEs. Most modules match regular expressions (REs). These are compiled into a finite state automata (FSA) engine we implemented. It allows, for example, to combine 190,000 protein and gene names from UniProt (see below) into one FSA, such that each name, when identified in text, can be linked to the database entry (or entries) where it is mentioned as primary name or synonym. Due to the way an FSA works, processing time for a text is linear in the length of the text and in wide ranges unrelated to the size of the dictionary [5].

*Basic NLP modules.* These comprise the sentencer and a part-of-speech (PoS) tagger. The sentencer splits text into sentences and wraps them into a SENT XML element with a unique ID. The PoS tagger<sup>12</sup> was trained on the British national corpus, but contains lexicon extensions for the biomedical concepts. Noun phrases (NPs) are identified with syntax patterns equivalent to DET (ADJ|ADV) N+.

*Controlled vocabularies.* Identification and tagging of terminology is a variant of NE recognition. In biology and medicine a large number of concepts is stored in databases like UniProt, where roughly 190,000 database entries link PGNs to protein function, species and tissue type. PGNs from UniProt are transformed into REs which account for morphological variability. For example *col1a1* is transformed into the pattern `(COL1A1|[cC]o11a1)` and *IL-1* into `(IL|[Ii]l) [-]1`. The PGNs available from the database automatically generate a suitable link from the text to one or more database entries. While adding more dictionaries is technically trivial, it creates the problem of conflicting definitions. Already UniProt introduces the uppercase concept names *CAT*, *NOT* and *FOR* as PGNs. Disambiguation of such definitions will be added as soon as available.

*Syntax patterns.* A number of IE tasks are solved with syntax patterns. The following modules are integrated into the server: (1) identification of abbreviations, (2) definitions of PGNs, and (3) identification of mutations.

<sup>8</sup> <http://www.gene.ucl.ac.uk/hugo>.

<sup>9</sup> <http://www.pdg.cnb.uam.es/BioLINK/BioCreative.eval.html>.

<sup>10</sup> <http://www.nlm.nih.gov/research/umls>.

<sup>11</sup> <http://www.dcs.shef.ac.uk/nlp/pasta>.

<sup>12</sup> Developed at CIS, <http://www.cis.uni-muenchen.de/>.

Abbreviation extraction is described for example in [2]. In our approach a variety of patterns equivalent to NP '(' token ')' is used, where the token has to be the abbreviation of NP. We gathered all abbreviations from Medline in a dictionary as previously proposed [10]. As for UniProt, the dictionary is compiled into an FSA. If an abbreviation is found in the text without its expanded form, however, it is often necessary to decide whether it is indeed an abbreviation and which expansion applies (work in progress).

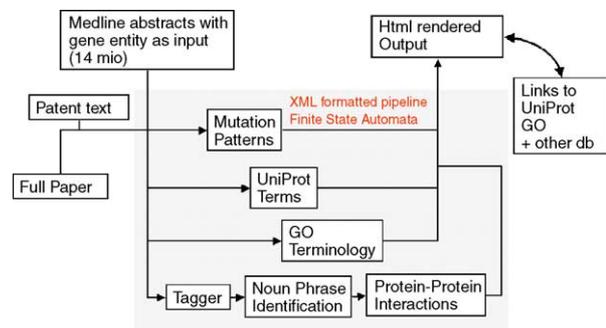
A separate module identifies sentence pieces where the author explicitly stated the fact that the concept denotes a PGN. Examples are *The AZ2 protein was...*<sup>13</sup> and *PMP22 is the crucial gene...*<sup>14</sup> Such examples were translated into the following four patterns: (1) the X protein, (2) the protein X, (3) T domain of NP, and (4) NP is a protein. The X denotes a single token and T represents a selection of concepts which are known to be used in conjunction with a protein. The tokens *the*, *is*, *a* and *protein* again represent sets of equivalent tokens.

Identification of mutations is integrated as described in [8]. Integrated patterns identify nomenclature equivalent to AA [0–9]+ AA, where AA denotes all variants of an amino acid or nucleic acid. Apart from the infix representation of the mutation, any postfix and prefix representation is covered as well as other syntactical variation.

*NLP based IE.* One component identifies and highlights protein–protein interactions. It is essential that a phrase describing an interaction contains a verb or a nominal form describing an interaction like *bind* or *dimerization*. In total, 21 verbs are considered including 10 verbs which are specific to molecular biology like *farnesylate*. A protein–protein interaction is identified and tagged, if such a verb phrase connects two noun phrases and if at least one of the NPs contains a PGN according to the terminology tagging.

### 3. Pipeline of modules shared in distributed computing

The presented modules do not work independently of each other (Fig. 1). For example the protein–protein interaction module uses NP detection (basic NLP module) which itself relies on PoS tagging. In addition, NP detection integrates marked up concepts from the terminology tagging



**Fig. 1** A selection of modules is available from the Whatizit server. These modules offer identification of UniProt and gene ontology (GO) terms. In case of a match the identified term is hyperlinked to the corresponding database entry. Combined tagging of UniProt and GO terms can be interpreted as functional annotation of proteins. In addition mutation patterns are identified in conjunction with Uniprot Terms. Last but not least protein–protein interactions are extracted based on modules, which allow chunk parsing.

module for the identification of protein–protein interactions. The modules form a pipeline equivalent to a UNIX pipe like `"cat input.txt|inputFilter|sentence|dictfilter|mutations|tagger|...> output.xml"`. Dependencies between the modules have to be kept in mind to determine their correct order. While the text passes through the pipeline, every filter picks the XML element it is responsible for and copies everything else unchanged to the output.

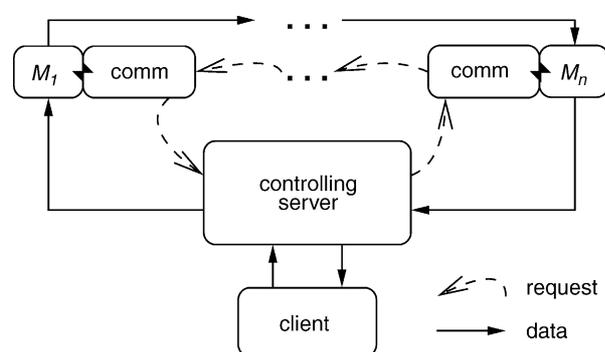
The input filter wraps arbitrary natural language text into an XML element describing the source of the document. Any further module analyses the text and adds meta data (XML tags). The following synthesis phase combines the facts available into larger structures, e.g. mutations of a gene or protein–protein interactions.

Running the pipeline of modules on a single compute node leads to insufficient response time, since the modules tend to have large memory footprints. In particular the PoS-tagger as well as terminology taggers load large dictionaries into memory and therefore have considerable startup time, whereas steady state operation is fast. To solve this problem, each module is a dedicated server process that is kept in memory for immediate response.

REs are applied for processing of data and meta data. This leads to a special constraint in the handling of XML tags. It is well known that REs cannot match recursive parenthesized structures. As a result, XML elements used as meta data are not allowed to contain themselves. If the XML elements denote parts of a phrase structure of a natural language sentence, this may in principle be a restriction, but in practical applications it is not.

<sup>13</sup> PMID 10580148.

<sup>14</sup> PMID 7628084.



**Fig. 2** Processing modules  $M_i$  are plugged into communication components (comm). The controlling server sends a request to the last component in the pipe. Each component contacts its predecessor for input and routes it throughout the module. The first component finally contacts back to the controlling server to fetch the input and send it down the pipe.

We implemented a set of Java classes which allows to set up distributed pipelined processing. It solves the details of client/server communication to run IE modules in a pipeline and allows modification to and replacement of modules through the developer (researcher). As a result, any class with a method that reads from an input stream and writes results to an output stream can serve as a module. In Java terms, the applied interface is a `java.lang.Runnable` calling its methods in `void run()`. A general purpose server class is available which, given a factory method to create the `Runnable`, handles all the details of setting up and shutting down the connections. In particular, connections to establish a pipeline  $M_1, \rightarrow \dots \rightarrow, M_n$ , are created as follows (Fig. 2).

The controlling server  $C$  generates the pipeline of modules  $M_1, \rightarrow \dots \rightarrow, M_n$ . Typically a component in the web server creates a reversed list of the modules and adds itself to the end of the list:  $M_n, \dots, M_1, C$ . Then it removes  $M_n$  from the list, contacts  $M_n$ , sends it the shortened list  $M_{n-1}, \dots, M_1, C$  and starts reading input from  $M_n$ . Module  $M_n$  follows the same procedure as the server and starts the `Runnable` which performs its function receiving input from the upstream server and writing output to the downstream server. All modules act the same way and finally  $M_1$  contacts the controlling server  $C$  to obtain the input data. Obviously  $C$  needs to write data to  $M_1$  and read data from  $M_n$  in parallel.

## 4. Conclusion

The presented server solution has been set up to support curators of biomedical facts in their work.

Its modules identify domain knowledge for molecular biologists and automatically link into public data resources. We are unaware of any existing solution like ours, which can integrate modules for information extraction tasks into a process pipeline based on XML. In collaboration with curation teams for UniProt and COSMIC, the modules will undergo evaluation for their usefulness in the curation process. Eventually, information will be automatically extracted and inserted into public databases.

Every module needs proper evaluation. Mutation extraction produces reliable data [8], but will be extended (chromosomal aberrations). The protein–protein interaction module relies on chunk parsing and demonstrates how NLP is integrated as a separate module. Together with curation teams single modules will be adapted to their needs. In particular the integration of controlled vocabularies for species and tissue types are of strong interest as well as additional NLP modules, e.g. for the identification of gene regulation.

A given combination of modules has to consider the dependencies between modules to allow efficient handling of information extraction tasks. When a user requests tagging of UniProt protein and gene names as well as information extraction for protein–protein interactions, the former is actually redundant, because it has to be run anyway for the latter to work. As a conclusion the curation teams will propose the proper combination of modules that they need.

Normalization of identified information is another step. One example is simplification of acronym definitions, e.g. transformation of "... androgen receptor (AR) ..." into "<ac id='1'>AR</ac>" with meta data accompanying the sentence specifying the expansion "<ex id='1'>androgen receptor</ex>". The result is normalized text which is easier to parse and thereby leads to better IE results.

The server has been tested on Medline abstracts and on PDF documents (full papers from Medline). As [11] have shown, the sections of full text scientific publications have noticeably different information content. The modular system described allows us to easily add a module for sectioning of full text publications.

## 5. Availability

The server is accessible via WWW on <http://www.ebi.ac.uk/Rebholz-srv/whatizit> and currently provides nine different pipelines. The software used to implement the server

infrastructure and the modules<sup>15</sup> is available on <http://www.ebi.ac.uk/Rebholz-srv/whatizit/software>.

## References

- [1] S. Albert, S. Gaudan, H. Knigge, A. Raetsch, A. Delgado, B. Huhse, H. Kirsch, M. Albers, D. Rebholz-Schuhmann, M. Koegl, Computer-assisted generation of a protein-interaction database for nuclear receptors, *Mol. Endocrinol.* 17 (8) (2003) 1555–1567.
- [2] J. Chang, H. Schutze, R. Altman, Creating an online dictionary of abbreviations from medline, *J. Am. Med. Assoc.* 9 (6) (2002) 612–620.
- [3] R. Gaizauskas, G. Demetriou, P. Artymiuk, P. Willett, Protein structures and information extraction from biological texts: the pasta system, *Bioinformatics* 19 (1) (2003) 135–143.
- [4] D. Hanisch, J. Fluck, H.-T. Mevissen, R. Zimmer, Playing biology's name game: identifying protein names in scientific text, in: *Proceedings of the Pacific Symposium on Biocomputing*, 2003.
- [5] J. Hopcroft, J. Ullman, *Introduction to Automata Theory Languages and Computation*, Addison Wesley, 1979.
- [6] M. Huang, X. Zhu, Y. Hao, D. Payan, K. Qu, M. Li, Discovering patterns to extract protein–protein interactions from full texts, *Bioinformatics* 20 (18) (2004) 3604–3612.
- [7] H. Muller, E. Kenny, M. Sternberg, Textpresso: an ontology-based information retrieval and extraction system for biological literature, *Publ. Library Sci.* 2 (11) (2004) e309.
- [8] D. Rebholz-Schuhmann, S. Marcel, S. Albert, R. Tolle, G. Casari, H. Kirsch, Automatic extraction of mutations from medline and cross-validation with omim, *Nucl. Acids Res.* 32 (1) (2004) 135–142.
- [9] A. Rzhetsky, I. Iossifov, T. Koike, M. Krauthammer, P. Kra, M. Morris, H. Yu, P.D. Weng, W. Wilbur, V. Hatzivas-siloglou, C. Friedman, Geneways: a system for extracting, analyzing, visualizing, and integrating molecular pathway data, *J. Biomed. Informat.* 37 (1) (2004) 43–53.
- [10] S. Schwartz, M. Hearst, A simple algorithm for identifying abbreviation definitions in biomedical text, in: *Proceedings of the Pacific Symposium on Bio-computing*, 2003.
- [11] P. Shah, C. Perez-Iratxeta, M. Andrade, Information extraction from full text scientific articles: where are the keywords? *Bioinformatics* 4 (20) (2003).
- [12] J. Temkin, M. Gilder, Extraction of protein interaction information from unstructured text using a context-free grammar, *Bioinformatics* 19 (16) (2003) 2046–2053.
- [13] A. Yeh, L. Hirschman, A. Morgan, Evaluation of text data mining for database curation: lessons learned from the kdd challenge cup, *Bioinformatics* 19 (2003) i331–i339.

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)



<sup>15</sup> Except the PoS tagger licensed from CIS, Munich.