# Distributed Modules for Text Annotation and IE applied to the Biomedical Domain

**Harald Kirsch** and **Dietrich Rebholz-Schuhmann**
EMBL-EBI
Wellcome Trust Genome Campus
Hinxton, Cambridge CB10 1SD, UK
{kirsch,rebholz}@ebi.ac.uk

## Abstract

Biological databases contain facts from scientific literature, which 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 which identifies biological facts in scientific text and presents the annotation to the curator. Such facts are: UniProt, UMLS and GO terminology, identification of 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 on NLP. All modules work independently of each other in single threads and are combined in a pipeline to ensure proper meta data integration. For fast response time the modules are distributed on a Linux cluster. The server is at present available to curation teams of biomedical data and will be opened to the public in the future.

## Contents

## 1 Introduction

Biologists rely on facts from public databases like GenBank[1], LocusLink[2] and UniProt[3] and increasingly integrate facts from scientific literature into OMIM[4], FlyBase[5], Gene Ontology (GO)[6] or COSMIC[7]. Curation of such facts is time-consuming and costly. It can be supported by text mining methods (Yeh et al., 2003), but information extraction is not yet able to fully replace curators, since these databases target 100% precision.

Tools to support curation extract facts and context. Results are then presented to curators for evaluation and eventually are added to the database. A supervisor may resolve conflicts (Albert et al., 2003).

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)[8] and UniProt, the sets are not complete and do not cover the full morphological variability encountered in the literature. Promissing automatic extraction methods have been reported (Hanisch et al., 2003), but the BioCreAtIve contest revealed lower performance[9], 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[10] and GO, but are also not complete. In the case of mutations, syntax patterns support proper identification (Rebholz-Schuhmann et al., 2004). Finally, identification of relationships between PGNs, e.g. extraction of protein-protein interactions (Temkin and Gilder, 2003), is relevant to determine protein function.

Current IE tools like PASTA[11] are geared towards special tasks (Gaizauskas et al., 2003). 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

[1] www.ncbi.nlm.nih.gov/Genbank
[2] www.ncbi.nlm.nih.gov/LocusLink
[3] www.ebi.ac.uk/uniprot
[4] www.ncbi.nlm.nih.gov/omim
[5] www.flybase.org
[6] www.geneontology.org
[7] www.sanger.ac.uk/perl/CGP/cosmic
[8] www.gene.ucl.ac.uk/hugo
[9] www.pdg.cnb.uam.es/BioLINK/BioCreative.eval.html
[10] www.nlm.nih.gov/research/umls
[11] www.dcs.shef.ac.uk/nlp/pasta

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 by 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 solution which annotates facts identified in biological text and links them to biomedical databases where possible. The server is typically accessed via 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 will be available through EBI's Web server[12] and accepts text via cut&paste or via a URL.

## 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 matched with a finite state automata (FSA) engine we implemented. It is optimized for pipelined execution and huge REs.

*Basic NLP modules* comprise the sentenciser and a part-of-speech (PoS) tagger. The sentenciser splits text into sentences and wraps them into a `SENT` XML element with a unique ID. The PoS tagger[13] 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 190000 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]ol1a1)` 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.

Abbreviation extraction is described for example in (Chang et al., 2002). In our approach a variety of patterns equivalent to `NP '(' token ')'` is used, where the token has to be the abbreviation of NP. If an abbreviation is found in the text without its expanded form, however, it is 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 explicitely stated the fact that the concept denotes a PGN. Examples are `The AZ2 protein was ...`[14] and `PMP22 is the crucial gene ...`[15]. 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 (Rebholz-Schuhmann et al., 2004). 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 base 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 to-

[12] Open to the public after assessment for heavy load

[13] developed at CIS, `www.cis.uni-muenchen.de`

[14] `PMID 10580148`

[15] `PMID 7628084`

tal, 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

Obviously the presented modules do not work independently of each other. 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 concepts from the terminology tagging module for the identification of protein-protein interactions. The modules form a pipeline equivalent to a UNIX pipe like `"cat input.txt | inputFilter | sentencise | 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. One solution which solves this problem is to implement each module as a dedicated server process, which 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.

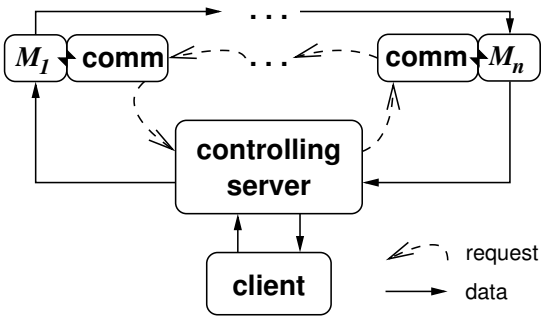


Figure 1: Processing modules $M_i$ are plugged into communication components (comm). The controlling server sends a request to the last component in the pipe. Each component contacts its predecessor for input and routes it throught the module. The first component finally contacts back to the controlling 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 $M_1, \rightarrow \ldots \rightarrow, M_n$, are created as follows (fig 1):

The controlling server $C$ generates the pipeline of modules $M_1, \rightarrow \ldots \rightarrow, M_n$ (fig. 1). Typically a component in the web server creates a reversed list of the modules and adds itself to the end of the list: $M_n, \ldots, M_1, C$. Then it removes $M_n$ from the list, contacts $M_n$, sends it the shortened list $M_{n-1}, \ldots, M_1, C$ 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 already produces reliable data (Rebholz-Schuhmann et al., 2004), 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 simplication 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 (Shah et al., 2003) have shown, the sections of full text scientific publications have noticably different information content. The modular system described allows us to easily add a module for sectioning of full text publications.

## References


S. Albert, S. Gaudan, H. Knigge, A. Raetsch, A. Delgado, B. Huhse, H. Kirsch, M. Albers, D. Rebholz-Schuhmann, and M. Koegl. 2003. Computer-assisted generation of a protein-interaction database for nuclear receptors. *Molecular Endocrinology*, 17(8):1555–1567.

J.T. Chang, H. Schutze, and R.B. Altman. 2002. Creating an online dictionary of abbreviations from medline. *Journal of the American Medial Association*, 9(6):612–620.

R. Gaizauskas, G. Demetriou, P.J. Artymiuk, and P. Willett. 2003. Protein structures and information extraction from biological texts: The pasta system. *Bioinformatics*, 19(1):135–143.

D. Hanisch, J. Fluck, H.-T. Mevissen, and R. Zimmer. 2003. Playing biology's name game: identifying protein names in scientific text. *Pacific Symposium on Biocomputing*.

D. Rebholz-Schuhmann, S. Marcel, S. Albert, R. Tolle, G. Casari, and H. Kirsch. 2004. Automatic extraction of mutations from medline and cross-validation with omim. *Nucleic Acids Research*, 32(1):135–142.

P.K. Shah, C. Perez-Iratxeta, and M.A. Andrade. 2003. Information extraction from full text scientific articles: where are the keywords? *Bioinformatics*, 4(20).

J.M. Temkin and M.R. Gilder. 2003. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19(16):2046–2053.

A.S. Yeh, L. Hirschman, and A.A. Morgan. 2003. Evaluation of text data mining for database curation: lessons learned from the kdd challenge cup. *Bioinformatics*, 19:i331–i339.