GeneTUC, GENIA and Google: Natural Language Understanding in Molecular Biology Literature

Rune Sætre[1], Harald Søvik[1], Tore Amble[1], and Yoshimasa Tsuruoka[2]

[1] Department of Computer and Information Science,
Norwegian University of Science and Technology,
Sem Sælandsv. 7-9, NO-7491 Trondheim, Norway,
Rune.Satre@idi.ntnu.no,
http://www.idi.ntnu.no/~satre
[2] Department of Computer Science, University of Tokyo,
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan

Abstract. With the increasing amount of biomedical literature, there is a need for automatic extraction of information to support biomedical researchers. GeneTUC has been developed to be able to read biological texts and answer questions about them afterwards. The knowledge base of the system is constructed by parsing MEDLINE abstracts or other online text strings retrieved by the Google API. When the system encounters words that are not in the dictionary, the Google API can be used to automatically determine the semantic class of the word and add it to the dictionary. The performance of the GeneTUC parser was tested and compared to the manually tagged GENIA corpus with EvalB, giving bracketing precision and recall scores of 70,6% and 53,9% respectively. GeneTUC was able to parse 60,2% of the sentences, and the POS-tagging accuracy was 86.0%. This is not as high as the best taggers and parsers available, but GeneTUC is also capable of doing deep reasoning, like anaphora resolution and question answering, which is not a part of the state-of-the-art parsers.[3]

Keywords: Biomedical Literature Data Mining, Google API, GENIA

1 Introduction

Modern research is presenting more new and exciting results than ever before, and it is gradually becoming impossible for the human reader to stay up-to-date in the sea of information. This is especially true in the Medical and Molecular Biology domains, where the MEDLINE database of publications is increasing with almost 2000 new entries every day. To help researchers find the information they are searching for in an efficient manner, automatic Information Extraction

[3] C. Priami et al. (Eds.): Trans. on Comput. Syst. Biol. V, LNBI 4070, pp. 68-82, 2006.

(IE) is needed. This paper describes a system that is using Natural Language Processing (NLP) in order to automatically read the abstracts of research papers, and later answer questions posed in English about the abstracts.

1.1 Information Extraction (IE) in Biology

The large and rapidly growing amounts of biomedical literature demands a more *automatic* extraction process than previously. Current extraction approaches have provided promising results, but they are not sufficiently accurate and scalable. A survey describing different approaches within the *information extraction field* is presented in [6], and a more recent "IE in Biology" survey is given in [15]. In the biomedical domain, IE approaches range from simple automatic methods to more sophisticated but also more manual methods. Some good examples are: Learning relationships between proteins/genes based on co-occurrences in MEDLINE abstracts [9], *manually* developed IE rules [24], protein name classifiers trained on *manually* annotated training corpora [1], and classifiers trained on *automatically* annotated training corpora [20].

A new emerging approach to medical IE is the heavy use of corpora. The workload can then be shifted from the extremely time consuming manual grammar construction to the somewhat easier and more teamwork oriented corpus/treebank building [12]. This means that the information acquisition bottleneck can be overcome, while still reaching state-of-the-art coverage scores (around 70-80 percent). In this chapter a corpus is used, namely the GENIA Tree Bank (GTB) corpus [19], first to train and then later to test how well the GeneTUC parser performs compared to other available parsers in this domain.

1.2 GeneTUC

The application that we want to improve and test, by incorporating alternative sources of information, is called GeneTUC. TUC is short for "The Understanding Computer", and is a system that is under continuous development at the Norwegian University of Science and Technology. Section 3 will explain in more detail how TUC, and especially GeneTUC, works.

1.3 Chapter Structure

The rest of this chapter is organized as follows. Section 2 describes the materials and programs that were used, section 3 explains in detail how GeneTUC works, section 4 presents our approach, section 5 presents the empirical results, section 6 describes other related work, section 7 contains a discussion of the results, and finally the conclusion and future work are presented in section 8.

2 Materials

One of the main goals was to test how good the current state of the GeneTUC parser is. To do this, some manually inspected parsed text is needed, and that is

exactly what the new syntactically enhanced GENIA corpus is [19]. It consists of text from MEDLINE (see subsection 2.1), and provides a gold standard that can be used both for training and testing the GeneTUC application. See Subsection 2.2 for more details.

2.1 MEDLINE

Medline[4] is an online collection of more than 14 million abstracts by now (November 2005). The abstracts are collected from a set of different medical journals by the US National Institutes of Health (NIH). NIH grants academic licenses for PubMed/MEDLINE for free to anyone interested. When it was downloaded in September 2004, the academic package included a local copy of 6.7 million abstracts, out of the 12.6 million entries that were available on their web interface at that time.

2.2 GENIA Tree Bank (GTB)

It was decided to use the GENIA Tree-Bank (GTB) corpus[5] for training and testing of GeneTUC. GTB consists of 500 abstracts from the GENIA corpus which consists of 2000 abstracts from MEDLINE. These 500 abstracts have been parsed, manually inspected and corrected to ensure that they contain the expected parse result for every single sentence. The format of the annotation is a slightly modified Penn Tree Bank-II format. The GTB is split into GTB200 with 200 abstracts and GTB300 with 300 abstracts. We used GTB300 as a training set, and GTB200 as test set to calculate the precision and recall scores for parsing of *unseen* text. It should be pointed out that GTB is still in a beta-release state, which means that it still contains some errors, and some manual inspection of the results are needed to determine if this has a great influence on the measured numbers.

A list of all composite terms in the GTB was also used as input to the system. This was done to ensure that the parsing performance was measured without being influenced by bad tokenization, which is handled by another module, namely the lexical analysis module, in GeneTUC.

3 GeneTUC

GeneTUC is on the way to be a full-fledged Question Answering system, but the coverage is still low. Figure 1 shows the general information flow in the TUC systems. The input sentence can be either a fact for example from a Medline abstract or a question from the user. The analysis is the same in both cases, but the answer will have two different forms. In the case of a factual input sentence, the facts are coded in a first order event logic form called TUC Query Language

[4] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi
[5] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/topics/Corpus/GTB.html

(TQL) and then stored in the Knowledge Base (KB) of the system. This is shown in Example 1, above the line. Later, when someone inputs a question, the question will also first be coded using TQL, but either the subject or one of the objects in the sentence may then be wildcards that should be matched against facts in the existing KB.

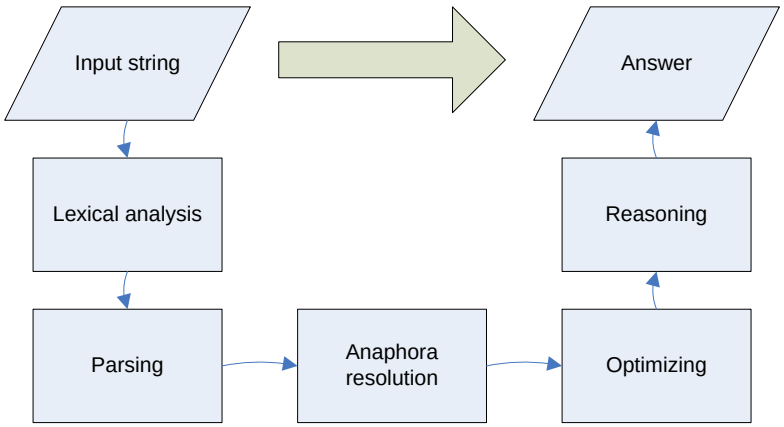

Fig. 1. Data flow in the TUC System

Example 1.

Statement : "CCK activates Gastrin."
Update to KB : activate(cck,gastrin).
Question : "What activates Gastrin?"
Answer : "CCK"

In this case it is very obvious that "What" is the placeholder for the answer, and also that it must be substituted with "CCK" to match the existing fact. So even if this is a very simple example, the method works in the same way also for more complex sentences. The only requirement is that the question is stated in a similar grammatical form as the factual statement.

3.1 GeneTUC Lexical Analysis

The lexical analysis in GeneTUC changes the input sentences from a long list of characters into tokens (words) and sentence delimiters. The current set of sentence delimiters includes the following:

Period	Colon	Semi Colon	Question Mark	Exclamation Mark
.	:	;	?	!

In the process of making the tokens, no distinction is made between upper and lower case characters, so the input to the syntactical analysis is a set of all lower case tokens.

3.2 GeneTUC Grammar and Syntactical Analysis

The GeneTUC grammar is what we call ConSensiCAL. That means it is a Context Sensitive Categorial Attribute Logic grammar formalism. It is based on the Prolog Definite Clause Grammar (DCG) with a few extensions to handle categorial movement and gaps etc. See [5] for more details on the Prolog programming language for Natural Language Processing (NLP)

Categorial Grammar TUC is inspired by Categorial Grammar which allows *gaps* in the sentence. This mechanism is very easy to use when parsing sentences like in the following examples:

Example 2.

Input: What activates Gastrin?

Grammar for Question, using Statement:

Statement → NounPhrase VerbPhrase
Question → *what* Statement\NounPhrase
VerbPhrase → Verb NounPhrase
...

Example 3.

Input:
Results of preliminary studies, which we have conducted,
suggest that use of this agent is useful.

Grammar (Forward Application):

NounPhrase → Det Nominal RelativeClause
RelativeClause → RelativePron Statement/NounPhrase
RelativePronoun → *that*|*which*|*who*
...

Example 4.

Input: A gene signal that results in production of proteins occurs.

Grammar (Backward Application):

Statement → NounPhrase RelativeClause
RelativeClause → RelativePronoun Statement\NounPhrase
...

Example 5.

Input: A gene signal resulting in protein production occurs.

Grammar for Gerund:

RelativeClause → Verb-*ing* RelativeClause*that*Verb-*s*
...

Example 2 shows how the *What-Question* from Example 1 can be parsed using the existing grammar rules for Statement. It states that a *"what-question"* consists of the word "what" followed by a *Statement*, which is missing the leading *Noun Phrase (NP)*. This kind of Categorial *movement* makes it possible to

connect the missing NP in the question ("what") with the actual NP in a corresponding fact statement ("CCK"), and then give a correct answer to the natural language query. This use of Backward (\) and Forward (/) application also reduces the number of grammar rules needed, since every new rule for statements implicitly creates corresponding new rules for questions.

In Example 3 the use of Forward application is shown. In GeneTUC, Forward application also includes Inward application, so "S/NP" also means that the NP can be missing anywhere in the Statement.

In Example 4, Backward application is used to define a Relative Clause. The missing NP in the Relative Clause can be found by going back to the NP that is preceding the Relative Clause.

Example 5 shows a different form of the sentence from Example 4. With the help of Backward application only one rule is needed to change this *gerund* sentence into a RelativeClause that can then be parsed by the grammar given in Example 3. This rule is context sensitive, meaning that *ing*-verbs like "resulting" can only be substituted by "that verb-*s*" phrases, like "that results", when the parser is already expecting to see a RelativeClause.

3.3 Reducing the Parsing Time

In GeneTUC parsing time is reduced by the use of *cuts* in the Prolog code. This means that once a specific rule, for example *Noun Phrase (NP)*, has been successfully applied to a part of the input sentence, this part of the sentence is *committed* and can not be parsed again even if the following rules causes the parser to fail. Usually, failing on one possible parsing attempt would cause the parser to back-track and use the rule on a different span of words to produce a different and successful NP. This kind of backtracking can be very computationally expensive, especially with highly ambiguous input, so *cuts* greatly reduces the parse time. The cuts are placed manually in strategic places in the code, based on experience from previous parsing of *run-away* sentences. Usually, the cuts do not affect the final result from the parser, but some phenomena can cause the parser to fail because the assumed partial parsing result before the cut is incompatible with the rest of the sentence. One such phenomenon, which is hard to parse even for humans, is *garden path sentences* [13].

4 Methods

The main goal of this research was to evaluate the GeneTUC parser on the GENIA corpus. Since GeneTUC and GENIA were not made using any common grammar standard, a lot of modifications in GeneTUC were needed. These adaptations can be thought of as a (manual, not statistical) training process for GeneTUC, but in order to measure how the GeneTUC parser will perform on unseen data, different parts of the GENIA Tree Bank (GTB) was used for training and testing, i.e. we used 300 abstracts (GTB300) for training and the remaining 200 abstracts (GTB200) for testing.

The training phase of the project is described in the following subsections, and includes the following tasks:

- Dictionary building. Adding all terms from GENIA to the GeneTUC dictionary.
- Ontology building. Mapping from GENIA to GeneTUC dictionary classes.
- Adding other missing words manually, with the help of Bioogle.
- Input new verb templates, based on predicate argument structures seen in GENIA.

4.1 Updating GeneTUC lexicon from GENIA

Since the goal is to test the parser, errors connected to the Tokenizer, POS tagger or other parts of the system should be removed. The ideal approach would be to use the tokenized and POS-tagged version of GENIA as input to GeneTUC, but this was not feasible since GeneTUC is based on plain ASCII-text input. Also, it would take more manpower/time than available in this project to split the tight connection between tokenizing, tagging and parsing in TUC, just to test if it would be useful to do so later. Instead, the GENIA multi-word-terms were added to the GeneTUC dictionary, trying to guide it into using the same tokenization as in the GENIA gold file. This was only successful in around 20% of the sentences, so we reduced the test set to only include sentence that were similarly tokenized and tagged by GENIA and GeneTUC.

During the process of importing all the terms from GENIA into the TUC, several considerations had to be made:

1. *Plural Forms.* Plural words were changed into their singular (stem) form by simple rules like: remove the final s from all words. Exceptions to this simple rule had to be made for words like virus (already singular), viruses (remove -es) and bodies (change -ies to y).
2. *Proper Names or Common Nouns.* Another point is that plural forms should exist only as ako[6] relations (class concepts), and *not* as isa[7] relations (proper names).
3. *Duplicate Entries.* Changing plural forms into singular forms often leads to duplicate entries in the dictionary, since the singular form
4. *Short Ambiguous Terms.* The title sentence "Cloning of ARE-containing genes by AU-motif-directed display" causes a problem since TUC does not distinguish "ARE" and "are". Words like "are", "is", "a" and so on are therefore removed from the dictionary.

4.2 Updating the GeneTUC Semantic Network

As mentioned in the introduction, GeneTUC is a deep parser, requiring that all the input words are already in the dictionary. In order to compare just the

[6] ako = A Kind Of (subclass of a class)
[7] isa = Is A (instance of a class)

parsing performance of GeneTUC with other systems, other error sources such as incomplete lexical tagging was reduced by importing all named entities from GENIA to GeneTUC. When new words are added to GeneTUC, it is also necessary to specify which semantic class they belong to, so a mapping between GENIA ontology and the ontology of GeneTUC was needed (Figure 2). One alternative way was to simply add all the ontology terms of GENIA (37) to GeneTUC, but many of the terms were already present in both systems, with slightly different classifications. We could also have changed the GeneTUC ontology terms to match those of GENIA, but that would have made many of the existing verb templates in GeneTUC useless or wrong, making this approach unappealing. The final choice was to create a mapping from GENIA ontology terms to existing GeneTUC ontology terms, as shown in Figure GENIA ontology. The GENIA term "other_name" and the corresponding GeneTUC term "stuff" are "bag" definitions, meaning that no effort was made to distinguish the terms that did not belong to one of the other classes. Many of these terms can easily be put into other existing GeneTUC classes, just by matching the last noun in the noun phrases as in the following example:

Example 6.

"nf_kappa_b_activation" *ako* activation
'0_95_kb_id_3_transcript' *ako* transcript
'17_amino_acid_epitope' *ako* epitope
asp_to_asn_substitution *ako* substitution

4.3 Adapting TUC to GTB/PTB Syntax Standard

Since we wanted to use the GENIA Tree Bank (GTB) for evaluating the GeneTUC parser, we needed to make sure that the output from the GeneTUC parser was in the same format as the parse trees from the GTB. This is a somewhat complicated process, since the TUC parser uses an internal syntax representation that is tightly connected to the semantics of the sentence, and this representation is different from the GTB syntax in a few important aspects. For example, the Categorial movement and gap mechanisms are implemented in TUC by doing parsing and reparsing. That means that the sentence will be parsed once, and then gaps will be filled with the syntax from the first parse, before the new resulting sentence is parsed again. This means that traces of the moved phrases will appear both where the phrase was originally, and where the gap was in the resulting parse tree. This leads to parse trees that look slightly different from the GTB parse trees, where each gap is given an Identifier (ID) and then the corresponding syntactical phrase is given the same ID-number. As long as no effort is made to implement this gap-ID system of the GTB grammar in TUC, these differences will lead to lower accuracy values in the evaluation, even though the parsing result is actually correct. To prevent this from happening, the internal workings of TUC had to be modified to produce output exactly equal to the expected output, and some pre- and post-processing scripts had to be made in order to smooth out the remaining systematical differences that could not be changed inside TUC. Still, some traces of these problems may be left in the final evaluation scores.

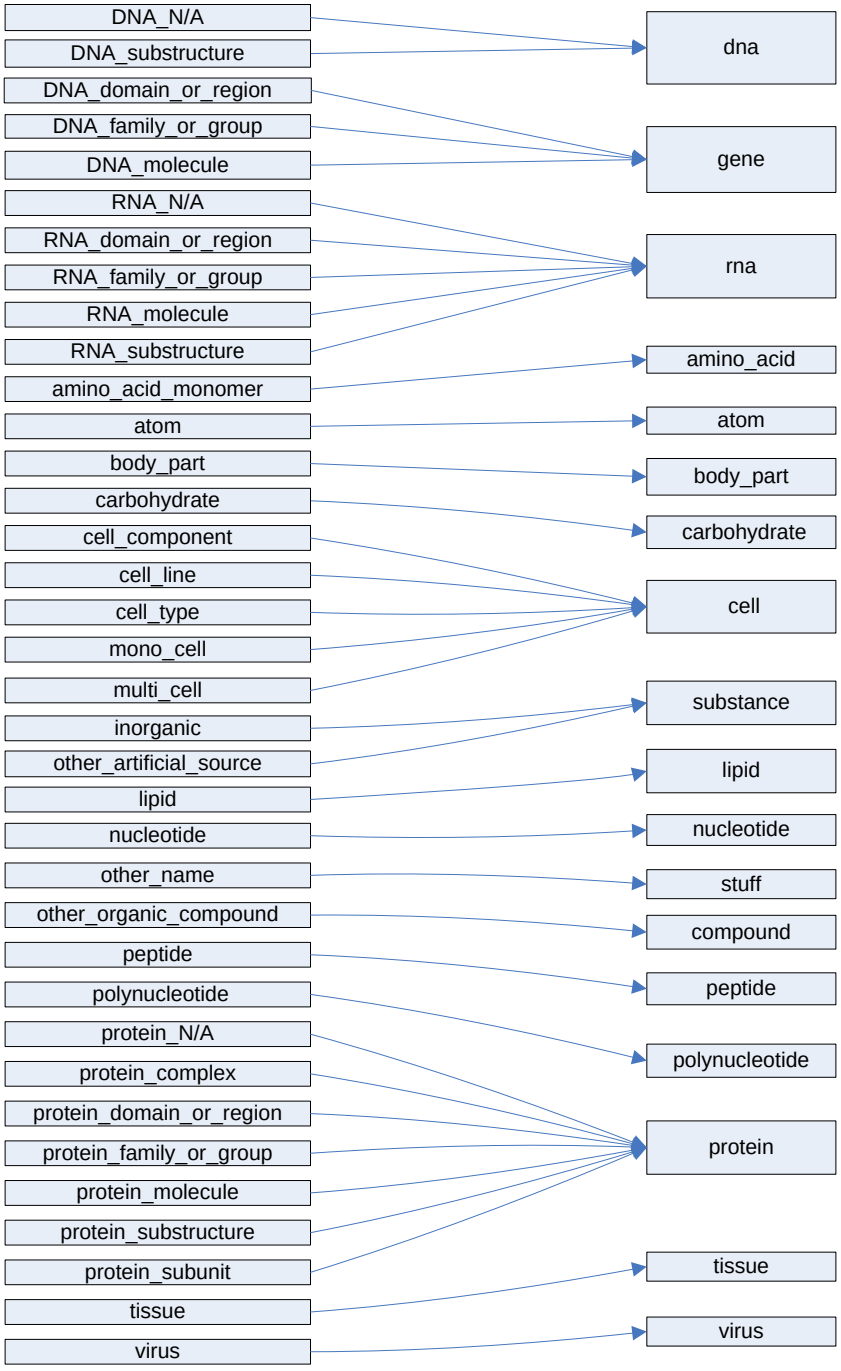

Fig. 2. Conversion from GENIA to GeneTUC Ontology

The creation of the grammar is currently done 100% manually. It is a slow and long-lasting job, but it ensures that all the rules are meaningful. The creation of a new rule is always rooted in the existence of old rules, as was shown in Examples 2 & 3.

4.4 EvalB and Tokenization

EvalB[8] was used for calculation of precision and recall scores for GeneTUC against the GENIA corpus. It requires that the number of tokens in the output text has to match the number of tokens in the input/gold text exactly. This is a challenge to GeneTUC, since it ignores the characters listed in Example 7.

Example 7. Ignored Characters: " : , & % { } ⟨ ⟩ [ ] (. . .)

Also, single tokens (like IL-2) are sometimes turned into two or three separate tokens ("il", "-" and "2"), because of hyphens. This happens when the word is not specifically defined in the dictionary as being just one word/token. Since the GTB is already tokenized and stored in XML format, the correct tokenization is known. The challenge is to ensure that TUC produces the expected output, even if the internal modules are using different tokens. Other features of GeneTUC that makes the comparison difficult is that some *noisewords* are removed from the text, and long Noun Phrases can sometimes be substituted with Canonical Identifiers.

There are two obvious solutions to this problem: The first is to use the tokenized version of GTB, instead of the plaintext version. The problem then, is that we have to circumvent the tokenization module (and lexical analysis) in TUC, and this might introduce problems in the later modules, for example because of ignored characters that were previously handled by the lexical module. Another example of problems introduced if the original tokenization is used, is parentheses with their contents. In the current implementation all level-1 parentheses are removed together with everything inside them, since this is usually ungrammatical constructions.

The second solution to the tokenization problem is to make a new plaintext version of the text, from the tokenized xml-version. In the new plain text version, all tokens containing hyphens and other troublesome characters, will be substituted by a new token using underscore (_) or some other character instead of hyphen, so that the lexical module does not split these token into extra tokens. In the opposite case, where the gold text contains more tokens than what TUC produces, we have to introduce some dummy tokens. These tokens can then act as placeholders for tokens that TUC ignores (and removes), like parentheses with all their content/tokens.

4.5 EvalB comparing Syntax Trees

Using the tokens in the sentence as basis for scoring, EvalB performs a strict evaluation. Any case of tokenization different from the "golden" tokenization

[8] http://nlp.cs.nyu.edu/evalb/

renders the parse incomparable; the sentences where the number of golden tokens and test tokens are unequal lead to an *error*. The same is true for those where the golden token and test token are character wise different. If the number of tokens equals zero (i.e. the sentence did not parse successfully), the sentence is *skipped*. Both *skip*- and *error*-sentences are ignored when calculating the score. The program provides a tolerance limit for how many incomparable sentences that are ignored.

Bracketing is measured from token[m] to token[n], where a *match* means those brackets covering the correct tokens, and having a correct *label*. The matches enable measurements of:

- Recall (the ratio between matched brackets and all brackets in the gold file)
- Precision (the ratio between matching brackets and all brackets in the test file)
- Crossing (the number of test-file-brackets exceeding the span of a matching bracket in the gold file, divided by the total number of brackets in test file)
- Tagging accuracy (the ratio of correct labeled tokens over the total number of tokens)

EvalB performs strict evaluation of the parse, as it originally was intended as a solemn bracketing evaluation program. Bracketing scores of GeneTUC may be reduced because of a right-orientation implied by the grammar of TUC (always preferring right-attachment unless it is semantically erroneous).

5 Results

This section shows the results from the training and test phases. Table 1 shows how much the performance of GeneTUC increased when the dictionary was extended with all the terms from GENIA. Table 2 shows that there was no significant difference in parsing scores between importing all the GENIA terms (36.692) or just the terms from GTB200 that were reported as unknown by GeneTUC (8.175). In terms of input to EvalB, it was possible to compare almost twice as many sentences when only the GTB200 dictionary was used. This is mainly because GeneTUC was given fewer chances to rewrite complex multi-word tokens, and thereby creating better accordance between the output and the gold file.

6 Related Work

Other recent and related IE techniques for biomedical literature includes systems using dynamic programming [8] or supervised machine learning [22] to find protein-protein-relations in molecular biology texts. The machine learning approach uses both parse trees and dependent tree structures as features, but they report that simple lexical features contribute more to the promising F-measure of 90.9. Other systems use predicate-argument structure patterns [23] or new self made architecture [21] to do more general Information Extraction from this kind of free text sources.

Table 1. Statistics parsing attempts before and after adding GENIA dictionary

Measurement	**Dictionary**	
Description	**Original**	**+GENIA**
Number of sentences:	2591	2591
Successful parses:	318	1531
Success rate:	12.3%	59.1%
Sources of Failure		
Dictionary:	1989	68
Grammar:	520	1126
Time out:	32	144
Processing time:	0.5 hrs	4.5 hrs

Table 2. Test results from EvalB

Measurement	**Dictionary**	
Description	**+GENIA**	**+GTB200**
Number of sentences	1759	1759
Error sentences	518	565
Skip sentences	1037	843
Valid sentences	204	351
Bracketing Recall	49.8%	53.9%
Bracketing Precision	69.0%	70.6%
Complete match	0.49%	1.14%
Average crossing	1.27	1.47
No crossing	47.1%	44.7%
2 or less crossing	79.9%	79.5%
Tagging accuracy	82.0%	86.0%

6.1 GeneTUC, Bioogle and GProt

This paper showed how important a proper dictionary is to this kind of semantic parsers. A new way to automatically build dictionaries with ontology information is presented as Bioogle in [18]. Bioogle[9] is a simple system that uses Google to determine the semantic class of a word, for example "CCK is a protein", so that it can be added to the semantic hierarchy (or dictionary) in a correct way.

GProt [17], is another application that is built on top of the Google API, like Bioogle. GProt[10] provides a way of automatically extracting information from the (biomedical research) literature. Most of the literature is already indexed in MEDLINE, and therefore also by Google and other major search engines. See [16] for more details and a description of how to access the online versions of Bioogle and GProt.

[9] http://furu.idi.ntnu.no:8080/bioogle/
[10] http://furu.idi.ntnu.no:8080/gprot/

7 Discussion

This section points out some of the lessons learned during the parsing project. This includes remarks about titles as a different sentence type and a discussion about the results presented in the two previous sections.

7.1 Sentence Types

MEDLINE (GTB) contains two fundamentally different sentence types: Titles and normal sentences. The titles are special, because they sometimes just state the object of the experiment, without the subject and verb phrase that should have started the sentence. Subject and verb-less sentences were already handled by GeneTUC before, but during this work we added a special "\title"-tag for the titles, so that we can implement some special processing of titles later. The first function we added to the "\title"-tag was resetting the temporary anaphoric database, so that terms like "the protein", "this" and "which" do not map to names or events in any previously parsed abstracts.

7.2 Comparing Different Systems

It turned out that evaluating the GeneTUC parser on a PTB gold standard file was harder than first expected. The main reason for this is that TUC was never meant to output PTB style tags in the first place. Also, there is not always a clear boundary between lexical, syntactical and semantic analysis. Of course, there are both advantages and disadvantages to this approach.

The problem that we encountered because of the tight connection between the modules in GeneTUC, is that it is very hard to construct output with the exact number of tokens as in the input text. TUC is based on receiving plain text input, and does its own tokenization and optimization of the text before passing it on to the parser. We could perhaps have used the already tokenized text as input, but this would introduce the parser to problems it is not meant to handle in the current configuration of the system. It would be much easier to cooperate with other researchers if the modules of GeneTUC were truly separate from each other, but it can also be argued that the good performance by a text processing system like this is really dependent on tight communication between the modules.

Tokenization is usually done before, and separate from, parsing, but sometimes it is necessary to do preliminary parsing in order to determine word and sentence boundaries. Parsing is usually done before semantic analysis, but sometimes it is important to know the semantic properties of a word in order to reduce the ambiguity, and thereby the parsing time. Maybe the time has come to start integrating the different modules more? This will require some effort to ensure that cooperation between different researchers is still possible, for example through the use of new standards/protocols for future parsers.

8 Conclusion

There is a great need for systems that can support biologists (and any other research) in dealing with the ever increasing information overload in their field. This project has proven that both the Google API and the GeneTUC systems are important pieces that can play a role in making the dream of real automatic Information Extraction come true in the not so distant future.

The precision and recall scores achieved by GeneTUC on general parsing are not very high compared to pure parsers like [4, 10, 7], but that does not mean that these systems are better than GeneTUC, because GeneTUC also performs deeper analysis such as anaphora resolution [2, 14]. The other systems consist of Context-Free Grammar (CFG) parsers that give only phrase structures as output. There are also some systems that use CCG parsers [3] or HPSG parsers [11] that can give predicate argument structures in addition to phrase structures, but they still do not perform anaphora resolution or question-answering, like GeneTUC does.

Appendix A

Acknowledgements

The first author would like to thank all the people who made the writing of this chapter possible. Especially, Professor Tsujii who invited me to his lab in Tokyo, and all his brilliant co-workers who helped me with anything related to the GENIA corpus. Much gratitude also goes to my co-supervisors Amund Tveit and Astrid Lægreid for continuous support, and for pointing out good opportunities.

References

1. Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, and Yuk Wah Wong. Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. *Journal Artificial Intelligence in Medicine: Special Issue on Summarization and Information Extraction from Medical Documents*, 2004.
2. J. Castano, J. Zhang, and J. Pustejovsky. Anaphora resolution in biomedical literature. In *International Symposium on Reference Resolution*, 2002.
3. Stephen Clark, Julia Hockenmaier, and Mark Steedman. Building Deep Dependency Structures with a Wide-Coverage CCG Parser. In *Proceedings of ACL'02*, pages 327–334, 2002.
4. Andrew B. Clegg and Adrian J. Shepherd. Evaluating and integrating treebank parsers on a biomedical corpus. In *Proceedings of the ACL Workshop on Software 2005*, 2005.
5. Michael A. Covington. *Natural Language Processing for Prolog Programmers*. Prentice-Hall, Englewood Cliffs, New Jersey, 1994.
6. J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

7. Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. Adapting a probabilistic disambiguation model of an HPSG parser to a new domain. In *IJCNLP 2005: Second International Joint Conference on Natural Language Processing*, 2005.
8. Minlie Huang, Xiaoyan Zhu, Yu Hao, Donald G. Payan, Kunbin Qu, and Ming Li. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612, Dec 12 2004.
9. Tor-Kristian Jenssen, Astrid Lægreid, Jan Komorowski, and Eivind Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28(1):21–28, May 2001.
10. Matthew Lease and Eugene Charniak. Parsing biomedical literature. In *Second International Joint Conference on Natural Language Processing (IJCNLP'05)*, 2005.
11. Yusuke Miyao and Jun'ichi Tsujii. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of COLING 2004*, pages 1392–1397, 2004.
12. R. O'Donovan, M. Burkea, A. Cahill, J. van Genabith, and A. Way. Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank. In *Proceedings of the 42nd Annual Meeting of the ACL.)*, pages 368–375, Barcelona, Spain, July 21-26 2004. Association for Computational Linguistics.
13. Lee Osterhout, Phillip J. Holcomb, and David A. Swinney. Brain potentials elicited by garden-path sentences: Evidence of the application of verb information during parsing. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(4):786–803, 1994.
14. J. Pustejovsky, J. Casta, J. Zhang, B. Cochran, and M. Kotecki. Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Pacific Symposium on Biocomputing*, 2002.
15. Rune Sætre. Natural Language Processing of Gene Information. Master's thesis, Norwegian University of Science and Technology, Norway and CIS/LMU München, Germany, April 2003.
16. Rune Sætre, Martin T. Ranang, Tonje S. Steigedal, Kamilla Stunes, Kristine Misund, Liv Thommesen, and Astrid Lægreid. Webprot: Online Mining and Annotation of Biomedical Literature using Google. In Tuan D. Pham, Hong Yan, and Denis I. Crane, editors, *Advanced Computational Methods for Biocomputing and Bioimaging*. Nova Science Publishers, New York, USA, 2006.
17. Rune Sætre, Amund Tveit, Martin Thorsen Ranang, Tonje Strømmen Steigedal, Liv Thommesen, Kamilla Stunes, and Astrid Lægreid. GProt: Annotating Protein Interactions Using Google and Gene Ontology. In *Lecture Notes in Computer Science: Proceedings of the Knowledge Based Intelligent Information and Engineering Systems (KES2005)*, volume 3683, pages 1195 – 1203, Melbourne, Australia, August 2005. KES 2005, Springer.
18. Rune Sætre, Amund Tveit, Tonje Strømmen Steigedal, and Astrid Lægreid. Semantic Annotation of Biomedical Literature using Google. In Dr. Osvaldo Gervasi, Dr. Marina Gavrilova, Dr. Youngsong Mun, Dr. David Taniar, Dr. Kenneth Tan, and Dr. Vipin Kumar, editors, *Proceedings of the International Workshop on Data Mining and Bioinformatics (DMBIO 2005)*, volume 3482 (Part III) of *Lecture Notes in Computer Science (LNCS)*, pages 327–337, Singapore, May 9-12 2005. Springer-Verlag Heidelberg.
19. Yuka Tateishi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. Syntax Annotation for the GENIA corpus. In *Proceedings of the IJCNLP 2005*, Korea, October 2005.

20. Amund Tveit, Rune Sætre, Tonje S. Steigedal, and Astrid Lægreid. ProtChew: Automatic Extraction of Protein Names from Biomedical Literature. In *Proceedings of the International Workshop on Biomedical Data Engineering (BMDE 2005, in conjunction with ICDE 2005)*, pages 1161–1161, Tokyo, Japan, April 2005. IEEE Press (Electronic Publication).
21. Aditya Vailaya, Peter Bluvas, Robert Kincaid, Allan Kuchinsky, Michael Creech, and Annette Adler. An architecture for biological information extraction and representation. *Bioinformatics*, 21(4):430–438, 2005.
22. Juan Xiao, Jian Su, and GuoDong Zhouand ChewLim Tan. Protein-protein interaction extraction: A supervised learning approach. In *Semantic Mining in Biomedicine (SMBM)*, 2005.
23. Akane Yakushiji, Yusuke Miyao, Yuka Tateishi, and Junichi Tsujii. Biomedical information extraction with predicate-argument structure patterns. In *Semantic Mining in Biomedicine (SMBM)*, 2005.
24. Hong Yu, Vasileios Hatzivassiloglou, Carol Friedman, Andrey Rzhetsky, and W. John Wilbur. Automatic Extraction of Gene and Protein Synonyms from MEDLINE and Journal Articles. In *Proceedings of the AMIA Symposium 2002*, pages 919–923, 2002.