Vol. 19 no. 13 2003, pages 1699–1706
DOI: 10.1093/bioinformatics/btg207

MedScan, a natural language processing engine for MEDLINE abstracts

Svetlana Novichkova, Sergei Egorov and Nikolai Daraselia*

Ariadne Genomics, Inc., 9100 Great Seneca HWY, Rockville, MD 20850, USA

Received on January 11, 2003; revised on February 11, 2003; accepted on March 17, 2003

ABSTRACT

Motivation: The importance of extracting biomedical information from scientific publications is well recognized. A number of information extraction systems for the biomedical domain have been reported, but none of them have become widely used in practical applications. Most proposals to date make rather simplistic assumptions about the syntactic aspect of natural language. There is an urgent need for a system that has broad coverage and performs well in real-text applications.

Results: We present a general biomedical domain-oriented NLP engine called MedScan that efficiently processes sentences from MEDLINE abstracts and produces a set of regularized logical structures representing the meaning of each sentence. The engine utilizes a specially developed context-free grammar and lexicon. Preliminary evaluation of the system's performance, accuracy, and coverage exhibited encouraging results. Further approaches for increasing the coverage and reducing parsing ambiguity of the engine, as well as its application for information extraction are discussed.

Availability: MedScan is available for commercial licensing from Ariadne Genomics, Inc.

Contact: nikolai@ariadnegenomics.com

INTRODUCTION

The need for automated data retrieval from biomedical publications is well recognized, given their exponentially increasing volume. Due to easy access and availability, the most widely used sources of scientific information are abstracts of scientific publications, accessed primarily through MEDLINE. Abstracts contain a concise description of the information within the paper and provide the best combination of availability, information density and brevity. Many approaches have been proposed for information extraction (IE) from scientific publications, ranging from simple statistical methods to advanced natural language processing (NLP) systems.

Basic information extraction approaches rely on the matching of prespecified templates (patterns) or rules (such as precedence/following rules of specific words). The underlying assumption is that sentences conforming exactly to a pattern or a rule express the predefined relationship(s) between the sentence entities. In some cases, these rules and patterns are augmented with additional restrictions based on syntactic categories and forms of words in order to achieve better matching precision. A number of groups reported application of pattern-matching-based systems for protein function information extraction (Sekimizu *et al*., 1998; See-Kiong and Wong, 1999; Blaschke *et al*., 1999; Ono *et al*., 2001). The shortcoming of such systems is their inability to correctly process anything other than short, straightforward statements, which are quite rare in information-saturated MEDLINE abstracts. They also ignore many important aspects of sentence construction such as mood, modality, and negation, which can significantly alter or even reverse the meaning of the sentence.

Several attempts have been made to utilize shallow-parsing techniques for the task of biological information extraction (Humphreys *et al*., 2000; Thomas *et al*., 2000; Park *et al*., 2001). Shallow parsers perform partial decomposition of a sentence structure. They identify certain phrasal components and extract local dependencies between them without reconstructing the structure of the entire sentence. In some cases, shallow-parsers are used in combination with various heuristic and statistical methods (Applet *et al*., 1995; Humphreys *et al*., 1998). The precision and recall rates reported for shallow parsing approaches are 50–80 and 30–70%, respectively. Shallow parsers perform well for capturing relatively simple binary relationships between entities in a sentence, but fail to recognize more complex relationships expressed in various coordinating and relational clauses. For sentences containing complex relationships between three or more entities, such approaches usually yield erroneous results.

Information extraction systems based on the full-sentence parsing approach (Yakushiji *et al*., 2001) tend to be more precise as they deal with the structure of an entire sentence, and variations of the full parsing-based approach have been applied for biomedical information retrieval. However, full parsers are significantly slower and require more memory than shallow analyses because they have to deal with general syntactic ambiguity and handle the full set of possible structures of whole sentences. A problem of parsing ambiguity can be eliminated or significantly reduced by employment of domain-specific context-sensitive grammars. This approach has been implemented in a system called MedLee

*To whom correspondence should be addressed.

(Friedman *et al*., 1994), which was initially developed for radiology report processing and subsequently adopted for protein function information extraction in another system called GENIES (Friedman *et al*., 2001). MedLee utilizes a parser and a semantic grammar consisting of a large set of nested semantic patterns (incorporating very little syntactic knowledge), which covers the most frequently used sentence structures. The downside of semantic grammar-based systems is the need for the complete re-development of grammar and lexicon for each particular knowledge domain.

Context-free parsing systems, on the other hand, are general enough to be applicable to any domain, but completely generic systems seem to be impractical and inefficient. We believe that a flexible and efficient information extraction system must contain two components—an NLP engine deducing the semantic structure of a sentence, and a configurable information extraction component to validate and interpret results produced by the NLP engine. We have chosen to build our NLP component on a special purpose context-free linguistic model for a particular domain of discourse. Such linguistic models can be developed using sublanguage analysis techniques, which result in some helpful restrictions on the range of linguistic data that need to be accounted for during parsing.

In this paper we present the first component of the proposed information extraction system—MedScan, a general biomedical domain-oriented NLP engine that efficiently processes sentences from MEDLINE abstracts and produces a set of semantic structures representing the meaning of each sentence. It is based on a context-free grammar and a lexicon developed specifically for MEDLINE as a result of its corpus analysis. The engine is implemented in C++ and compiled into a Windows application and is significantly faster than existing full sentence parsers. The described NLP engine will be used as a part of a pipeline aimed at the extraction of information about pathways and molecular networks.

METHODS

The MedScan architecture

The processing of a text is conducted in several stages: sentence tokenizing, word recognition and morphological analysis, recognition of compound lexemes, syntactic parsing, and semantic interpretation. Accordingly, the system is comprised of the following components, each having a specific function (Fig. 1):

Preprocessor—selects sentences potentially discussing protein functions and brackets protein and chemical names using provided name dictionaries.

Tokenizer—breaks the given input string into a number of primitive tokens (words, punctuation, numbers, etc.) according to the set of tokenizing rules.

Recognizer—identifies tokens as lexicon entries and performs morphological analysis to deduce their grammatical form.

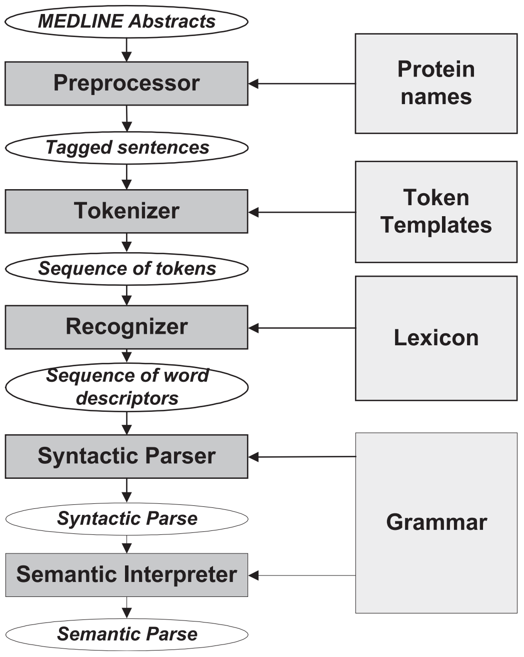

Fig. 1. The components and processing steps of the MedScan system.

Syntactic Parser—processes a sequence of input tokens augmented with syntactic categories and morphological information, to build a number of alternative syntactic structures of a sentence using a set of rules defined in grammar.

Semantic Interpreter—transforms a syntactic tree into a normalized semantic tree, which represents logical relationships between the words in a sentence.

In the following sections the structure of the MedScan lexical database and implementation of each component is described.

Lexicon and grammar

The MedScan NLP engine is based on the original grammar developed specifically to represent MEDLINE language. The grammar belongs to a class of Unification Grammars, and is built on the ideas closely related to Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982). The context-free core of the grammar is represented by means of transition networks augmented with *programs*. Each network represents the structure of a particular phrasal category. The lexicon stores lexemes and associated lexical and morphological information.

Lexicon The key concept of the lexicon is the *lexeme*. Most of the lexemes are stored in their canonical form with an indication of their morphological properties by means of inflection patterns. For those lexemes that do not obey general morphological principles, the lexicon stores a full set of *irregular forms* along with their syntactic features. Each lexeme is assigned a single lexical category and has a corresponding feature set defining specific syntactic properties of the lexeme (case and gender for nouns and pronouns, number–person, inflection pattern and subcategory for nouns, verbs, pronouns and adjectives). In addition to regular single-word lexemes, the lexicon also stores *compound lexemes*. A compound lexeme consists of two or more words, which can be substituted by a single lexeme. Proper identification and handling of such phrases in a sentence allows for the reduction of computational complexity and ambiguity of the parsing process. Compound lexemes are stored as lexical records of a special type with an indication of the head-lexeme (usually the last lexeme of the multi-word phrase), which defines the lexical category and inflection pattern of an entire compound lexeme.

The syntactic and semantic argument structure of a lexeme is described by its *subcategory* and *semantic frame*. Subcategorization is an important part of a grammar imposing selective lexical-dependent constraints on phrase structure by restricting order, number, and type of lexeme complements. This allows, for instance, to distinguish between syntactic arguments and adjuncts among the constituents attached to a verb in a sentence. Semantic frame is important for recovery of the correct predicate–argument relationships of a lexeme in a sentence.

Subcategories are defined as families of similar *syntactic frames*, each encoding one of the possible syntactic argument structures for a lexeme. Each syntactic frame is defined as a set of special features specifying a phrase category of each complement and in some cases its additional properties—such as a preposition name for prepositional phrases or control type for verbal phrases.

Semantic frames encode the semantic structure (or semantic pattern) of a lexeme. The semantic frame consists of a set of enumerated slots each assigned a specific thematic role (theta-role). We utilize a generally accepted set of roles such as ‘agent’, ‘patient’, ‘instrument’, ‘statement’. The relationships between a lexeme, its subcategories, and its semantic frames are stored in a special structure called *Binding*. Binding is defined as the combination of a lexeme with a single subcategory and a single semantic frame and is critical for the proper mapping of the syntactical complements of the lexeme to the slots of the corresponding semantic frame. Each lexeme may have one or more assigned bindings, each describing a possible argument pattern applicable to the lexeme.

The MedScan lexicon currently contains about 10 000 lexemes describing various domain-specific concepts and relations. It was manually constructed on the basis of various sources including Gene Ontology and Unified Medical Language System with consideration of the word frequencies measured in entire MEDLINE release 2001.

Grammar. We have chosen the *Augmented Transition Networks* (ATN) formalism to describe the context-free core of the English language grammar. The phrase structure of each non-terminal category is represented by a corresponding ATN. Networks used in the grammar are recursive transition networks with *jump arcs*. Each network has one initial state, one exit state, and a number of intermediate states (nodes) connected by arcs labeled with terminal or non-terminal categorical symbols. Each arc may be augmented with *test* and *action* programs (described below), which are executed during traversing of the arc. Most network arcs are also assigned a ‘syntactic function’ declaring the functional role each constituent plays in a phrase. These syntactic functions are used during semantic interpretation in mapping a syntactic structure of the constituent onto its possible semantic argument structure. MedScan grammar defines a commonly accepted set of 34 terminal (noun, verb, adjective, etc.) and 19 non-terminal (sentence, noun phrase, prepositional phrase) categories containing a total of 130 states and 281 arcs. The structures of these networks are the result of analysis of the language of MEDLINE abstracts and are unique to our text-processing engine.

Various grammatical properties of lexemes and their subcategories are defined in the grammar by means of *Feature Sets*. A feature set is a list of attribute-value pairs serving as a pre-initialized feature structure associated with each lexeme and stored in the lexicon. The feature system used in the grammar is capable of handling a wide range of morphological and syntactic phenomena, such as analytical word forms, word agreement, subcategorization, correct modifier attachment. We use a predefined set of features that can be roughly divided into the following categories:

- features handling grammatical agreement and morphology (including analytical forms)—*number, person, gender, case, tense, form*;
- ‘lexical’ features [*determiner, complementizer, preposition, wh-word* (such as which, who, where), etc.]—in most cases they are names of prepositions, determiners and other special lexemes. These features are used to prevent inappropriate lexemes from passing an arc of the network;
- subcategorization features (*comp1, comp2, control, preposition)*—this group of features is used to define syntactical frames and subcategories for nouns, verbs and adjectives.
- validation features (*path, endpoint*)—features describing child structure of an assembled constituent. These features are used in dealing with such phenomena as gap propagation, wh-movement and others.

The feature unification is performed by special *programs* augmenting arcs of networks. All programs can be divided into the following three categories. *Confirmation (test) programs* describe the restrictions on feature values of a constituent (grammatical form, number, person, subcategory, etc.) that can traverse an arc. In most cases, confirmation programs just verify if specific features can be unified. *Action programs* perform actual unification of feature values. *Validation programs* perform the final check of the feature values of a constituent that has been completed.

Preprocessor

The preprocessor reads the XML-based format of a MEDLINE record and splits the MEDLINE abstract text into individual sentences. Next, it utilizes a provided protein name dictionary to identify and tag the name of proteins and select the sentences containing at least one protein name.

Our approach to protein name identification is based on application of specialized tokenizer designed to ignore many variations of protein name spelling (‘TNFalpha1R’, ‘TNF alpha-1 R’, ‘TNFalpha-1(R)’, ‘TNF alpha1R’, etc.), followed by a simple and efficient subsequence search algorithm applied to token sequences. Preprocessor does not attempt to decipher abbreviations and relies on alternative protein names being provided explicitly. The example below shows the sentence output by preprocessor where identified protein names are surrounded by curly brackets, and are preceded by an identifier of the corresponding gene in the HUGO consortium index of human genes. This sentence will serve as a model sentence in the description of all subsequent stages of the processing.

{**11768** = TGF-beta 2} inhibits {**11892** = tumor necrosis factor} activity in resting lymphocytes after treatment with nitric oxide.

Tokenizer

The tokenizer converts an input text stream into a sequence of *tokens*. Our implementation uses a set of hard-coded finite-state machines to recognize the most frequently occurring tokens (words, numbers, punctuation), and a configurable set of regular expressions to deal with special cases (mutations, cell line genotypes, etc.). The first set allows optimal performance for the most frequent types of tokens while the second set provides flexibility in tuning the tokenization process to various domain-specific texts.

All finite-state automata are applied to an input text in the order defined by their priorities. If all of them fail to recognize a token, the stream is skipped to the next clear marker (space or punctuation) and the skipped portion of text is converted into a special ‘unknown’ token treated subsequently as an unrecognized word.

Recognizer

The goal of the recognizer is to convert a sequence of tokens into a sequence of *word descriptors*. We define a word descriptor as a unit representing several alternative variants of assignment of a particular lexeme (each with proper grammatical features such as syntactic category, subcategory, semantic frame) to a token. Morphological analysis is performed to identify each token as a lexeme in a certain grammatical form, based on a set of inflection rules for each syntactic category. All unknown words are considered to be nouns.

After word recognition is completed, a *compound lexeme matching* procedure is performed. This procedure quickly identifies the multi-word compound lexemes occurring in a sentence by efficient searching for patterns stored in a lexicon and replacing the corresponding subsequence of tokens with a single *compound word descriptor*. The grammatical form of a compound word descriptor is deduced from the grammatical form of its head lexeme.

Syntactic parser

The MedScan parser is based on the active chart parser algorithm (Allen, 1994) in combination with bottom-up parsing approach. Active chart parsing has a number of benefits compared to other algorithms. It keeps record of all created constituents and thus eliminates the necessity of backtracking and allows the reuse of constructed constituents. It also allows for compact representation of local ambiguity and implementation of various ‘packing’ techniques, which accelerate processing significantly. Since our goal was to implement an extremely efficient parser, MedScan utilizes a number of additional algorithmic improvements and advanced programming techniques.

Unlike the classical chart parser, which operates with rewrite rules, MedScan uses the grammar in a form of augmented transition networks. ATNs are formally equivalent but are a significantly more compact and efficient representation of rewrite rules, where common parts of rules are packed into a single network path and are traversed only once during the parsing. The most computationally expensive step is an arc extension, which includes search for candidate arcs and performing feature unification. We have implemented a two-dimensional indexing structure that uses phrasal category and constituent position as keys to quickly look up the candidate arc extension. In addition, a special filtering procedure rapidly identifies and filters out arcs that cannot be further extended, resulting in more efficient memory management.

Operations with features also consume significant processing time, since they are performed at every attempt of an arc traverse. The MedScan feature system and augmentation programs were designed to allow effective implementation of feature comparison and unification. MedScan contains a special-purpose virtual machine executing feature operations. Programs stored in MedScan grammar in a simple text form are compiled into a run-time representation and interpreted by this virtual machine.

MedScan grammar uses a pre-defined set of features and feature values. We found that this approach is capable of handling most of the grammatical phenomena we have encountered in MEDLINE and has a significant technical advantage: compact feature implementation in a form similar to bit vectors. Bit-vector representation of features allows extremely efficient comparison, unification, and other set-theoretical operations. We also found that feature value comparison is significantly less expensive computationally than new feature structure initialization. However, by the very nature of a natural language-to-grammar mapping, most of the evaluated run-time constituent combinations fail the unification process. We therefore split the unification process into two steps—feature evaluation, performed by test programs, and actual unification (initialization of new feature structure and value assignment), performed by action programs. The second step is performed only if the first has succeeded. In addition, each test program is designed to evaluate the features with the highest probability of unification failure first.

Packing was found to be a very effective method to encode ambiguity; it tremendously decreases the computational complexity of parsing by concurrently maintaining thousands of alternative syntactic trees in a compressed form. The general idea of packing is to introduce a special 'packed' type of constituent, which serves as a single representative of a set of similar constituents. 'Similarity' means that all the properties of constituents taken into account during an arc extension are identical. In our implementation, similar constituents have the same category, start, and current positions, and an identical set of features. When a particular active arrow is completed and a new constituent is created, a special routine evaluates whether it can be packed with any existing constituents. If it occurs, this single 'packed' constituent is used in place of original ones in all subsequent processing steps. Effective implementation of constituent packing in MedScan accelerates parsing up to three orders of magnitude, depending on the structure of a sentence.

An example below shows partial syntactic structure and main constituents (surrounded by square brackets) of the model sentence. The phrasal category is shown immediately following each constituent—separated by a slash and printed in a bold typeface (**NP** designates noun phrase, **V**, verb, **PREP**, preposition, **PP**, prepositional phrase, **VP**, verbal phrase, **S**, sentence).

[[{11768=TGF-beta 2}]/**NP** [[inhibits]/**V** [{11892=tumor necrosis factor} activity]/**NP**]/**VP** [in/**PREP** [resting lymphocytes]/**NP**]/**PP** [after/**PREP** [treatment with nitric oxide]/**NP**]/**PP**]/**S**

After the parsing, this sentence will generate at least four alternative syntactic structures—due to the variations in prepositional phrase attachment: both '*in lymphocytes*' and '*after treatment with nitric oxide*' can independently and alternatively attach to a verb phrase forming verb phrase modifiers ('*phosphorylates ... in lymphocytes*' and '*phosphorylates . . . after treatment with nitric oxide*'), and to the subject noun phrase '*tumor necrosis factor activity*', forming larger noun phrases '*tumor necrosis factor activity in lymphocytes*' and '*tumor necrosis factor activity after treatment with nitric oxide*'.

Semantic interpreter

The semantic interpreter transforms the representation of the surface structure of a sentence produced by the syntactic parser into its semantic representation. Semantic representation generated by the interpreter can be considered 'shallow'; it is much closer to the linguistic concept of deep structure than to the complex representations of meaning used in artificial intelligence and knowledge representation systems. Semantics in MedScan represent the meaning of a sentence as a tree of categorized predicate–argument relationships between lexemes. The interpreter does not validate the rendered representation and makes no attempts to restrict the generated structures by taking into account any domain-specific information. This task is delegated to the *ontological interpreter*, the module, which is currently under development. The general principles of ontological interpretation are briefly described in the results section, and its full description will be published elsewhere.

A semantic tree is built from elementary *semantic nodes*, each representing a particular sentence lexeme. Semantic nodes reference other *argument* semantic nodes through their *slots* and contain two types of slots. A fixed number of categorized *role slots* are encoded by a semantic frame, possibly associated with a node's lexeme in a lexicon. Role slots represent the most important lexeme relationships, such as subject or object of action; the name of a role slot (e.g. 'agent', 'patient') reflects the nature of slot–argument relationship. In addition, each semantic node can contain an arbitrary number of labeled *attribute slots* representing auxiliary lexeme relations such as time, place or mode of action. The arguments of attribute slots are constructed from various sentence modifiers such as prepositional phrases, nouns in noun phrases, adjective and adverbial phrases, relative clauses, appositions, and the attribute slot label reflects the source and method of argument construction (e.g. name of preposition for prepositional phrases). An example of a slightly simplified semantic structure generated from one of the syntactic trees of the model sentence is presented below.

```
inhibition
{
     agent: {11768=TGF-beta 2}
     patient: activity
                 { patient: {11892=tumor necrosis factor} }
     attribute: [type: ''in''; value: resting lymphocytes]
     attribute: [type: ''after''; value: Treatment
                                           {agent:
                                             subject:
                                             substance: nitric oxide
                                           }]
```

MedScan's approach to semantic interpretation is similar to the one introduced in LFG (Sells, 1985), and relies on special 'syntactic functions' to differentiate the syntactic roles of child constituents, and to populate the respective slots of parent semantic node with references to child nodes. Semantic interpretation of each constituent is performed by a set of special hard-coded scripts—one for each non-terminal phrasal

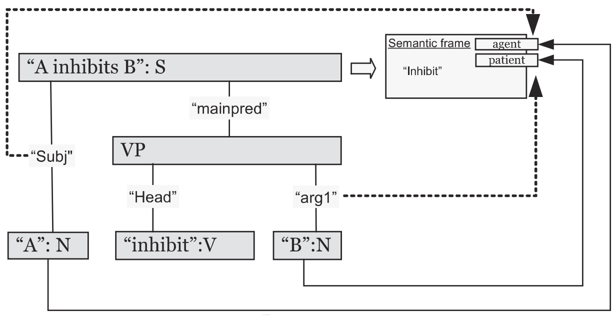

Fig. 2. An outline of the semantic interpretation procedure. Constituents are shown as shadowed rectangles with indication of their category. Each parent–child constituent relation (vertical lines) is labeled by syntactic function. Dotted arrows illustrate analysis performed by a script, which uses syntactic functions to distinguish between the slots of semantic frame 'inhibit' and populates them (solid arrows) with proper references to child semantic nodes.

category. These scripts also analyze the feature set of a parent constituent to account properly for active or passive voice, presence of gaps, modality, negation, and to resolve more sophisticated syntactic cases such as rising to subject and object. The process of semantic interpretation is outlined in Figure 2.

All cases of syntactic conjunctions are represented by a special type of semantic node, called packed node. Packed semantic nodes always contain exactly two *packing slots* referencing packed conjuncts and are labeled with a logical type of conjunction. Packed semantic nodes can be nested and similarly to regular nodes can contain an arbitrary number of attribute slots.

Relative clauses represent another special case when a particular semantic node serves as an argument of more than one parent node. In a sentence, relative clauses are expressed by various syntactic constructs, such as *wh*-phrases, active or passive verbal phrases, and appositions and are represented by attribute slots labeled 'Rel Agent' or 'Rel Patient' depending on their syntactic activity or passivity in a relational clause.

RESULTS AND DISCUSSION

We ran our evaluation on the complete MEDLINE 2001 release, which is about 40 GB and contains about 5.9 million abstracts (roughly 61 million sentences). The scope of our experiments was limited to sentences describing human protein function. The dictionary of human protein names was compiled on a basis of HUGO consortium data and additionally enriched by incorporating protein names, aliases, descriptions and gene names from the linked SwissProt, TREMBL and Locus Link database entries. This dictionary was then curated in order to remove entries constituting single frequently used normal English words. The resulting non-redundant list contained approximately 68 000 protein aliases and descriptions for 15 000 human proteins.

MedScan preprocessing step was completed in less than 5 h and yielded about 4.6 million sentences containing at least one notation of human protein. To estimate the precision and coverage of the preprocessing step, we visually inspected the results of protein name tagging in sentences from 500 randomly selected MEDLINE abstracts discussing functions of well-studied human proteins related to apoptosis, and determined these parameters to be 97 and 92%, respectively.

The following three parameters of MedScan parsing step have been further estimated: performance (average processing time per sentence), coverage (proportion of MEDLINE sentences which have been successfully parsed and generated correct sentence structure), and ambiguity (number of generated alternative sentence structures per sentence). Performance of MedScan was estimated by parsing 4.6 million sentences containing at least one notation of human protein selected by the preprocessor. The parsing took 23 h (18 ms per sentence) on a 600 MHz Pentium III processor with 128 MB of RAM. This means that MedScan is approximately 20 times faster than similar systems (Friedman *et al.*, 2001), and preliminary evaluation indicates that performance can be further increased by a factor of 3–5 using better implementations of programming components such as more efficient memory management.

Out of 4.6 million sentences only 1.56 million (34%) have been successfully parsed and generated at least one semantic structure. To estimate the correctness of parsing we have analyzed complete sets of alternative semantic structures generated from 168 randomly selected parsed sentences, and determined that in each set the single correct structure was present. We therefore conclude that current MedScan's coverage is 34%. To determine the reasons of parsing failure of the remaining 66% of sentences, we have analyzed 332 randomly selected unparsed sentences. A total of 713 reasons of parsing failures have been identified; categorization and frequencies of each error category are presented in Table 1.

Failure to recognize and correctly assign categories to all words in a sentence accounts for 28.7% of all errors. These errors are caused by the presence of domain-specific concept notations including residue substitutions, chromosome positions, concentrations, cell line names, measurements of various parameters, etc. We have observed, however, that a significant portion of these terms can be described using regular expression formalism and usually constitutes a part of a noun phrase. To solve the problem of domain-specific concept identification, we plan to introduce an additional step of the noun phrase detection and classification in the preprocessing module. It will be based on application of an HMM-based syntactic tagger and specialized finite-state automaton for noun phrase boundary identification in combination with a set of developed regular expressions for recognition of some specific terms.

Lexicon errors constitute the major portion (39.1%) of parsing failures. These include mostly cases of missing lexeme, and, occasionally, incorrect or incomplete assignment of a

Table 1. Classification and frequency of parsing failures (for some types of errors an example is given in parentheses)

Failure category	Failure subcategory	Failure proportion
Sentence structure errors	Incorrect sentence boundary detection	20 (2.8%)
Total: 25 errors (3.5%)	Ill-formed sentence	5 (0.7%)
Word recognition errors	Unrecognized molecule notation (‘Pol III’)	35 (4.9%)
Total: 205 errors (28.7%)	Incorrectly tagged protein name	45 (6.3%)
	Other domain-specific notations (‘Ala → Glu’)	125 (17.5%)
Lexicon errors	Undefined compound lexeme (‘in order to’)	15 (2.1%)
Total: 279 errors (39.1%)	Incorrect or absent lexeme subcategory	136 (19.1%)
	Incorrect or incomplete lexical category assignment	92 (12.9%)
	Other incorrect lexeme feature assignment	15 (2.1%)
	Absence of lexeme in a lexicon	21 (2.9%)
Errors in grammar	Relative phrase construction problems	46 (6.5%)
Total: 204 errors (28.7%)	Unknown domain-specific noun phrase constructs	66 (9.3%)
	Rare and specific type of conjunctions (‘age- and sex-dependent’)	20 (2.8%)
	Unresolved anaphoric expressions	22 (3.1%)
	Over-generation	5 (0.7%)
	Other insignificant grammar inconsistencies	45 (6.3%)

lexeme category, and incorrect or incomplete subcategory assignment. To overcome this problem we plan to increase the size of our lexicon (up to an estimated 50 000 lexemes sufficient for MEDLINE parsing) and also to improve its quality. This lexicon revision will be done based on NLM SPECIALIST (McCray, 1991) lexicon by importing, converting, and manually curating of lexical information (features and lexeme subcategories) for the lexemes most frequently occurring in MEDLINE.

Lastly, 28.7% of parsing failures occurred due to the incomplete grammar. We performed an exhaustive analysis of all cases where the structure of a sentence did not conform to the MedScan grammar. Although the absolute number of such cases is high, the majority of them belong to four well-defined groups (listed in the Table 1), which indeed constitute the ‘gaps’ in the developed grammar; grammar improvement aimed at covering identified gaps is currently underway.

Ambiguity of the syntactic processing is a critical issue in practical applications of NLP systems. Due to the general ambiguity of syntactic knowledge, each sentence usually yields a number (sometimes very large) of potential sentence structures, but only one of them is generally considered correct. Table 2 shows the distribution of the number of generated alternative parses over the number of sentences. The source of ambiguity was investigated on a set of 168 parsed sentences by observing the structure of each alternative parse tree and correlating it with the compositional structure of the corresponding sentence. This analysis revealed that the major sources of ambiguity are variations in prepositional phrase attachment, and structures of coordinate conjunctions. We have designed and preliminarily tested a domain-specific filtering module, for selection of the single correct structure from a set of alternative trees. Briefly, the filters are designed

Table 2. Distribution of number of alternative sentence structures generated by MedScan

Number of alternative structures	Number of sentences	Percentage (%) of sentences
0–10	58	34.5
11–100	45	26.8
101–1000	38	22.6
1001–10 000	20	11.9
10 001–100 000	3	1.8
>100 000	4	2.4
Total	168	100

in a form of ontology, which can be viewed as a set of conceptual frames (each with a set of concept-specific slots) and a set of ontological links [in a form of (Frame, Slot -> Frame) triplets] specifying other frames as admissible arguments of each frame’s slot. Each lexeme can be assigned an ontological frame representing its domain-specific meaning, and thus every node of a semantic tree can be mapped onto an ontological frame through a corresponding lexeme. A special mechanism also maps the slots of the semantic node onto the slots of the ontological frame for each particular lexeme. Therefore, each branch of a semantic tree can be viewed as a potential ontological link, and can be evaluated for being defined in ontology. The semantic tree is considered valid if all of its branches have equivalent ontological links defined. In our preliminary experiments, the single correct structure was selected from sentences with even more than 100 000 of alternative ones. However, the development of the extraction module is still an ongoing research project and formal results will be published in detail elsewhere.

In summary, we have developed and evaluated MedScan, a general purpose NLP component for analysis of biomedical literature. From the results of the MedScan evaluation we can conclude that its performance is satisfactory for the real-time MEDLINE processing. The current MedScan's coverage rate is 34%, but we estimate that by increasing the lexicon size, improving its quality, and by slightly improving its grammar, MedScan's coverage can be increased up to 90%. In addition, even with a 34% coverage MedScan is still immediately applicable for an information extraction task, since the most reliable information in MEDLINE is frequently reiterated and is unlikely to be missed. We are currently testing an algorithmic approach, which utilizes the full-sentence semantic structures generated by MedScan to extract a protein function information with high (above 90%) precision. The details of its design and implementation will be described elsewhere. It is important to note, that the context-free nature of MedScan makes it applicable to a large number of areas ranging from pathway analysis to clinical informatics and protein structure–function relationships.

REFERENCES

Allen,J. (1994) *Natural Language Understanding*. Benjamin-Cummings Publishing Company, New York.

Applet,D., Hobbs,J., Bear,J., Israel,D., Kameyaqma,M., Kehler,A., Martin,D., Myers,K. and Tyson,M. (1995) SRI International FASTUS system: Muc-6 Test Results and Analysis. *Proceedings of the Sixth Message Understanding Conference*, pp. 237–248.

Blaschke,C., Andrade,M.A., Ouzounis,C. and Valencia,A. (1999) Automatic extraction of biological information from scientific text: protein–protein interactions. *Ismb*, 60–67.

Friedman,C., Alderson,P.O., Austin,J.H., Cimino,J.J. and Johnson,S.B. (1994) A general natural–language text processor for clinical radiology. *J. Am. Med. Inform. Assoc.*, **1**, 161–174.

Friedman,C., Kra,P., Yu,H., Krauthammer,M. and Rzhetsky,A. (2001) GENIES: a natural language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, **17** (Suppl. 1), S74–S82.

Humprhreys,K., Gaizauskas,R., Azzam,S., Huyck,C., Mitchell,B., Cuningham,H. and Wilks,Y. (1998) Description of the LaSIE-II System as used for MUC-7. *Proceedings of the Seventh Message Understanding Conference*.

Humphreys,K., Demetriou,G. and Gaizauskas,R. (2000) Two applications of information extraction to biological science journal articles: enzyme interactions and protein structures. *Pac. Symp. Biocomput.*, 505–516.

Kaplan,R. and Bresnan,J. (1982) Lexical-functional grammar: a formal system for grammatical representation. In Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pp. 173–281.

McCray,A.T. (1991) Extending a natural language parser with UMLS knowledge. *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*, pp. 194–198.

Ono,T., Hishikagi,H., Tanigami,A. and Takagi,T. (2001) Automated extraction of information on protein–protein interactions from the biological literature. *Bioinformatics*, **17**, 155–161.

Park,J.C., Kim,H.S. and Kim,J.J. (2001) Bidirectional incremental parsing for automatic pathway identification with combinatory categorical grammar. *Pac. Symp. Biocomput.*, **6**, 396–407.

See-Kiong,N. and Wong,M. (1999) Toward routine automatic pathway discovery from on-line scientific text abstracts. *Genome Informatics*, **10**, 104–112.

Sekimizu,T., Park,H.S. and Tsujii,J. (1998) Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. *Genome Informatics*, **9**, 62–71.

Sells,P. (1985) *Lectures on Contemporary Syntactic Theories*. C S L I Publications.

Thomas,J., Milward,D., Ouzounis,C.A., Pulman,S. and Caroll,M. (2000) Automatic extraction of protein interactions from scientific abstracts. *Pac. Symp. Biocomput.*, 541–552.

Yakushiji,A., Tateisi,Y., Miyao,Y. and Tsujii,J. (2001) Event extraction from biomedical papers using a full parser. *Pac. Symp. Biocomput.*, **6**, 408–419.