Technical notes provide unique applications, innovative methods, and clear protocols designed specifically for Stratagene reagents, instruments, and software.

Technical Note

Interactions between biological entities such as proteins and small molecules are central to the understanding of biological systems. However, compiling information on interactions is challenging because these are scattered across the breadth of scientific literature. Manual curation of this literature is expensive due to its huge size and massive ongoing publication growth. This Technical Note describes how the PathwayArchitect® NLP Engine, a fully automated Natural Language Processing (NLP) system, extracts interactions from literature and can be a practical alternative to manual literature curation.

Introduction

A number of systems have been reported in the literature to process text available through sites like MEDLINE. The approaches used by these can be classified into two broad categories, shallow and deep. Shallow approaches use general techniques like pattern matching,[1,2] machine learning, and statistical methods with limited grammatical structure, examples and shallow parsing possibly combined with statistical methods.[2,9,10] Deep approaches use detailed grammar to comprehend sentence structure and meaning; these are more laborious to create but provide better accuracy.[5,7,11]

The PathwayArchitect NLP Engine is based on deep parsing and driven by an elaborate sentence grammar. The deep approach was chosen to have maximum control over different aspects of the sentence so as to maximize accuracy (i.e., correctness of facts extracted) without compromising recall (i.e., the number of facts extracted). Extraction of interactions is done in four main phases: named entity recognition, syntax analysis, semantic analysis and semantic inferencing. Note that the PathwayArchitect NLP Engine works sentence by sentence and extracts only interactions which occur completely within a single sentence. Each sentence is put through the four phase pipeline illustrated in Figure 1.

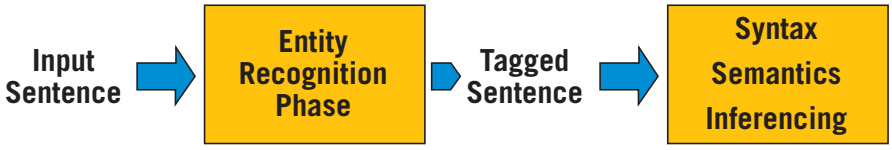

Figure 1
Entity Recognition

The first step in processing literature is to identify terms in the sentence corresponding to entities of interest. The PathwayArchitect NLP Engine currently considers the following terms: proteins, protein families, enzymes, small molecules, biological processes, and molecular functions. Other categories of interest, e.g., disease terms, will be added in the future. Statistics on the numbers of each term type in the mammalian database are shown in Figure 2.

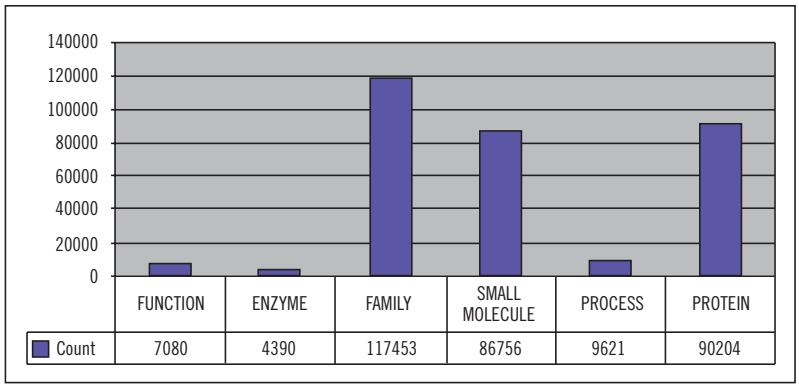

Figure 2
Statistics on the Numbers of Each Term Type in the Mammalian Database

Protein entities were obtained by parsing Entrez Gene data available in ASN format. Human, mouse, and rat entries are combined into a single mammal database. In this process, entries that share a common official symbol are merged. Results of the merger are illustrated in the Venn diagram (see Figure 3). Protein dictionaries for other organisms are in the process of being created. The small molecule dictionary was created using four sources: MeSH 2005 descriptor records, MeSH supplementary concept records, ChEBI and MEDLINE Name of Substance. Since there is redundancy amongst these sources, records from these sources were merged based on the presence of common aliases and registry numbers. The Enzyme dictionary was obtained from the Expasy site. The Protein Family dictionary was extracted from MEDLINE abstracts using a rule-based algorithm.

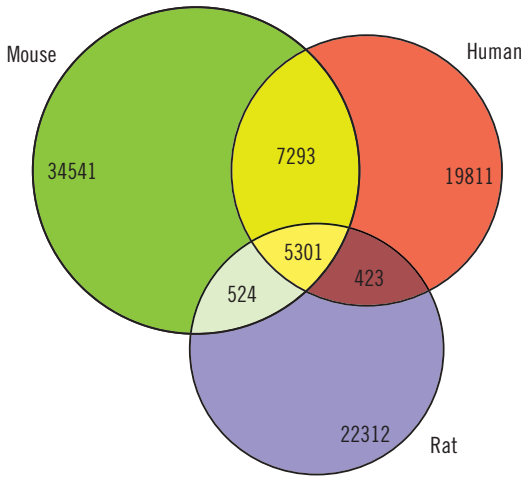

Figure 3
Results of Combined Mammalian Protein Database

For each term category, a detailed list of entity names is collected. Often, entities are referred to by multiple names (either alternative names or aliases), so synonyms are identified and grouped together. Finally, an algorithm for finding these entities' names and synonyms in text is needed. A straightforward dictionary lookup is insufficient as there are a number of variations in the way these terms appear in the text, e.g., the presence/absence of hyphens, brackets, and the use of Greek letters like α instead of alpha etc. The PathwayArchitect NLP Engine uses a tokenization procedure which breaks each term into basic units called tokens (e.g., NFKappaB2 (NF-κB2) is broken into tokens NF, Kappa, B, and 2), identifies matching tokens first, and then strings back collections of matching tokens into terms allowing for some mismatches, spaces, hyphens and other variations in the process (so NF-κ B2, NF-κ-B2, NFκB-2 will all match).

Syntax Analysis

The next phase is the syntax analysis phase in which the syntactic structure of the sentence is derived using a *context free grammar* for English, i.e., a set of transformation rules as illustrated in the figure below. The syntactic structure, also called the *syntax tree*, is a hierarchical breakup of the sentence into its underlying linguistic constituents like nouns, verbs, and adverbial/prepositional clauses and phrases. The syntax tree also captures the functional roles of different parts of the sentence like the subject, the object, the subject modifier, the object modifier and the predicate modifier. The grammar rules are specified in the form of Augmented Transition Networks (ATN)[1,6]. The syntax analyzer or the context free parser is based on the chart parsing algorithm[1]. An example of a syntax tree is shown in Figure 4.

Sentence → Subject Predicate Object
Sentence → Subject SubjectMod Predicate Object
Subject → Noun
Subject → Pronoun
Object → Noun | 'that' Sentence | ...
SubjectMod → NounPhrase | Sentence | ...
•~200 such rules.

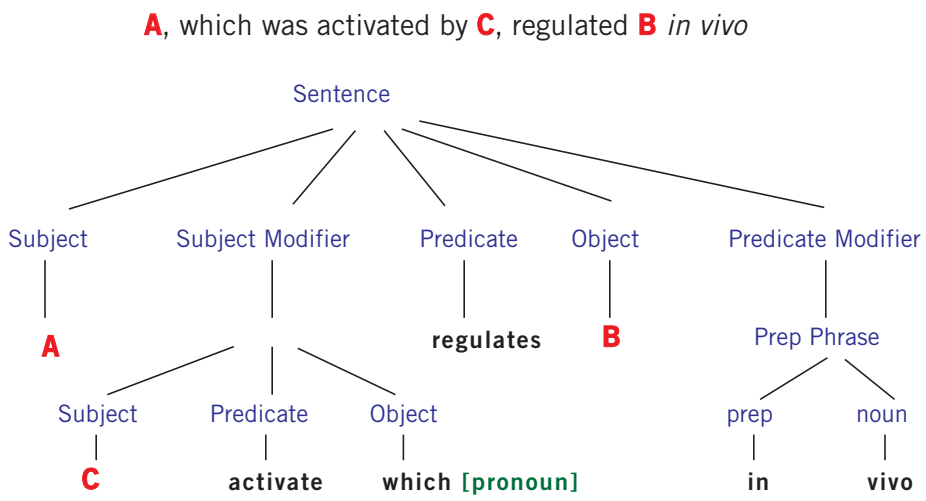

Figure 4
Example Syntax Tree

Before entity recognized sentences (the stretch containing an entity term is marked up) can be parsed for syntax tree construction, a lexical analysis step is needed. In this step, each word in the sentence is subjected to morphological analysis using a dictionary of English words derived from the UMLS Lexicon. The purpose of the morphological analyzer is to identify a word match with the dictionary handling variations caused by change of tense, number and person.

Note that syntax analysis could lead to multiple parse trees for each sentence due to inherent ambiguity in language as well as ambiguity in the grammar used to parse the sentence. All the parse trees resulting from a sentence share portions and are packed into a space-efficient structure for further processing.

Also note that several different syntactical structures and sentence formations could carry the same semantic content, as illustrated in the sentences in the figure on the right. The Semantic Analysis and Semantic Inferencing phases aim to round off syntactical structures so different syntactic structures carrying the same meaning are identified as one.

A regulates **B**.
B is regulated by **A**.
A plays a role in the regulation of **B**.
A belongs to the family of regulators of **B**.
B activity was found to be modulated by the addition of **A**.

All mean A $\xrightarrow{\text{regulates}}$ **B**.

Semantic Analysis

The conversion from syntax to semantics is driven by a manually created *semantic dictionary* which maps all words of interest to corresponding semantic concepts. For instance, the word *modulate* would map to the concept *regulate*. The semantic tree would then need to identify who regulates what. It uses the sentence structure imposed by the syntax tree to identify these agents. This process involves elaborate rules for each type of syntactical structure found in a sentence, i.e., prepositional phrases, subordinate clauses etc. Each syntax tree of the sentence could possibly define a different semantic tree; as in the case of syntax trees, these multiple semantic trees are packed into a space-efficient structure for further processing. An example semantic tree is shown in Figure 5.

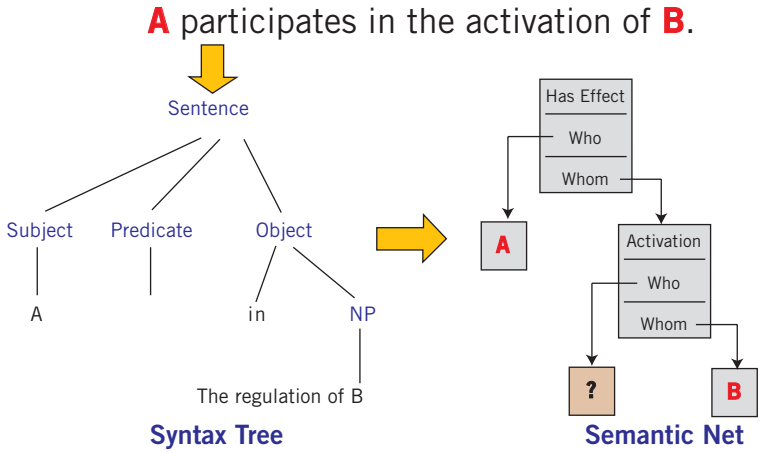

Figure 5
Example Semantic Tree

Note that the relationships captured in a semantic tree are only those which are immediately and directly indicated by the sentence structure. Only a fraction of the interactions are captured directly by such relationships. A good fraction of the interactions are captured indirectly and extracting these interactions requires making inferences across multiple relationships present in the semantic tree. For instance, in the semantic tree in Figure 5, who activates B is not directly available unless one runs an inference across the Activate and Has Effect concepts. The next phase of information inferencing does exactly this.

Inferencing: A good example of a sentence requiring inferencing is "Treatment by A causes an increase in the amount of B". The semantic tree would yield the following concepts: *cause, treatment, increase, amount*. To conclude that A regulates B positively, we would need to make inferences across these four semantic concepts, often using agents in one relationship to fill in missing holes in other relationships. This process is again driven by a set of domain-specific inference rules which make several passes through the semantic tree trying to unify the various concepts present in the semantic tree and inferring new concepts. In this process, it is possible that we augment the semantic network with new semantic nodes to represent the inferred concept. The inference rules are encoded as *tree transformation* rules, which specify the mapping from a source semantic subtree to a destination semantic subtree. Finally, we extract interactions by searching the resulting semantic network for interaction nodes with all its arguments filled.

Performance and Statistics: The PathwayArchitect NLP Engine analysis pipeline described above was run on all of the MEDLINE abstracts available until mid 2004. The run was performed on multinode Linux clusters. The resulting interactions were sampled at random and examined for correctness. Initial examination showed the need for a *post-processing step* to eliminate interactions obtained from false tagging (e.g., words like ml (milliliter) coming up as proteins, calcium being confused as both a protein and small molecule, etc). A post-processing module was then implemented and rules were added to it repeatedly to remove gross errors. Finally, about 500 interactions were drawn at random from the results and manually evaluated for accuracy. The pie chart in Figure 6 shows the accuracy results.

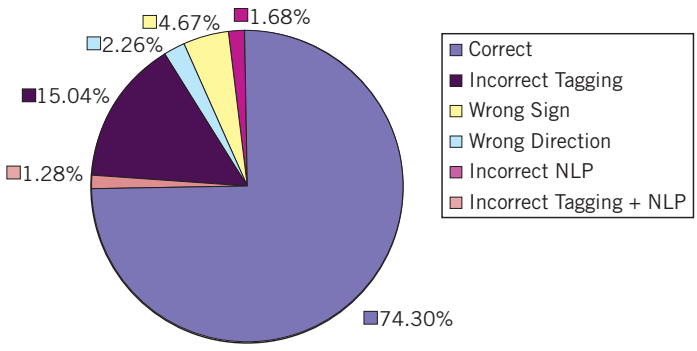

Figure 6
Accuracy of the Interaction Results (%)

Accuracy of the Interaction Results: About 74.3% of the roughly 500 interactions examined were found correct. An additional 4.67% had only a wrong sign (i.e., positive regulation became negative regulation or vice versa) but were otherwise correct. 2.26% had only a wrong direction (i.e., A regulates B became B regulates A). 18% were clearly incorrect and came about due to a variety of reasons. A majority of these, about 15%, were NLP related errors (i.e., syntax, semantics, inferencing), while a minority, about 1.6%, were related to incorrect entity tagging, and a small fraction, about 1.2%, were due to both types of errors occurring together.

An examination of the NLP related errors showed a variety of problematic sentence structures, but mostly negations (A does not have the ability to regulate B), and implicit and explicit conjunctions which result in mis-parsing (A increased, B decreased, D remained unaltered). Entity tagging related errors were caused by either entities which were wrongly tagged or entities which were partially tagged, i.e., only part of the entity was recognized.

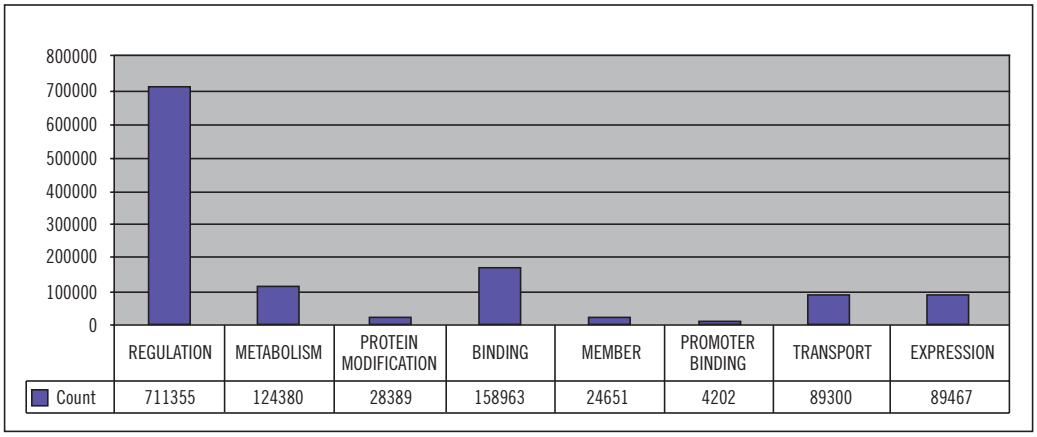

Figure 7
Interaction Statistics

The chart in Figure 7 shows the number of distinct interactions of each type that were extracted. The total number of distinct interactions over all interaction types was about 1.2 million of which more than half were of the generic regulation variety. Metabolism and Binding were well above 100,000 in number while Transport and Expression were just below 100,000.

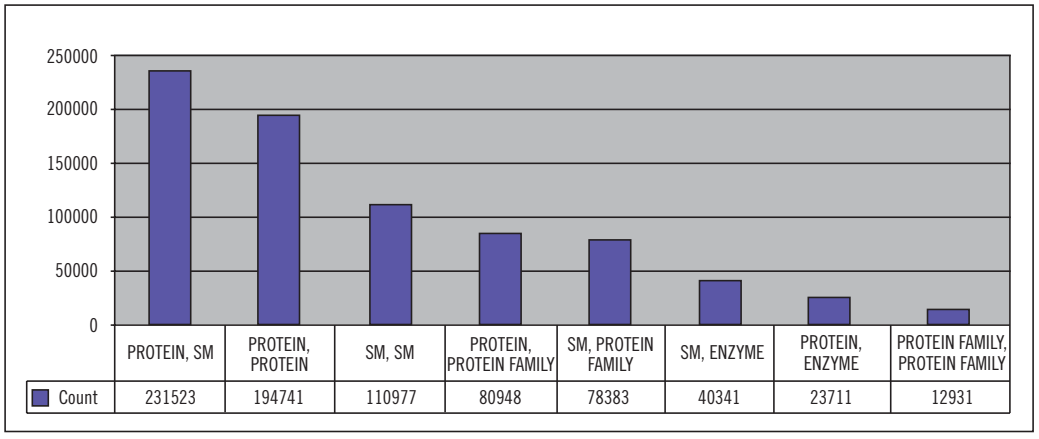

Figure 8
Entity Pair Statistics
SM = small molecules

Note that the same pair of entities (proteins or small molecules, etc.) could appear multiple times in the above counts as they could participate in different types of interactions or have multiple interactions with the same interaction type but with different effects (positive/negative/unknown regulation). Statistics on just *pairs of interacting entities (irrespective of interaction type or effect)* are presented in Figure 8. In all there are about 800,000 distinct pairs of entities detected as interacting. The break up across entity types is given on the chart in Figure 8. The current version of the PathwayArchitect software also includes interactions involving process and function terms.

Conclusion

The PathwayArchitect NLP Engine is an advanced biological text-mining tool and is an efficient method for searching literature for biological interactions. It functions as the natural language processing engine in the PathwayArchitect software that reads biological literature to extract facts.

We have generalized the PathwayArchitect NLP Engine to create interaction databases for *E. coli*, Arabidopsis, *C. Elegans*, Yeast, and Drosophila. The longer-term goal is to make the engine customizable to answer a wider range of queries, e.g., what cleavage sites are known about a protein, what genes have been patented for what function, etc. Additional goals include expanding the pool of full text articles on which the PathwayArchitect NLP Engine runs for greater coverage and to improving the accuracy by tightening mis-parsing and incorrect entity tagging.

REFERENCES

1. James Allen, (1995) *Natural Language Understanding*. Benjamin/Cummings, 2nd edition.
2. Applet, D., et al. (1995) *SRI International {FASTUS} System: Muc-6 test results and analysis.In {\em Proceedings of the 6th Message Understanding Conference}*, pages 237–248.
3 Blaschke, C., et al. (1999) *Automated Extraction of Biological Information from Scientific Text: Protein-Protein Interactions*. ISMB, pages 60–67.
4. Donaldson, I., et al. (2003) BMC Bioinformatics, 4(1).
5. Friedman, C., et al. (2001) *Bioinformatics* 17 (Suppl.1):S74–S82.
6. Jurafsky, D. and Martin, J.H. (2000) *Speech and Language Processing*. Prentice-Hall.
7. Novichkova, S., Egorov, S., and Daraselia, N. (2003) *Bioinformatics* 19:1699–1706.
8. Ono, T., et al. (2001) *Bioinformatics* 17:155–161.
9. Park, J.C., Kim, H.S., and Kim, J.J. (2001) *Pac. Symp. Biocomput.* 6:396–407.
10. Thomas, J., et al. (2000) *Pac. Symp. Biocomput.* pages 541–552.
11. Yakushiji, A., et al. (2001) *Pac. Symp. Biocomput.* 6:408--419.

LEGAL

For more information on this Technical Note
or to contact our Technical Service department:

Stratagene US and Canada
Technical Service: 800-894-1304 x2

Stratagene Europe
Technical Service: 00800-7400-7400

Stratagene Japan K.K.
Technical Service: 3-5821-8076

Visit our web page at:
www.stratagene.com/contacts/TechServices

Email our Technical Service department at:
techservices@stratagene.com

www.stratagene.com