**NTNU**
Innovation and Creativity

# GeneTUC
Event extraction from TQL logic

**Harald Søvik**

**Master of Science in Computer Science**
Submission date: June 2006
Supervisor: Tore Amble, IDI
Co-supervisor: Rune Sætre, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

# Problem Description

As NLP-applications becomes able to extract huge amounts of knowledge from articles, precise measures for their quality is needed. The goals of this project is (1) to investigate a number of annotated corpuses to determine which is most appropriate for benchmarking of GeneTUC, and (2) convert as much as possible from GeneTUCs acquired knowledge to this (foreign) format, and produce an (automated) evaluation of the resulting data.

Assignment given: 20. January 2006
Supervisor: Tore Amble, IDI

# Summary

As Natural Language Processing systems converge on a high percentage of successful deeply parsed text, parse success alone is an incomplete measure of the "intelligence" exhibited by the system. Because systems apply different grammars, dictionaries and programming languages, the internal representation of parsed text is often different from system to system, making it difficult to compare performance and exchange useful data such as tagged corpora or semantic interpretations.

This report describes how semantically annotated corpora can be used to measure quality of Natural Language Processing systems. A selected corpus produced by the GENIA project were used as "golden standard" (event-annotated abstracts from MEDLINE). This corpus were sparse (19 abstracts), thus manual methods were employed to produce a mapping from the native GeneTUC knowledge format (TQL). This mapping were used to produce an evaluation of events in GeneTUC. We were able to attain a recall of 67% and average precision of 33% on the training data. These results suggest that the mapping is inadequate. On test data, the recall were 28% and average precision 21%.

Because events is a new "feature" in NLP-applications, there are no large corpora that can be used for automated rule learning. The conclusion is that at least there exists a partial mapping from TQL to GENIA events, and that larger corpora and AI-methods should be applied to refine the mapping rules. In addition, we discovered that this mapping can be of use for extraction of protein-protein interactions.

# Acknowledgements

I wish to thank my two supervisors Tore Amble and Rune Sætre for providing great motivation, tutoring and help. Your effort and dedication to the field of Natural Language Processing is admirable.

This thesis could not have been written without the support of my family, which always provided a safe place away from study and work whenever needed.

And my friends, with whom I have spent five smashing years.

*Ars sine scienta nihil est.*

# Preface

This project is only a small piece in the great puzzle of GeneTUC. Yet, as the results and implications become evident, it may prove to be a very interesting piece.

The focus of this thesis has drifted a lot since the beginning in January, 2006. Initially, the goal were benchmarking: testing the quality of semantic understanding in GeneTUC. The method applied proved to be of mediocre suitability, and the results in no way revolutionary. However, after investigation of the implications of the results, it proved to enable some highly interesting features of GeneTUC.

This report investigates the difficulties with mapping of knowledge from one format to another. In this mapping, a clue to solving *"one of the most pressing biological problems"* were discovered.

Harald Søvik
June 8, 2006

# Contents

# 7   Discussion

# 8   Conclusion

# A   Report appendix

# List of Figures

# List of Tables

# Chapter 1

# Introduction

GeneTUC is an Natural Language Processing system intended to be a tool for researchers in the biochemistry/genetics -domain. It is built on the TUC (The Understanding Computer) architecture at the Norwegian University of Technology and Science. It's primary aim is to extract factual assertions from biomedical research articles and compile these into a knowledge base, which later can by queried with questions or statements presented in natural language. Ultimately, GeneTUC may become a full-fledged knowledge system, both capable of answering queries, performing story summarisation and common sense reasoning.

## 1.1 Formal task definition

The original task definition mid- January 2006 read: *As NLP-applications becomes able to extract huge amounts of knowledge from articles, precise measures for their quality is needed. One such measure is their ability to produce the correct answer to a question for which the knowledge has been stated explicit or implicit in the parsed text. This project will apply the BioCreAtIvE corpus to measure the quality of QA-ability in GeneTUC.*

The definition were changed several times during the project, and was eventually agreed to be: *As NLP-applications becomes able to extract huge amounts of knowledge from articles, precise measures for their quality is needed. The goals of this project are (1) to investigate a number of annotated corpuses to determine which is most appropriate to produce as a cross-project semantic representation, and (2) convert as much as possible from GeneTUCs acquired knowledge to this (foreign) format, and produce an (automated) evaluation of the resulting data.*

## 1.2 Task motivation

An important tool in the process of developing NLP systems, is to track changes and their effect on performance. The most straightforward measure is to count the percentage of successfully parsed sentences. Such measure reflects the quality of the grammar (in a optimistic way - sentences may have been incorrectly parsed). Text often contains sentences more and less important to the topic and subsequent querying, and such percentage alone may be an inadequate measure of success when considering that *knowledge* is our building bricks, and that sentences may have more than one grammatical interpretation. "Benchmarking" of systems presents a challenge for two major reasons: First, every development project uses an internal representation of the

knowledge, which is developed by themselves to fit their own areas of utilisation. The representation may even be proprietary. This renders comparison of systems difficult. Second, there is no measure that alone is able to describe the "intelligence" of a NLP-system. Such assessment needs to be compiled from a large number of factors, measuring everything from grammar precision to knowledge recall. Consequently, to ensure successful continuation of GeneTUC development, means to measure success factors have to be identified and implemented.

The target is to identify and implement measure factors on the semantic level. This project is a continuation of an earlier project. Previously, GeneTUC was enabled to print Part-of-Speech tagged text in a format known as PTB, which is an established standard. This allowed us to compare the tagger[1] and grammar to those of other systems.

GeneTUC represents semantic information in TUC Query Language (TQL), which is a type of predicate logic. This shall serve as a basis for identifying factors. The relation between linguistic levels, representation languages is illustrated in Figure 1.1. An example of TQL is given in Table 1.1. It represents the "understanding" of the sentence, and could thus be compared to other systems' "understanding", given some common format.
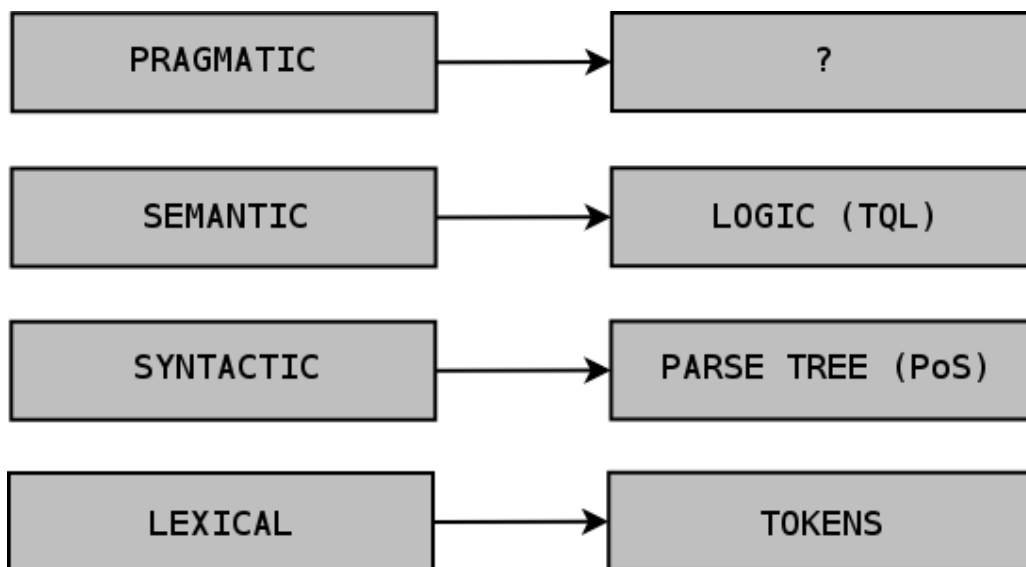


**Figure 1.1:** Linguistic levels and their representations

The course of this project has been pointed out as we came along. Initially, we were going to make *something* that enabled benchmarking. As pointed out by Tore Amble:

*It's too simple to say we are aiming at a running target. It's rather like we pick up the target and run to put it down.*

## 1.3 Related Work

The GENIA-project is currently working on a similar task, quote: "*We are currently working on the key task of extracting event information about protein interactions. This type of information extraction requires the joint effort of many sources of knowledge, which we are now*

---

[1]A tagger assigns possible PoS-classes to each token (word), while the grammar parser tries to find a coherent structure between those tokens with respect to the grammar definitions.

*developing. These include a parser, ontology, thesaurus and domain dictionaries as well as supervised learning models.*"

Other similar project are those working with the Gene Ontology, of which many are participating in the BioCreAtIvE[2] contest. The pinpoint from BioCreAtIvE: "*Many groups are now working in the area of text mining. However, despite increased activity in this area, there are no common standards or shared evaluation criteria to enable comparison among the different approaches. The various groups are addressing different problems, often using private data sets, and as a result, it is impossible to determine how good the existing systems are, whether they will scale to real applications, and what performance can be expected.*"

The Caderige project also works on event-based information extraction[ESP+04], cite: *This project involves teams from different areas (biology, machine learning, natural language processing) in order to develop high-level analysis tolls for extracting structured information from biological bibliographical databases, especially Medline.* The project states their similarity to GENIA, but is specialised on one single organism, *Bacillus Subtilis*.

The Conference on Computational Natural Language Learning (CoNLL) focused[Con05] both in 2004 and 2005 on "semantic role labeling", which in theory is very similar to what we will try to produce. 19 groups participated in last years challenge. As with the majority of mostly similar projects, all of the participants used statistical and/or machine learning approaches. The relevance of Semantic Role Modelling is supported by the CFP mentioned below.

In the end of March 2006, a very interesting Call For Papers[3] were received from the journal "Computational Linguistics", for a "Special Issue on Semantic Role Labeling". From the CFP: *The call for papers of this special issue invites submissions of articles describing novel and challenging work and results in Semantic Role Labeling (SRL) and its applications, with emphasis on the evaluation of qualitative and quantitative aspects that give a deep insight on the SRL task and, in general, on the syntactico-semantic processing of natural language.*

A range of topics were suggested, and among these were: *inclusion of deep semantic information and relations* (for semantic role labeling). This is a very apt description of the solution this project had come to choose.

## 1.4 GeneTUC

I initially assume that the reader is familiar with the GeneTUC system and the TUC architecture. This report will not recite previous work on GeneTUC, and the reader should thus be familiar with earlier publications on the system. However, if you already are familiar with Natural Language Understanding, most of this report will make sense. The background chapters summarise some useful theory on semantics, and will be familiar to linguists.

The development of GeneTUC is described by these influential works:

- Tore Amble, The Understanding Computer [Amb04]

- Anders Andenæs, GeneTUC [And00a]

- Anders Andenæs, GeneTUC - An NLP System for BioMedical Texts [And00b]

---

[2]Critical Assessment of Information Extraction systems in Biology
[3]http://www.lsi.upc.edu/~carreras/srlcl.html

- Rune Sætre, GeneTUC v2 - A Biolinguistic Project, Next Generation [Sæt02a]

- Rune Sætre, Natural Language Understanding - Automatic Information Extraction (IE) from Biomedical Texts [Sæt02b]

- Rune Sætre, GeneTUC: Natural Language Understanding in Medical Text [Sæt06]

This project is neither a solemn software development project nor a solemn research project. It is an exploratory path-finding mission between point A and B, where A is GeneTUC anno 2005 and B is semantic evaluation of GeneTUC anno 2006. We will have to backtrack. We will have to navigate unchartered waters. At times we will ride the tide, other times we will fight the current. None the less will this report document every step we take.

The software development part of this project is to generate some numbers that describe the utility of GeneTUC. The research part is determining how to do this. As mentioned in Section 1.3, this project is operating on subjects that is the focus for many other NLU and NLP-projects today. As papers are published continuously, with references to other possibly interesting articles, we have a lot of opportunities (or dead ends) to investigate. Those papers that have contributed directly to the course of the project, is referenced in the text. In addition, those not directly contributing, but containing relevant or interesting knowledge, have been credited in the bibliography.

## 1.5 Foundations and expectations

Last autumn, a method for lexical and syntactic tag conversion was invented and implemented for GeneTUC. More precisely, we constructed a translation from internal parse structures to an official syntactic representation (PTB). There are numerous texts tagged in the PTB format, among them the GENIA-corpus which is a collection of genetic research articles. This corpus were used for testing and benchmarking in the previous project. The similarity will serve as a priori knowledge for this project, even though the actual product cannot be refactored to fit our new requirements.

This previous project lead to an article published in *Special Issue of LNCS Transactions on Computational Systems Biology* [SSAT05]. It may interest the reader, and has been included in Appendix B.

From this earlier work, we have learned that implementing new representations is possible, but may induce many problems that may not have a perfect solution. A lot of compromises may be made in cases where we i.e. expect to find a 1-to-1 mapping, but discover that the relation is different in a few cases. Rather than causing the whole project to search for another path to the solution, these cases may often be ignored. Much work within natural language processing is best-effort, and this project is no exception. The GeneTUC system is constructed on proof-of-concept basis, and may lack features that would seem compulsory for a system in production use.

That being said, the opportunities and possibilities involved with this project ignites a wildfire of inspiration. As mentioned in Section 1.3, there are many related projects, workshops, conferences and call for papers. While this project progress, we will discover new ways to participate in the peer-reviewed world of science. By submitting results to shared-task conferences, we will be able to get response on the strengths and weaknesses of the system, and discover how it is possible to adapt the system to performing new tasks.

Such possibilities makes it important to keep the basic goal in mind at all times. Whereas we certainly will learn a lot and find a lot of new applications for the system, we must not lose track of what we try to do - construct a solution for simple benchmarking of the system. This is expected to be an achievable goal once we decide on how to do it. But as we gain momentum into the previously unexplored usages of GeneTUC, it is just as important to keep track of all the possibilities that appear.

## 1.6 This report

This report is divided into 8 chapters and 3 appendices.

Chapter 1 contains this introduction. Chapters 2, 3 and 4 constitute the background research part: A study of semantics, a study of linguistic events and a comparison between TQL and GENIA events. Chapter 5 deal with details on implementation of prototypes and full-scale software. Chapter 6 presentes the results. Chapter 7 discusses the project and results in particular. Chapter 8 presents the conclusion and future work.

Appendix A contains appendices directly related to the report: glossary, example code and a comment about possible improvements to the GENIA corpus. Appendix B contains two related papers about GeneTUC and evaluation. Appendix C contains software documentation.

*"Studies on RA time-response or pulse treatment in semi solid or liquid culture show that early RA addition is most effective, thus indicating that early but not late HPC are sensitive to its action."*

```
sk(1)isa study
sk(2)isa response
adj/ra/sk(2)/real
adj/time/sk(2)/real
nrel/on/study/thing/sk(1)/(sk(2);sk(3))
sk(3)isa treatment
adj/pulse/sk(3)/real
sk(4)isa culture
adj/semi_solid/sk(4)/real
adj/liquid/sk(4)/real
nrel/in/treatment/thing/sk(3)/sk(4)
show/id/that/sk(1)/sk(5)/sk(6)
event/real/sk(6)
sk(7)isa addition
adj/ra/sk(7)/real
adj/effective/sk(7)/sk(9)
event/sk(5)/sk(9)
srel/during/thing/sk(8)/sk(9)
indicate/id/that/sk(7)/sk(10)/sk(11)
event/real/sk(11)
sk(12)isa hpc
adj/early/sk(12)/real
sk(13)isa action
sk(8)isa reason
adj/sensitive/sk(12)/sk(14)
srel/being_the/reason/sk(8)/sk(14)
srel/to/thing/sk(13)/sk(14)
event/sk(10)/sk(14)

sk(8)is_the reason
sk(13)is_the action
sk(12)is_the hpc
sk(7)is_the addition
sk(4)is_the culture
sk(3)is_the treatment
sk(2)is_the response
sk(1)is_the study
```

**Table 1.1:** An example of a sentence and its resulting TQL-code

# Chapter 2

# Semantics

The famous linguist Noam Chomsky once said: *"Colourless green ideas sleep furiously"*. On the basis of a collection of rules known as *the grammar*, we may read this sentence and capture the meaning of individual words. We may even read the sentence without stopping, because it is *grammatically sound*.

But when we try to make sense of the sequence of words, *to understand the meaning*, we realize that there are none. The sentence is completely meaningless - none of the words in the sentence can possibly be related to another word:

- nothing can be both colourless and green

- ideas can't have colour-property at all

- nor do ideas sleep

- and sleeping cannot be done furiously

You may disagree. "Colourless" can be a way to say "boring". "Green" can express environmental concern, immatureness, creativity and so on. They are *polysemes* - words with multiple interpretations. In this chapter, we will ignore such phenomena, since they rarely occurs in scientific texts anyway. The point is that meaning is something separate from grammar and something separated from pragmatics. It is your comprehension as competent lingual interpreter.

This chapter investigates the most common methods for transforming a valid sentence to a representation that is useable for representing meaning.

## 2.1   Understanding meaning

*"The meaning of linguistic utterances can be captured in formal structures"*, according to [JM00]. We may relate such formal structures to data, meta data or both. But we may have to think twice to relate them to language. It may be more intuitive to think of it the other way around - language is a way of expressing a formal structure. In an ideal world, translating between language and formal structure should be "lossless". This has proven difficult to achieve. There are several widely used representations that each has their strengths and weaknesses:

- First Order Predicate Calculus

- Conceptual Dependency Diagrams

- Frame-based Representations

- Semantic Networks

On one end of the scale, predicate calculus is very strong with respect to inference and correct logic. On the other end, semantic networks is very expressive. Depending on what sort of meaning one wishes to represent, one may be more proper than the others. For computational linguistics, one often choose an descendant of predicate calculus. For the rest of this report, we will focus on using predicate logic for representing meaning.

There are some common problems associated with semantics, which deserves some thought:

- verifiability - does "no" mean negatively "no" or "I don't know"

- unambiguity - which semantic interpretation is correct

- canonical form - different lingual expressions can have the same meaning

- inference - facts may not be expressively stated, yet available

- variables - indefinite references may cause problems

- expressiveness vs. computionality

We will not elaborate further on basic semantic theory or problems involved therein, but rather skip to aspects that is more interesting from a computer linguistics point of view.

## 2.2 Semantic Analysis

Semantic analysis is the process of figuring out the *meaning* of an utterance, and often somehow resolving the complexity of e.g. synonyms, and ambiguities like homonyms. The result is represented in a structure of some sort. It will be referred to as "meaning structure" or "semantic structure".

The "meaning structure" of the language enables us to extract simple assertions from sentence structures. Predicate-Argument Structures (PA-structures) is an example of such meaning structure. The parsed sentence is matched in a frame-like manner, so that parses matching "NP VP NP" can be translated into a binary predicate: vp(np,np). This simplification renders it easy to understand how a "which"-question can be answered by substituting one of the predicates with an variable: vp(np,X). X unifies with the possible answers. See Table 2.1 for a simple example. (Un)fortunately, sentences in common language is far more complicated, but the basics of this example is illustrative.

The analysis depends on some inputs, often previous steps taken by a system, like part-of-speech tagging (what does the words mean) and grammatical parsing (how are they put together). These tools are applied on a sentence-by-sentence basis. Other knowledge sources concerning more than one sentence may also be included: Knowledge about the discourse (earlier utterances). The context where the discourse is taking place. Common sense knowledge. All of these sources may provide information crucial to the construction of a semantic representation.

*"The dog saw a monkey."*

```
saw(dog,monkey).
```

*"What saw a monkey ?"*

```
saw(X,monkey).
X = dog.
```

**Table 2.1:** Example of a PA-structure

A popular method is the *syntax-driven semantic analysis*. This method is powerful in limited domains, but suffer when flexibility is a requirement. Another approach is to use *semantic grammars*. A quite different solution is to use *information extraction*, which is based on statistics and finite-state automata instead of grammar. We will browse each of these techniques to get a broader view on semantic representations.

### 2.2.1 Syntax-driven semantic analysis

The approach is based on the *Principle of Compositionality*[1], which states that the meaning of a statement can be determined by dividing the statements into smaller parts, and applying the Principle of Compositionality to these parts. For statements which are atoms (not divideable), the meaning is given by the mere atom. Once the statement has been reduced to a tree with atoms as leafs, the meaning can be composed by traversing the tree, using rules attached to the non-terminal nodes to interpret the meaning of descendant nodes. (The mathematician Friedrich Ludwig Gottlob Frege is credited for this principle, but the origin has been disputed.)

Or stated somewhat easier: "The meaning of the sentence is given by the ultimate meaning of the words and their ordering". Although not entirely correct, this is the very foundation. Given a second thought, it states that the semantic meaning can be read out of syntactic structure.

The process of analysing a syntax tree can best be explained by an example: *CCK activates gastrin*. A parse tree for this sentence is shown in Figure 2.1. (The sentence is parsed by GeneTUC, and has been slightly modified for simplicity.)
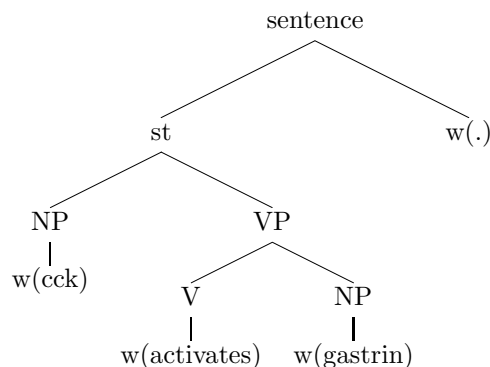


**Figure 2.1:** Parse tree

This parse tree will generate a statement in logic (written using plainform notation) shown in Table 2.2:

---

[1]http://en.wikipedia.org/wiki/Principle_of_compositionality

9

```
(
 (exist e):
 Isa(e, activation) and
 Activator(e, cck) and
 Activatee(e, gastrin)
)
```

**Table 2.2:** Plainform logic example

| Noun | => | cck | | { cck } |
|---|---|---|---|---|
| Noun | => | gastrin | | { gastrin } |
| Verb | => | activates | $\lambda x \lambda y \exists e$ e Isa(activation,e) and activator(e,y) and activatee(e,x) } | |
| | | | | |
| NP | => | Noun | | { Noun.sem } |
| VP | => | Verb NP | | { Verb.sem(NP.sem) } |
| S | => | NP VP | | { VP.sem(NP.sem) } |

**Table 2.3:** Grammar for syntax-driven semantic analysis

The creation of this statement is motivated by the verb "activate". This verb creates an activation-event, which has two predicates attached. Each of these predicated is connected to the event by a constant, e. Then, according to rules of semantics, the subject-predicate (activator) is unified with the subject in the sentence (cck), and the object predicate (activatee) with the object (gastrin).

This method requires specific knowledge: the system must know about the verb and which predicates it has. This becomes a problem when considering all the verbs in the English language, and a lot of verbs may occur in different settings, e.g. both transitively and ditransitively. It would be a laborious task to construct such templates for each of them, with according NP's. Instead, there is a more general way.

It is possible to construct rules for semantic understanding out of rules for syntactic understanding, as stated in the rule-to-rule hypothesis [E.76]. The concept is not complicated. Each grammar rule has an attached semantic rule. By applying these attached rules in parallel with the syntactic analysis, we will build a semantic representation. Again, we will illustrate with an example, using the same sentence as above: *CCK activates gastrin.*

The observant reader probably notices the rather lengthy expression attached to the Verb-rule in Table 2.3. A brief explanation: each verb still needs some template. But, by using $\lambda$-calculus, we can construct a rule where the lambda-variables is unbound until combined with the cck and gastrin-semantic atoms. The verb-rule expression may even be formalised according to verb class, so that intransitive verbs, transitive verbs, and ditransitive verbs has a supertemplate, and an underlying function fills this template according to special rules.

TQL is built by syntax-driven semantic analysis, and is described in detail in section 4.1.

## 2.2.2   Semantic grammars

According to [JM00], syntactic grammars are not well-suited for compositional analysis of semantics. This is apparently not a surprise, *since capturing elegant syntactic generalisations and avoiding overgeneration carry considerably more weight in the design of grammars than*

*semantic sensibility does.* This mismatch manifests itself in tree ways:

- Key semantic elements are distributed widely across syntactic parse trees, thus complicating composition

- Syntactic parse trees contain constituents irrelevant to semantics

- The general nature of many syntactic constituents lead to meaningless semantic attachments

Semantic grammars are more oriented towards the needs in compositional analysis. Rules are constructed so that key components occur in the same rule, and such rules are not overly general. They also enable resolution of anaphora and ellipsis, and are in fact more computationally effective. One of the major drawbacks is that the grammar becomes domain specific. Another drawback is that the grammar may grow very large when constructed to distinguish fine details in the domain.

Using the sentence from the section above, we construct a simple semantic grammar for protein interaction. This is shown in Table 2.4. As it clearly appears, the crucial difference from syntactic grammar is simply that "noun phrase" has become "protein", and "verb phrase" has become "interaction". This illustrates that the domain specific properties of this grammar type.

| | | |
|---|---|---|
| S | => | Protein Interaction Protein |
| Interaction | => | activates |
| Protein | => | cck |
| Protein | => | gastrin |

**Table 2.4:** Semantic grammar

### 2.2.3 Information Extraction

Some applications and domains do not need deep understanding to produce interesting results. I.e. daily news, stock market reports and such where the languages follows a more or less common pattern, more superficial techniques can be used to extract factual information. The two key points in IE is: 1) the information will fit in a pre-defined frame, and 2) the relevant facts are relatively sparse in the text. The usage of such systems is focus of the annual Message Understanding Conference, which in the recent years have had topics including "Satellite launch reports", "News articles on management changes" and "Joint ventures and microelectronics domain".

Superficially, Information Extraction is similar to the techniques discussed above. A typical frame, or "relation" is shown in Table 2.5. However, the problems in this approach is slightly different. Primarily, recognition of entities is the major obstacle. In the example, names of the companies have to be identified, in addition to the date. Secondarily, the entities may be referenced by an ellipsis or another reference phenomena. Such references have to be resolved.

Rune Sætre has composed a very informative illustration (Figure 2.2) of the intersections among Information Retrieval (search), Information Extraction (shallow linguistic knowledge) and Natural Language Processing (deep linguistic knowledge). It is clearly shown how the different techniques can be employed to structure information in text. Syntax-driven semantics and semantic grammar are contained within the deep-knowledge NLP-box.

*"Yesterday, New-York based Foo Inc. announced their acquisition of Bar Corp."*

.. applied to the relation:

```
merge(company1, company2, date).
```

.. should result in the extracted fact:

```
merge(foo inc, bar corp, 2006-02-04).
```

**Table 2.5:** Information Extraction illustrated
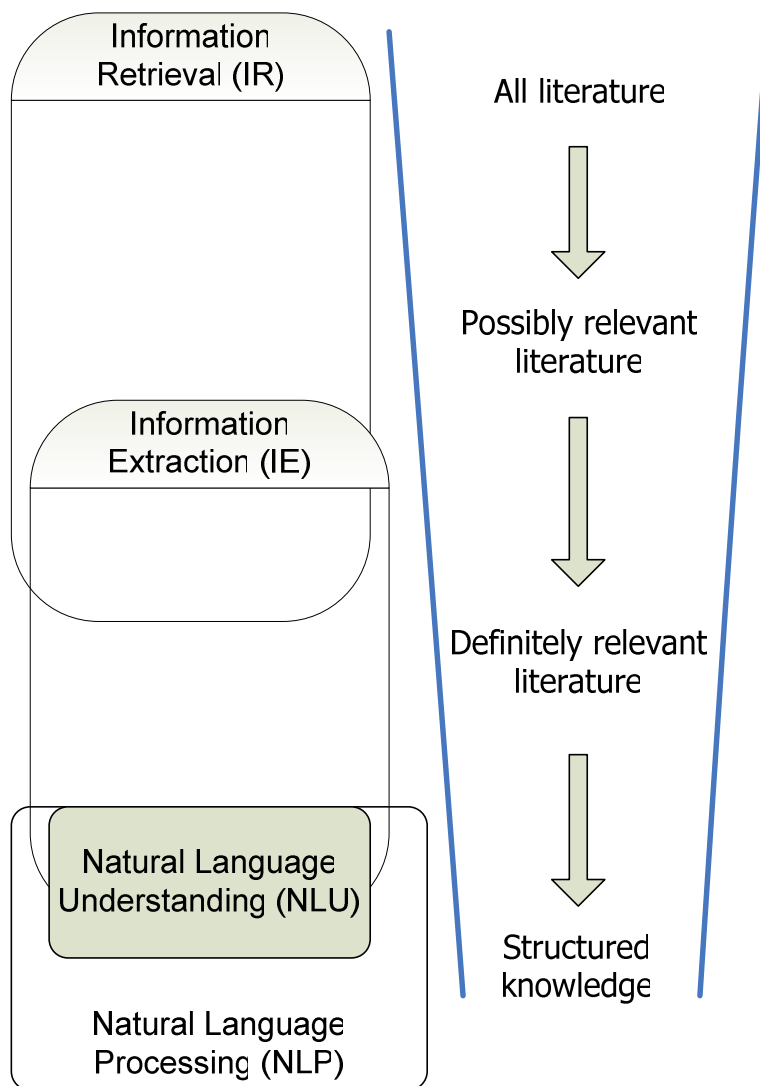


**Figure 2.2:** Conceptualisation of information treatment

## 2.3 Common evaluation criteria

There are some common scoring methods that is used by all NLP groups today. They are based on a atomic unit, e.g. *facts*, and can be somewhat ambiguous or difficult to measure in the semantics domain. What is a *fact*? Well, it should be well known now that cck activates

gastrin. Counting all occurrences of such informal word sequences in the text is a laborious task, but is required and feasible. Counting them and comparing them in the representation in your own system can be more difficult when dealing with semantics. It is easier if one instead of *facts* deal with e.g. *part of speech tags*, which is strictly related to tokens. One tag per token makes it relatively easy to compare "golden" tagging against local tagging.

Recall and precision are the most common measurements. The values they measure are tightly connected, a system which achieves good recall, often has poor precision. And visa versa.

### 2.3.1 Recall

Recall is a measure of how much relevant information the system has extracted (out of what possibly can be extracted).

$$recall = \frac{factslearned}{factsintext} \tag{2.1}$$

### 2.3.2 Precision

Precision (also known as accuracy) is a measure of "information correctness", that is, what percentage of the learned facts are learned correctly.

$$precision = \frac{correctfactslearned}{factslearned} \tag{2.2}$$

### 2.3.3 Fallout

Fallout is a measure of the systems ability to ignore facts which are contradictory to previously learned facts. It could also be called "contradictory recall".

$$fallout = \frac{incorrectfactslearned}{spuriousfactsintext} \tag{2.3}$$

### 2.3.4 F-score

Because of the mutual nature of recall/precision, their features have been combined in a measure named "F-score". Using this measure, it is easier to compare different system.

$$F = \frac{2 * Recall * Precision}{Precision + Recall} \tag{2.4}$$

The measure can also be balanced by a parameter $\beta$, which favours precision for $\beta > 1$, and recall for $\beta < 1$. $\beta = 1$ gives the equation above.

$$F = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall} \tag{2.5}$$

## 2.4 Established representations

This section investigates existing standards for representing knowledge that is located approximately at the semantic level. These differ from the *semantic representations* listed above (start of Chapter 2) by representing *partial, domain specific content* rather than the *complete content* of the lingual expression. Such limited content is useful in applications where a certain kind of knowledge has to be structured, e.g. in biochemistry journals, protein interaction is a favourite topic. See examples in the sections below for examples of different applications.

### 2.4.1 Predicate-argument structures

PA-structures has been employed by several projects for information extraction purposes: the GENIA-project[Tat] in "*Automatic Construction of Biomedical Information Extraction Rules as Predicate-Argument Structure Patterns*"[Yak], and "*Annotation of Predicate-argument Structure on Molecular Biology Text*"[TOiT04]. PASbio in "*Predicate-argument structures for event extraction in molecular biology*"[WSC04]. Language Computer Corp in "*Using Predicate-Argument Structures for Information Extraction*"[SHWA03] to name a few.

PA-structures is a way of representing deep knowledge of the verb and it's arguments by fitting them in a structure. The understanding arises from the point that many verbs represent interesting *events*, and their arguments the participants in the event. The concept is based on the idea of *frames*. A frame is simply a set of slots, consisting of a key and a value. However, the point of a frame is that the keys is predefined, so that a (key,value)-pair can be used to fill a certain slot in certain frames. This may suggest that the frame should be sort of "activated", meaning that one should attempt to fill the other slots as well. This gives us sort of a heuristics for whether we have chosen the right frame or not. Picture a frame labeled "protein-protein interaction". Intuitively we need two slots that is restricted for protein names. In addition, a slot for interaction types is required. See Table 2.6 for a simple visualisation of a frame. This representation might just as well serve as a PA-structure.

| label | protein-protein interaction |
|---|---|
| protein 1 | cck |
| protein 2 | gastrin |
| action | activate |

**Table 2.6:** Frame / PA-structure

PA-structures can be employed in many fashions within the information-extraction domain. GENIA tried it as an approach for generalising IE-rules: "*Because PA-structures represent generalised structure of syntactical variants for the same relation, patterns on PA-structures are expected to be more generalised than those on surface sequences of words.*" This effort were invested to simplify the time-consuming and labour-intensive task of crafting rules for surface IE. An interesting aspect of the research paper was that a full parser were used to determine POS-tags. The overall best result was 65.2% precision, 41.4% recall and 50.6% F-score ($\theta =$ 0.45). Event though these are promising numbers for a system in development, GENIA later abandoned this trail of research.

The Language Computer Group showed in their paper[SHWA03] that *"accurate predicate argument structures enable high quality IE results"*. Claiming that their results in general is *"no more than 10% worse than the results of hand-crafted rule systems"*, they certainly show an interesting use of PA-structures. Unfortunately, these results were produced on hair-thin domains, like "market change" or "death" in newspaper articles.

As for these and other projects using PA-structures, the utility for our project is limited to that of reading of interesting experiences and food-for-thought. We have no intention of applying PA-structures for actual information extraction, but rather *representation* of extracted information. It also appears that no actual "golden" corpus for PA-structures exist. Thus, we abolished the idea of PA-structures as a benchmarking factor.

### 2.4.2    Gene Ontology and BioCreAtIvE

Gene Ontology (GO) is an effort to create a "common ground" in biochemistry. Their summary reads: *"Genomic sequencing has made it clear that a large fraction of the genes specifying the core biological functions are shared by all eukaryotes. Knowledge of the biological role of such shared proteins in one organism can often be transferred to other organisms. Knowledge of the biological role of such shared proteins in one organism can often be transferred to other organisms. The goal of the Gene Ontology Consortium is to produce a dynamic, controlled vocabulary that can be applied to all eukaryotes even as knowledge of gene and protein roles in cells is accumulating and changing. To this end, three independent ontologies accessible on the World-Wide Web (http://www.geneontology.org) are being constructed: biological process, molecular function and cellular component."*

GO provides us with a very high-level approach to semantics, and it can even be argued that it is beyond linguistics and way into biology. An illustration is the simplest way to demonstrate some features of GO. See Table 2.7. This set of data illustrates the data associated with the term *activation of MAPK activity during osmolarity sensing*[2]. It corresponds to a regular entry in an ontology. The item in question belongs in some hierarchy of classes, it has a set of equivalents/synonyms, an identificator and so on. These data are probably well known or at least deducible for biologists working with MAPK-activation.

Now, consider what great effort it would take to identify and annotate this expression among tens and hundreds of others like this in a biochemistry paper. In addition consider what explicit understanding of the actions and processes described such an identification carries.

BioCreAtIvE is a workshop which focuses on identifying GO-entities in biomedical papers. The existing entities needs to be recognised and labeled with the correct tag. Entities that do not exist in GO has to be inserted into the ontology at a estimated location. These are the goals of BioCreAtIvE, paraphrased. We will have a closer look at them after a short introduction to BioCreAtIvE.

The motivation for BioCreAtIvE is similar to that of numerous other biolinguistics biolinguistics[3] projects. Because of technological development and advances in biochemistry (e.g. sequencing of the human genome), the number of experiments in biochemistry has exploded, leading to corresponding exponential growth in number of research articles. To find the most relevant articles to your research can be a labourous task. This motivates the development of automated computer systems for maintaining knowledge.

---

[2]http://www.godatabase.org/cgi-bin/amigo/go.cgi?view=details&query=GO:0000169
[3]biolinguistics refers to the intersecting domain between linguistics, computer science and biology

```
activation of MAPK activity during osmolarity sensing
Accession: GO:0000169
Ontology: biological_process
Synonyms: exact: osmolarity sensing, activation of MAPK activity
Definition: Any process that initiates the activity of the inactive enzyme
            MAP kinase activity during osmolarity sensing.
Comment: None
Term Lineage

Graphical View
all : all (166882)
 GO:0008150 : biological_process (118771)
  GO:0009987 : cellular process (72040)
   GO:0007154 : cell communication (10510)
    GO:0007165 : signal transduction (9299)
     GO:0007166 : cell surface receptor linked signal transduction (5201)
      GO:0007231 : osmosensory signaling pathway (35)
       GO:0007234 : osmosensory signaling pathway via two-component system (26)
        GO:0000161 : MAPKKK cascade during osmolarity sensing (15)
         GO:0000169 : activation of MAPK activity during osmolarity sensing (2)
```

**Table 2.7:** Gene Ontology example

The problem with these systems is attributed to the people creating them. Computer scientists tend to have difficulties of understanding the biologists needs, and biologists have problems expressing their needs *in a way computer scientists can understand*. Because of these problems, more effort have to be focused on establishing standards and common goals.

As for the 2004-edition of this workshop, the two tasks ("stepping stones") were:

- **entitiy extraction**
  This task trains the ability of systems to recognise names of genes and proteins. Training data is readily available as huge databases with names of genes and proteins. More precisely, the task was divided in two subtasks, where the first required identification of a number of names, and the second required listing of a number of identificators associated with the genes mentioned in an abstract.

- **functional annotation of gene products**
  The second task concerned GO annotation of protein names in full-text biomedical articles. It was divided into three subtasks: First, provide a substring from the article that motivates (the) GO annotation. Second, provide the actual GO annotation. Third, suggest a number of other articles that is relevant to the GO annotation of the annotation in the second subtask.

Whereas participation in this workshop is outside the current scope of GeneTUC development, the training and evaluation data may be a source for benchmarking our understanding of biomedical text. However, the connection between text and entities in the training data is weak. The training text is not marked up, but rather consists of a list of article identificators, identified entities in this article and the estimated GO-term for this entity. This presents a problem e.g. if we should try to "learn from mistakes", by not letting us identify exactly where the mistake has been made.

Fibroblastic tumours express nf-kappa b-inducible early genes.

```
EVENT E2
TYPE : Gene_expression
THEME : T11
CAUSE : T10


<sentence id=''S2''>
 <termid=''T10'' lex=''fibroblastic_tumor'' sem=''Tissue''>
  Fibroblastic tumors
 </term>
 that express
 <term id=''T11'' lex=''NF-kappa_B-inducible early genes
  sem=''DNA_familiy_or_group''>
  NF-kappa B-inducible early genes
 </term>
</sentence>
```

**Table 2.8:** GENIA representation of a sentence

### 2.4.3   GENIA events

It proved that the GENIA project had abolished the PA-structures standard in favour of a standard based on *events*. This section will show brief examples of events in GENIA and the similarity with TUC Query Language (TQL). The reader is referred to chapter 3 for a throughout explanation of events in general linguistics.

Table 2.8 shows a sentence with matching GE-code. It contains an identified event, E2, of type "gene_expression". The "theme" and "cause" is given by references to the marked-up sentence, where the reference resolves to "fibroblastic tumour" and "NF-kappa-B-inducible-early-genes". The marked-up sentence also contains delimitation of the terms, and a pointer to an ontology/semantic network.

The suitability of this representation is illustrated by pointing out similarities with corresponding TQL-code. This is shown in Table 2.9. TQL-code will be explained in detail in Section 4.1. Superficially, the similarities is that both systems has identified an event, which is about express(ion) of a (fibroblastic) tumour and (NF-kappa-B-inducible-early) genes.

## 2.5   Thematic roles

Thematic roles (semantic roles) is not a semantic representation in the sense of the ones mentioned above, but is nonetheless interesting in this case. Some of these roles appear in event theory. We spent some time researching the opportunity for how this type could be used, and it is included as a matter of relevant digression. Such thematic roles has been subject of some research in the past, and is closer to a complete lingual representation representation effort, than partial domain specific such. It is related to PA-structures in that it is applying properties to the surface structure of a sentence.

17

Fibroblastic tumors express nf-kappa b-inducible early genes.

```
sk(1)isa tumor
adj/fibroblastic/sk(1)/real
sk(2)isa gene
adj/nf/sk(2)/real
adj/kappa/sk(2)/real
adj/b/sk(2)/real
adj/inducible/sk(2)/real
adj/early/sk(2)/real
express/sk(1)/sk(2)/sk(3)
event/real/sk(3)


sk(2)is_the gene
sk(1)is_the tumor
```

**Table 2.9:** TQL-representation of a sentence

The roles may serve as slots in a frame system. Table 2.10 shows abstract roles from FrameNet. They are abstract since they are domain-independent. Focusing on a single domain, would allow us to call the roles something more descriptive for humans.

## 2.6 Summary of evaluation

Each representation mentioned has been qualitatively investigated, and two criterias can be identified for ranking the candidates:

- **Corpus availability (1)**
  We need a marked-up corpus that can serve as a "golden standard" for the parsing result of a given collection of texts. This will allow us to use precision and recall to evaluate the quality of GeneTUC in the areas which the chosen representation defines.

- **Representation appropriateness (2)**

Each representation has a set of features that allows representation of certain types of knowledge. These types may be more or less appropriate for our cause ("semantic benchmarking"). These features has been subjectively evaluated through discussion.

**GENIA Events** is most appropriate, and we will thus continue investigating events, first in a broad, linguistic sense and then later in the specific sense we will employ them. (See Table 2.11 for the evaluation.)

| Role | Example |
|------|---------|
| Agent | *Henry* pushed the door open. |
| Cause | *That* amazes me. |
| Degree | I *rather* eat at ICSI. |
| Experiencer | It may have been that *John was hungry* at the time. |
| Force | If this is the case, can it be *sustained by evidence*. |
| Goal | He went *to London*. |
| Instrument | I used *a spoon* to eat the soup. |
| Location | He found a coin *amongst the weed*. |
| Manner | His brow *arched delicately*. |
| Null | *It* would be foolish to say no. |
| Path | He *walked* slowly *over*. |
| Patient | He *grips it* tightly. |
| Percept | What is *apparent* is *that could have been a long sentence*. |
| Proposition | The roles does not *demonstrate independence*. |
| Result | He was able to *charm* them *into hiring him*. |
| Source | He knew he was *approached from behind*. |
| State | He *spied* out her *hollering at all*. |
| Topic | We should be *alert with fireworks*. |

**Table 2.10:** Thematic roles from FrameNet

| | (1) | (2) |
|---|-----|-----|
| **Predicate Argument Structures** | Low | Low |
| **Gene Ontology** | High | Low |
| **GENIA Events** | Medium | High |
| **Thematic Roles** | Low | Low |

**Table 2.11:** Evaluation of semantic representations

# Chapter 3

# Events

A part of *meaning* is things that *happen*, i.e. some interaction (or lack of) regarding one or more entities in the world of discourse. An investigation was conducted in [Ros99], focusing on the two questions: (1) what are the primitive elements of events, and (2) where in the grammar are they represented. This section contains a summary of this research, and an asset of implications for this project.

## 3.1   History lesson on linguistic events

To understand the current theory of events, one have to compile and understand a lot of previous work. There is no high-level point of entry or shortcut to understanding events. Therefore, a quick history lesson is in order.

### 3.1.1   Classification

Classification of events has been a target of much research in the past. This research has established knowledge on how to represent events, and a vocabulary on the topic. Most of the effort has been focused on classifying a sentence, predicate or verb. The overall goal is to classify enough low-level parameters or attributes of a passage to suit a purpose, often *searching* or *document* classification. The research has not conclusively established how events is represented in language.

*Classes as event primitives* was the initial suggestion for event-classification of verbs. The notion was to distinguish between states and events, where *states* were defined by classes that had a terminal/culmination feature. A *movement* expressed an incomplete process or state with no terminus. An *action* were a process with an inherent end.

This research has been carried further on, by listing verbs belonging to each of the three classes, and developing diagnostics for membership withing each class. These diagnostics were based on semantic entailments, like if an event has already taken place when it is in progress. It turns out that a crucial difference is delimitation of an event. Observe the difference between:

- *Terry is running.*
  (which entails that he has run)

- *Terry is building a house*
  (does not entail that he already has built a house)

Further, a influential research branch came up with a four-way classification approach, where verbs can be denoting one of the following: *states*, *activities*, *achievements* or *accomplishments*.

- activities - events that go on for a time, but do not have to terminate
  *Terry walked for an hour.*

- accomplishments - events that proceed toward a logically necessary terminus
  *Terry built five houses in two months.*

- achievements - events that occur at a single moment
  *Terry reached the summit in 15 minutes.*

- states - non-actions that hold for a while, but lack continuous tense
  *Terry knows the answer.*

Based on this, a fifth class has been proposed:

- semelfactives - instantaneous non-culminating events (that do not change state)
  *Terry knocked the door.*

### 3.1.2 Extra-verbal factors

The approach for classification presented above, assumes that the verb denotes the event, and thus should be classified. However, it has been noted that other items in a sentence contributes to the construction of an event, e.g. objects and adjuncts. Also, one verb may belong to multiple classes depending on the context. Therefore it has been argued that classification should be compositional. Observe this example, with one verb, but with and without a direct object:

- *Bill ran **the mile** in 5 minutes.* (accomplishment)

- *Bill ran for 5 minutes.* (activity)

In some languages, the case of an object may lead to multiple interpretations. German is such a language. English is not, and therefore we skip the examples of this phenomenon.

Verb particles and resultive predicates may change the event class:

- *Terry thought for an hour.* (activity)

- *Terry thought **up an answer** in an hour.* (accomplishment)

Co-native alternation and antipassive alternation can also change the event classification of a verb:

- *Terry cut the bread in 10 minutes.* (accomplishment)

- *Terry cut **at** the bread in 10 minutes.* (activity)

These examples show that (a) not only the verb determines the event type, and (b) systematic relations link sentence structure and event type. It is not yet entirely clear how syntax and event type is related. In general, classifications may not be the best primitive type for describing events. And they certainly does not bring us to understand where events are encoded.

### 3.1.3 Parametrisation of event classification

A lot of work has been conducted on identifying characteristics that identity an event and place it in a certain class. The method of searching for such underlying parameters is often referred to as the neo-Vendlerian approach, based on the theory of *Vendler classification*, which make up the four-way classification approach mentioned earlier in this section.

This research branch has concluded that event classes are not primitive, and thus could classification be based on deeper characteristics. The four Vendler classes may be generated by the two binary terms *continuousness* (duration) and *boundedness* (terminus). Table 3.1 show the combinations.

|  | +bounded | -bounded |
|---|---|---|
| +continuous | accomplishment | activity |
| -continuous | achievement | state |

**Table 3.1:** Verkyul's parameters for encoding of Vendlerian classes

Multiple researchers have reached conclusions similar to this model, and I shall not recite each of them in detail. The point is to keep these classes in mind when we investigate the entrancé to events in at the lexical, syntactic and semantic level.

## 3.2 Current views

Events can be thought of as related to the lexicon, where all words that is verbs encode actions and all nouns encode entities (agents, patients, instruments). Another way to identify events is through the use of semantics - the meaning contains a representation that can be identified as an event. A third possibility is identification by syntax, because events is strongly related to initiation and termination, which again related to syntactic functions like case and agreement. The contents of this section is heavily distilled, and is meant to give a brief overview of different approaches to event discovery. There are three approaches, which all vary in what they are trying to represent. Semantics represent the event itself, as a primitive. Syntactic approaches sees the event as a compositional entity, and tries to decompose and construct it. Lexical approaches associate the event with arguments and functional projections. All of them may be correct.

### 3.2.1 Events at lexical/syntax mapping level

As suggested in the previous section, there is a tight (but possibly incomplete) relation between the lexical properties of a head verb of a sentence and the event it denotes. Likewise, the lexical properties is tightly connected to the syntactic structure the verb is used in. Research on

the relation between lexical properties have been strongly influenced by the syntactic relations, resulting in two points of view: An approach based on mapping from lexical properties to argument structure and an event classification related approach.

The first point of view - mapping theory - sums up to that there exist a one-to-one mapping for verbs from lexical to syntactic and semantic elements. This means that we can construct a set of rules which examines arguments in certain positions, and derive an event. To specialise these rules, one could put verbs in semantic categories, and assign a set of rules to each category. This intersects with the second, classification based approach, which employs the idea of creating "frames" for the verb to fill, both for syntax (e.g. a verb takes a direct object) and semantics (see Pustejovsky's Lexical Concept Structure[Pus91]).

### 3.2.2 Events at syntactic level

This approach describes events as structures that match a predefined pattern regarding case and agreement. Interesting research has been conducted in this field lately, see [Bor94], [Bor96], [BB96], [Tra92], [Tra94], [Tra96], [Tra00], [RR98], [RR00]. We will not go into the details for these papers, as they are superficially similar. An example of the rule-structure is shown in Figure 3.1.
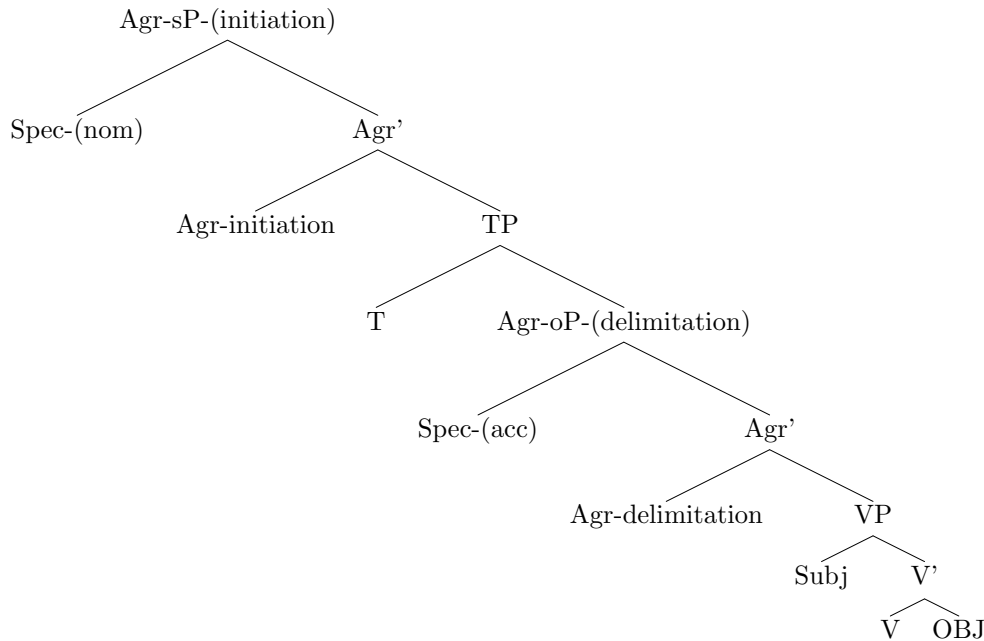


**Figure 3.1:** "Ritter and Rosen's Event Syntax."

The structure is a chain of rules, where each (parent) node that is matched, must have the left child(-rule) satisfied. The root node (initiator) tries to determine the "agent" in the action. Likewise, the delimitation is the "patient" of the action. Between theses, there may have to be some kind of agreement. Likewise, for the verb, there must be some other kind of agreement, at least with the agent.

There are several advantages with this approach: The syntactic nature provides a sort of compositionality. It implies that case and agreement correspond to an interpretative material. It provides a natural way for event initiation and delimitation to follow case and agreement checking. And it observes, but is not limited, by differences in languages.

### 3.2.3   Events at semantic level

Ancient philosophers Panini and Plato observed that the language consists of words that denote action and non-actions/objects, e.g. verbs and nouns. Davidson[Dav67] proposed that action sentences include an event variable in their semantic representation. His subsequent word has lead to the notion of *the Davidsonian e*, referring to usage of the constant $e$ in logical expressions built on this idea.

Davidson argues that a sentence expressed in logic, can contain an event which is expressed as another "thing", and thus be modified and quantified. This eliminates the problem of arity[1] of event expressions. Consider the sentences:

- "*I patted a dog.*"

- "*I patted a dog carefully.*"

We would like to have a predicate for the intransitive verb *patted*. Then, without the Davidsonian e, we would need a special predicate for patting something slowly:

- pat(I,dog).

- pat_carefully(I,dog).

The problem with this approach is that to pat something carefully is a specialisation of patting something in general. Thus, the *carefully*-property should be inheritable or in some other way transferable to the pat-action:

- $\exists e$ pat(I,dog,e) and carefully(e).

This example shows the ability of the Davidsonian e to capture modifications of verbs, and thus letting us specify verbs and modifications apart. Further refinements, neo-davidsonianism, suggest that the e could be treated differently with regards to whether it culminates: *Cul* (achievements and accomplishments) or not: *Hold* (states, activities).

- I patted a dog.

- $\exists e$ (patting(e) and agent(I) and patient(dog) and carefully(e) and $\exists t$ (t < now and Cul(e,t)))

## 3.3   State of the art

These papers were suggested as representative background reading in [CFP06]. Events and semantic roles are almost the same phenomenon. They may seen somewhat different at the surface, but they both deal with "who and what" in semantics.

In [NH04], question answering based on semantic structures is examined. The abstract states that the ability of a system to answer questions is based on 1) the depth of available semantic

---

[1]arity is the number of parameters a function take

representations and 2) the inferential mechanisms supported. The problem with todays systems is their dependency of questions and facts being stated in the same way. This could, according to the authors, be solved by using deep and precise semantic knowledge.

[SHWA03] presents a similar paper, focusing on Information Extraction using events to fill *event frames* ("templettes"). The approach focuses on extracting the most interesting datas of a event and insert them into a structure, and has been mentioned in an earlier section. It is pointed out that frames would have to be domain-specific. A novel approach for extraction is presented, using statistical and machine-learning methods. Their F-score were up to 88%, which is better than earlier statistical approaches. The method is, however, still domain dependant.

[GJ02] focuses on identifying semantic relationships/roles "filled by constituents of a sentence within a semantic frame". As various methods excel in narrow domains, still systems rely on hand-crafted rules and domain knowledge. The method presented is (highly) statistical, and is employing frames and ontologies from FrameNet. An interesting exploration in this paper, was using semantic/thematic roles for generalising beyond the domain. They confirm that semantic roles do not seem to be a function of the syntax tree. They further conclude that sentence-level features may prove to be an essential mechanism to support frame representations, and learning generalisation rules.

[XP04] *"takes a critical look at the features used in the semantic role tagging literature and show the the information in the input ... has yet to be fully exploited"*. The approach is based on a "maximum entropy classifier", so this is a machine learning approach. While achieving a F-score of 82,8%, their method only involves using verbs as root for frame templates, and thereby events.

[ea05] describes a system that uses a full parser for text summarisation. The SRL (Semantic Role Labeling) is based on *augmented pushdown transducer*, which draws a semantic graph for a set of sentences. However, the system relies strongly on Part-Of-Speech tagging, and it is obvious that verb heads is the anchor for predicates.

Tenth Conference on Computational Natural Language Learning[Con05] also focuses on Semantic Role Labeling. This conference is organised by some of the same researchers that assemble the editorial board in [CFP06]. As with the papers above, *"the general goal is to come forward with machine learning strategies"*, and not deep semantic understanding.

Several of these papers suggest that deep semantic knowledge is a key factor in development of NLP-applications.

# Chapter 4

# Event identification

We believe to have an advantage over the approaches mentioned in the previous chapter, since we build the system on top of semantics (logic), instead of "surface text" and syntax structures.

This chapter examines TQL-code and event-annotated text from GENIA. TQL-code is a form of logic that could be descriptively named *relation logic* or *event logic*. Section 4.1 and 4.2 defines the syntax of these two data representations, and section 4.4 aligns examples of both where an event occurs. This is the solution of the actual problem - how to extract events from GeneTUC in a way that is compatible with those used by GENIA. Last, we present the identified patterns.

Finding such patters is difficult, and there are no existing tools to assist the extraction but syntax highlighting and line indents. Since we are going to construct a program that strips away a lot of tags and presents the events in a pretty fashion, it may be wise to postpone identifying all but the simplest patterns.

If we eventually are unable to identify the difficult patterns, we may have to instead try a machine learning approach for learning extraction rules.

## 4.1   TQL definition

The examples in this section is based on parts of the sentence below, which are shown with the resulting TQL-code. They have been carefully selected to demonstrate all features of TQL. The authorative reference for TQL can be found in the TUC-tutorial[Amb]. A quick note on the notation: words starting with a lowercase letter is *constant*, and words starting with an Uppercase letter are *variables*. An *underscore* represents a wildcard.

### 4.1.1   Individual isa Class

A connection between a proper noun and a noun class, as defined by an ISA-relation in the semantic database. This individual/class-relation is shown for all nouns in the sentence. Proper and common nouns are treated differently; Proper nouns that belongs to some class retains their original lexical representation. Common nouns are "skolemised" - replaced by placeholding constants.

In this example, the common noun *effects* is skolemised, while the proper noun *RA* is shown

with it's superclass.

```
E: we have examined the effects of RA on normal (...)

% TQL:
sk(7) isa effect
ra isa lipid
```

### 4.1.2 event/World/Skolem

This rather important statement cannot be demonstrated alone, but will be seen in almost all other examples. The event states that something has happened, usually because of a verb occurring in the sentence. It is used to connect other statements together. The connection between events and worlds is complicated. It is usually manifested by the usage of world-creating words like "think" and "dream". We postpone the discussion until we know *if* it proposes a problem and in what part of the system the problem can be handled most efficiently. In GO, there are levels indicating 1) the certainty of an event and 2) the certainty of the GO annotation code. If we try to produce GO-annotated sentences, this may help us to determine the certainty of an event.

```
% TQL:

event/real/sk(12)
```

### 4.1.3 Verb/Agent/Event

This line states that a noun performs an action of the type defined by the verb, related to the event. The representation is shown when an intransitive verb occurs in a sentence. *Our example sentence does not contain an intransitive verb, so we demonstrate with a simpler alternative.*

As described above, the noun is recognised and skolemised. Then, the verb is shown as an invocation over the noun in an event. This happens in a world - in this case the world "real".

```
E: cells proliferate.

% TQL:

sk(5)isa cell
proliferate/sk(5)/sk(6)
event/real/sk(6)
```

### 4.1.4 Verb/Agent/Patient/Event

States that a noun in the role of an agent performs an action of the type defined by the verb in relation to the noun in the role of patient or object. This is also related to an event. The statement represents a transitive verb occurring in the sentence.

This specific statement reads that there has been an examination, performed by 'I' (which is a simplification of different ways of referring to oneself); The examination was done on a skolem which is a placeholder for "effect". All of this happened in a world "real".

```
E: we have examined the effects of RA on normal (...)

% TQL:

examine/'I'/sk(7)/sk(12)
'I' isa self
sk(7)isa effect
event/real/sk(12)
```

### 4.1.5 srel/Modifier/Class/Individual/Event

This is the representation of a verb modifier phrase, where an individual of a class is modified in an event. The Modifier is usually a preposition, but may also be an adverb.

```
E: (...) cells purified from adult peripheral blood

% TQL:
srel/from/thing/adult_peripheral_blood/sk(11)
```

### 4.1.6 nrel/Modifier/Class1/Class2/Individual1/Individual2

States the same as srel, but in addition individual 1 is modified by individual 2 in class 2. This is the representation of a noun modifier phrase.

```
E: (...) the effects of RA on normal hematopoiesis (...)

% TQL:
nrel/on/effect/thing/sk(7)/sk(8)
```

### 4.1.7 adj/Adjective/Individual/_

This is a simple adjective phrase, stating that the individual has the property defined by adjective. Adjectives may also be nouns used in an adjectival sense, which is the case in this example.

```
E: (...) on normal hematopoiesis by using (...)
```

```
% TQL:
sk(9)isa hematopoiesis
adj/normal/sk(9)/real
```

### 4.1.8   has/SubjectClass/Attribute/Subject/Value

This statement means that the subject of class subjectclass has an attribute with a value. It is for instance used to represent set-membership.

```
E: (...) was also unaffected by the latter agents.

sk(24)isa set
has/set/member/sk(24)/A=>A isa agent
has/set/member/sk(24)/A=>adj/latter/A/real
has/set/member/sk(24)/A=>adj/unaffected/mnda_mrna_level/sk(25,A)
has/set/member/sk(24)/A=>srel/by/thing/A/sk(25,A)
has/set/member/sk(24)/A=>event/real/sk(25,A)
```

## 4.2   GE definition

This section describes how GENIA has defined the representation. There are two types of top-level datatypes: A *sentence* encapsulates the sentence as it is appearing in the text, and includes identification of the entities appearing in the text. An *event* encapsulate information that is specific to the actual event and the entities involved.

GENIA has not yet published any DTD[1] for their events, and we will therefore have to investigate the examples to determine what is acceptable syntax and what is not.

GENIA specialises in annotating biological events, which is defined as *a temporal occurrence involving one or more biological entities*. The GENIA Event Ontology specifies what is within the scope of "biological events".

### 4.2.1   <sentence>

Indicates the span of a sentence in the original text, which consists of one or more terms which is covering some of the words in the sentence - typically nouns. Term is the only tag allowed to be contained within the sentence tag.

### <term>

A term is one or more words which represent an expression, e.g. a [complex] noun. Complex nouns may be nested, such as *(((human T cell leukemia) virus) Tax gene)*. The tag may be

---

[1]Document Type Definition, a way of specifying grammar for an instance of XML

used redundantly, but is commonly just applied to subclasses of the head noun. Adjectives may be included to contribute to subclassification. I.e. will the term *fibroblastic tumor* be tagged, while *tumor* will not be tagged redundantly. In some cases, long chains of nouns do have 3-4 levels of redundancy. Terms often carry those attributes listed below. There are however some semantic terms that does not have a lexical string.

- id
  Used for referential purposes in the <event>-tag. Should be unique at least within the anaphora scope, probably within the entire article.

- lex
  A redundant string containing the coverage of the tag.

- sem
  The class of the noun (possibly complex noun) that is covered.

### 4.2.2   <event>

An event is not merely "something that happens", but in a broader sense "something that has an effect". A generalised example: *"Expression of a gene causes development of a tumor."*. This sentence contains two events: The expression, and the development. Note that all events has a theme property, but that the cause property is optional.

- id
  Identification of an event is important, because one event may serve as theme or cause for another event. The relation between events must be established from the TQL code.

### <type>

The type of a event refers to the ontology that was used to construct a semantic understanding of the text. It may lead to a problem when comparing systems that is using different ontologies. Ultimately, BioCreAtIvE and Gene Ontology may solve this problem by constructing a universal ontology (for genetic sciences)[ea00].

- class
  This is the attribute where the value of type is stored.

### 4.2.3   <theme>

The theme of an event is something that is happening, e.g. the development of a tumor.

- idref
  This refers to a previously defined term.

### 4.2.4 <cause>

A cause is a previous event that is used to explain a theme, e.g. a gene expression.

- idref
  This refers to a previously defined term.

### 4.2.5 <clue>

A clue is a sentence or part of a sentence that suggests the correct interpretation of the event. The clue is for human verification.

#### <clueType>

The textual evidence enabling classification of the event.

#### <linkTheme>

The textual evidence linking a theme to the event.

#### <linkCause>

The textual evidence that links cause to the event.

#### <ClueLoc>

The environmental location of an event.

#### <ClueTime>

The environmental time of an event.

## 4.3 Known problems

In [GEN], these problems were identified:

- some events relate to more themes
- some events act upon other events
- some events are connected with causes
- an event may cause another event

- an event may be negated

It is probable that we also experience these situations as problems, so we will pay special attention to such cases as we figure how to extract our own events.

*ENTITY1 activates ENTITY2* can be expressed in various ways. In [Yak], several variations were identified. These are shown in Table 4.2.

The 7 first sentences is simple in that sense that they all have a verb that invokes the event. The latter ones have a unique pattern that have to be identified. We will now compare GE and TQL example by example to find possible extraction patterns in the examples available to us.

*We have examined the effects of RA on normal hematopoiesis by using early hematopoietic progenitor cells stringently purified from adult peripheral blood.*

```
% TQL:

'I' isa self
sk(7)isa effect
ra isa lipid
nrel/of/effect/thing/sk(7)/ra
sk(8)isa hematopoiesis
adj/normal/sk(8)/real
sk(9)isa cell
adj/early/sk(9)/real
adj/hematopoietic_progenitor/sk(9)/real
adult_peripheral_blood isa tissue
purify/sk(10)/sk(9)/sk(11)
srel/from/thing/adult_peripheral_blood/sk(11)
event/real/sk(11)
nrel/with/hematopoiesis/thing/sk(8)/sk(9)
nrel/on/effect/thing/sk(7)/sk(8)
examine/'I'/sk(7)/sk(12)
event/real/sk(12)
```

**Table 4.1:** Sentence to show examples of TQL-statements

ENTITY1 recognises and **activates** ENTITY2.
ENTITY1 can **activate** ENTITY2 through a region in its carboxy terminus.
ENTITY2 are **activated** by ENTITY1a and ENTITY1b.
ENTITY2 **activated** by ENTITY1 are not well characterized.
The herpesvirus encodes a functional ENTITY1 that **activates** human ENTITY2.
ENTITY1 can functionally cooperate to synergistically **activate** ENTITY2.
The ENTITY1 play key roles by **activating** ENTITY2.

ENTITY2-activation by ENTITY1.

**Table 4.2:** Variations of E1 activates E2

## 4.4 Identification examples

This section contains the set of examples that is available for identifying extraction rules. Each example contains TQL and GE, and a comment on the similarities and possible contradictions observed. Some of the similarities are not so obvious, since there are a lot of synonyms and almost-synonyms in biochemistry. Note that the "events" identified in TQL is different from the event-predicate defined in Section 4.1.2.

We will try to identify "patterns" in TQL that corresponds to an event. We define pattern to be a sequence of statements in logic that can be translated into an event, if certain conditions are fulfilled. Such conditions are usually related to limiting the domain, since GENIA only produces biologically related events.

Each pattern will be throughoutly explained. An example initially contains the sentence, GE-NIA events and then the *essential* TQL. We then try to construct a matrix that identifies correspondences line-by-line. Each correspondence is given a number, an commented in the text. Finally, a FSM (Final State Automata) will be drawn to illustrate the steps that must be taken to recognise the event. The automata will serve as blueprint for implementation of rules in the prototypes.

The FSM scheme consists of arrows, ellipsis and boxes. An arrow denotes flow. The ellipsis are states, and they usually refer to a statement in the TQL code or an explicit description of the current state. Uppercase letters are "unbound" variables. The boxes are program modules, i.e. Ontology, which can answer if a word belongs to a certain ontology, and TQLAbstract which can resolve placeholding constants.

### 4.4.1 Example 1

**Sentence**
*Lipopolysaccharide induces phosphorylation of mad 3*

**GENIA**

- Event 1

  - Type: portein_amino_acid_phosphorylation
  - Theme: MAD3 (Protein_molecule)

- Event 2

  - Type: Positive_regulation
  - Theme: Event 1
  - Cause: Lipopolysaccharide

**GeneTUC**

- lipopolysaccharide isa compound

- mad_3 isa protein_molecule

- phosphorylation isa process

- induce/lipopolysaccharide/phosphorylation/sk(1)

- nrel/of/stuff/thing/phosphorylation/mad_3

- event/real/sk(1)

| GE | TQL | |
|---|---|---|
| Event 1<br>Type: portein_amino_acid_phosphorylation<br>Theme: MAD3 (Protein_molecule) | phosphorylation isa process<br>nrel/of/stuff/thing/phosphorylation/mad_3 | 1 |
| Event 2<br>Type: Positive_regulation<br>Theme: Event 1<br>Cause: Lipopolysaccharide | induce/lipopolysaccharide/phosphorylation/sk(1)<br>phosphorylation isa process<br>induce/lipopolysaccharide/phosphorylation/sk(1) | 2<br>3<br>4 |

**Table 4.3:** Matrix for example 1

**Equality matrix**

A stepwise comparison is shown in Table 4.3, and a Final State Automata illustrates the recognition steps in Figure 4.1 and 4.2.

1. usage of "process" triggers an event

2. an inducement is a positive regulation

3. backreference to the same TQL-statement that triggered event 1

4. the subject of a intransitive verb-relation is the cause
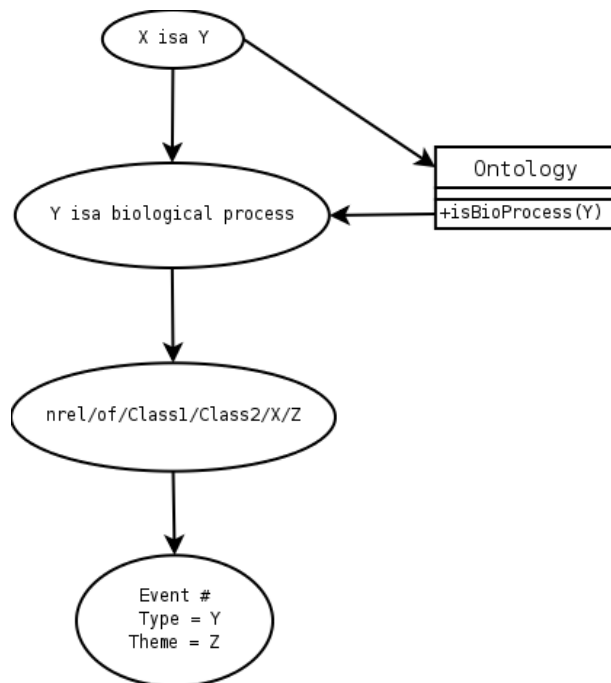
**Figure 4.1:** Pattern for Event 1
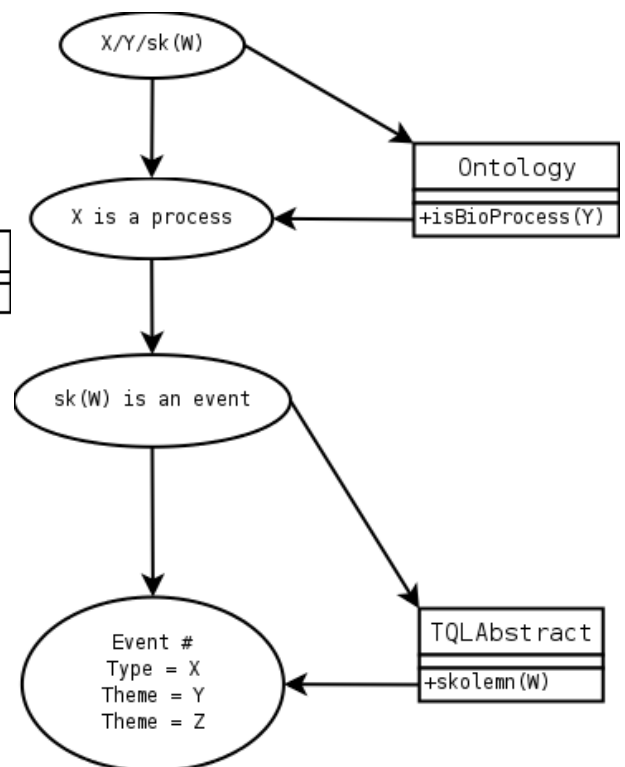


**Figure 4.2:** Pattern for Event 2

### 4.4.2   Example 2

**Sentence**
*Enhancement of human immunodeficiency virus 1 replication in monocytes by 1,25 dihydroxy-cholecalciferol*

**GENIA**

- Event 3

  - Type: RNA_metabolism
  - Theme: human immunodeficiency virus 1

- Event 4

  - Type: Positive_regulation
  - Theme: Event 3
  - Cause: 1,25-dihydroxycholecalciferol

**GeneTUC**

- sk(3)isa enhancement

- human_immunodeficiency_virus_1_replication isa process

- nrel/of/enhancement/thing/sk(3)/human_immunodeficiency_virus_1_replication

- sk(4)isa monocyte

- '1_25_dihydroxycholecalciferol' isa lipid

- nrel/by/stuff/substance/sk(4)/'1_25_dihydroxycholecalciferol'

- nrel/in/enhancement/thing/sk(3)/sk(4)

| GE | TQL | |
|---|---|---|
| Event 3 Type: RNA_metabolism Theme: human immunodeficiency virus 1 | hiv1_replication isa process | 1 |
| Event 4 Type: Positive_regulation Theme: Event 3 Cause: 1,25-dihydroxycholecalciferol | sk(3)isa enhancement nrel/of/enhancement/thing/sk(3)/hiv1_replication | 2 3 |

**Table 4.4:** Matrix for example 2

**Equality matrix**
A stepwise comparison is shown in Table 4.4.

1. "process" triggers event, replication = metabolism (?)

2. enhancement triggers event, enhancement = positive regulation

3. noun modification of trigger for previous event

### 4.4.3 Example 3

**Sentence**
*I kappa b/mad-3 binds to nf-kappa b p50*

**GENIA**

- Event 5

    - Type: Binding
    - Theme: I kappa B/MAD-3 and NF-kappa B p50

**GeneTUC**

- i_kappa_b_mad_3 isa protein_molecule

- nf_kappa_b_p50 isa gene

- bind/i_kappa_b_mad_3/sk(6)

- srel/to/thing/nf_kappa_b_p50/sk(6)

- event/real/sk(6)

| GE | TQL | |
|---|---|---|
| Event 5 | | |
| Type: Binding | bind/i_kappa_b_mad_3/sk(6) | 1 |
| Theme: I kappa B/MAD-3 and NF-kappa B p50 | srel/to/thing/nf_kappa_b_p50/sk(6) | 2 |

**Table 4.5:** Matrix for example 3

**Equality matrix**
A stepwise comparison is shown in Table 4.5.

1. the verb "bind" is in the ontology and invokes an event

2. verb modification, connected to the same event via sk(6)
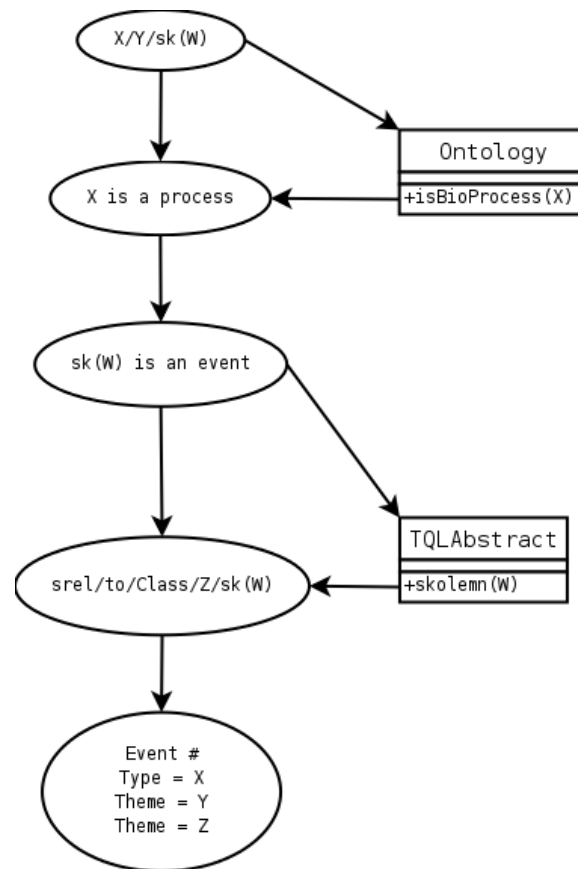
**Figure 4.3:** Pattern for Event 5

### 4.4.4  Example 4

**Sentence**
*Expression of m10 did not affect induction of transcription of hiv*

**GENIA**

- Event 6
    - Type: Gene_expression
    - Theme: M10

- Event 7
    - Type: Transcription
    - Theme: HIV

- Event 8 (negation)
    - Type: Positive_regulation
    - Theme: Event 7
    - Cause: Event 6

**GeneTUC**

- (incomprehensible)

| GE | TQL | |
|---|---|---|
| Event 6<br>Type: Gene_expression<br>Theme: M10 | | |
| Event 7<br>Type: Transcription<br>Theme: HIV | | |
| Event 8<br>Type: Positive_regulation<br>Theme: Event 7<br>Cause: Event 6 | | |

**Table 4.6:** Matrix for example 4

**Equality matrix**
A stepwise comparison is shown in Table 4.6.

1. (incomprehensible)

### 4.4.5 Example 5

**Sentence**
*In this report , we demonstrate that a novel ets-related transcription factor (elf-1) binds specifically to two purine-rich motifs in the hiv-2 enhancer.*

**GENIA**

- Event 9

    - Type: Binding
    - Theme: Elf-1 and HIV-2 enhancer

- Event 10

    - Type: Binding
    - Theme: Elf-1 and two purine-rich motifs (in HIV-2 enhancer)

**GeneTUC**

- 'I' isa self

- demonstrate/id/that/'I'/sk(1)/sk(2)

- event/real/sk(2)

- ets_related_transcription_factor isa protein_family

- adj/novel/ets_related_transcription_factor/real

- sk(3)isa set

- has/set/cardinality/sk(3)/2

- has/set/member/sk(3)/A=>A isa motif

- has/set/member/sk(3)/A=>adj/purine/A/real

- has/set/member/sk(3)/A=>adj/rich/A/real

- has/set/member/sk(3)/A=>nrel/in/motif/thing/A/hiv_2_enhancer

- has/set/member/sk(3)/A=>bind/ets_related_transcription_factor/sk(4,A)

- has/set/member/sk(3)/A=>srel/to/thing/A/sk(4,A)

- has/set/member/sk(3)/A=>event/sk(1)/sk(4,A)

**Equality matrix**
A stepwise comparison is shown in Table 4.7, and a Final State Automata illustrates the recognition steps in Figure 4.4.

1. we consider the set an exclusive TQL block, and identify the intransitive verb

2. apart from A being a motif, we know nothing more of what the binding involves

| GE | TQL | |
|---|---|---|
| Event 9 | | |
| Type: Binding | bind/ets_related_transcription_factor/sk(4,A) | 1 |
| Theme: Elf-1 and HIV-2 enhancer | srel/to/thing/A/sk(4,A) | 2 |
| Event 10 | | |
| Type: Binding | bind/ets_related_transcription_factor/sk(4,A) | 1 |
| Theme: Elf-1 and two purine-rich motifs | srel/to/thing/A/sk(4,A) | 2 |

**Table 4.7:** Matrix for example 5
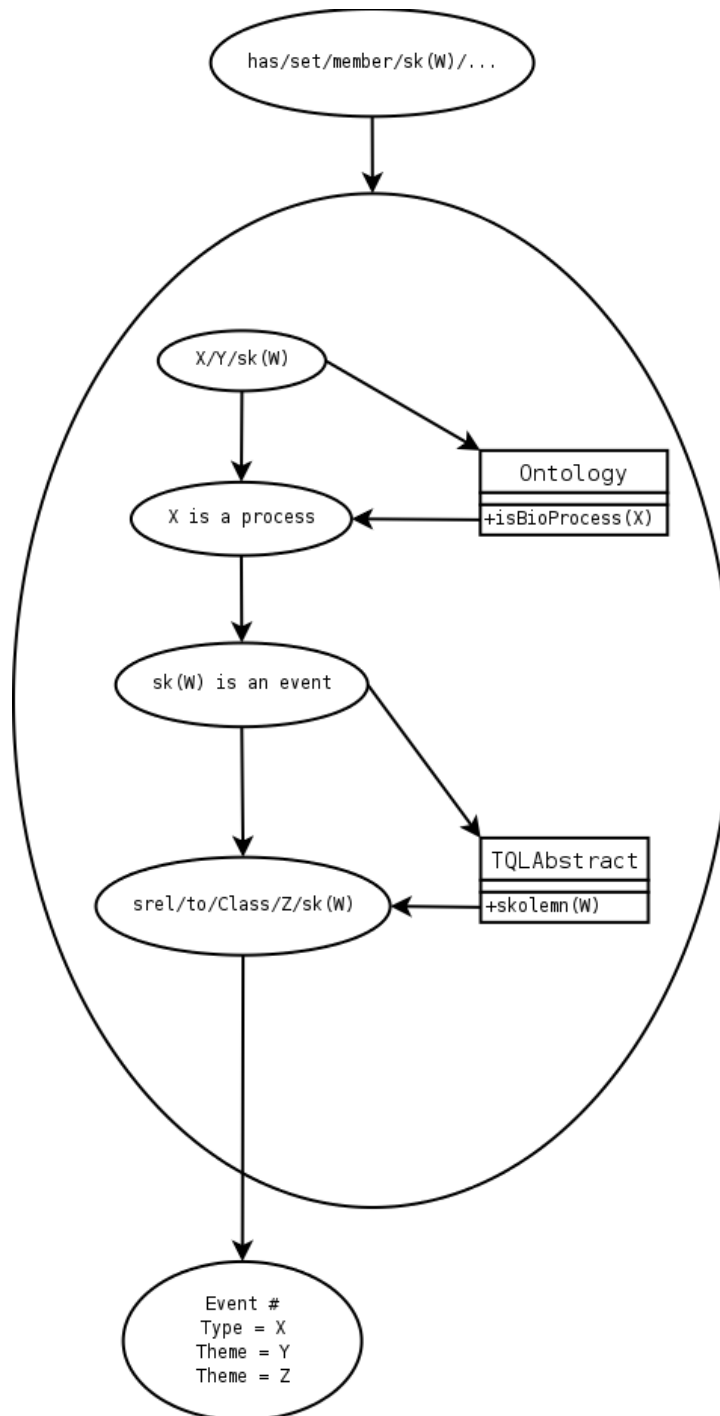


**Figure 4.4:** Pattern for Event 9,10

### 4.4.6 Example 6

**Sentence**
*Similar to its effect on the induction of ap1 by okadaic acid, pma inhibits the induction of c-jun mrna by okadaic acid.*

**GENIA**

- Event 11
    - Type: Positive_regulation
    - Theme: c-jun mRNA
    - Cause: okadaic acid

- Event 12
    - Type: Negative_regulation
    - Theme: Event 11
    - Cause: PMA

- Event 13
    - Type: Positive_regulation
    - Theme: AP1
    - Cause: okadaic acid

- Event 14
    - Type: Negative_regulation
    - Theme: Event 13
    - Cause: PMA

**GeneTUC**

- pma_ isa stuff

- sk(17)isa induction

- mrna isa rna_domain

- adj/c_jun/mrna/real

- nrel/of/induction/thing/sk(17)/mrna

- sk(18)isa acid

- adj/okadaic/sk(18)/real

- nrel/by/induction/substance/sk(17)/sk(18)

- inhibit/pma_/sk(17)/sk(19)

- srel/during/time/sk(16)/sk(19)

- event/real/sk(19)

- sk(21)isa effect

- sk(22)isa induction

- ap1 isa complex

- nrel/of/induction/thing/sk(22)/ap1

- sk(23)isa acid

- adj/okadaic/sk(23)/real

- nrel/by/induction/substance/sk(22)/sk(23)

- nrel/on/effect/thing/sk(21)/sk(22)

- adj/similar/sk(20)/sk(24)

- srel/to/thing/sk(21)/sk(24)

- event/real/sk(24)

- srel/in/time/sk(16)/sk(0)

| GE | TQL | |
|---|---|---|
| Event 11<br>Type: Positive_regulation<br>Theme: c-jun mRNA<br>Cause: okadaic acid | sk(17)isa induction<br>nrel/of/induction/thing/sk(17)/mrna<br>nrel/by/induction/substance/sk(17)/sk(18)<br>sk(18)isa acid | 1<br>2<br>3<br>4 |
| Event 12<br>Type: Negative_regulation<br>Theme: Event 11<br>Cause: PMA | inhibit/pma_/sk(17)/sk(19) | 5 |
| Event 13<br>Type: Positive_regulation<br>Theme: AP1<br>Cause: okadaic acid | sk(22)isa induction<br>nrel/of/induction/thing/sk(22)/ap1<br>nrel/by/induction/substance/sk(22)/sk(23)<br>sk(23)isa acid | 6<br>7<br>8<br>9 |
| Event 14<br>Type: Negative_regulation<br>Theme: Event 13<br>Cause: PMA | | |

**Table 4.8:** Matrix for example 6

**Equality matrix**

A stepwise comparison is shown in Table 4.8, and a Final State Automata illustrates the recognition steps in Figure 4.5.

1. induction establishes an event, because it is a positive regulation

2. of introduces theme

3. by introduces cause

4. the cause resolves to be an acid

5. inhibit is a negative regulation in the ontology and establishes an event

6. induction establishes an event, because it is a positive regulation

7. of introduces theme

8. by introduces cause

9. the cause resolves to be an acid
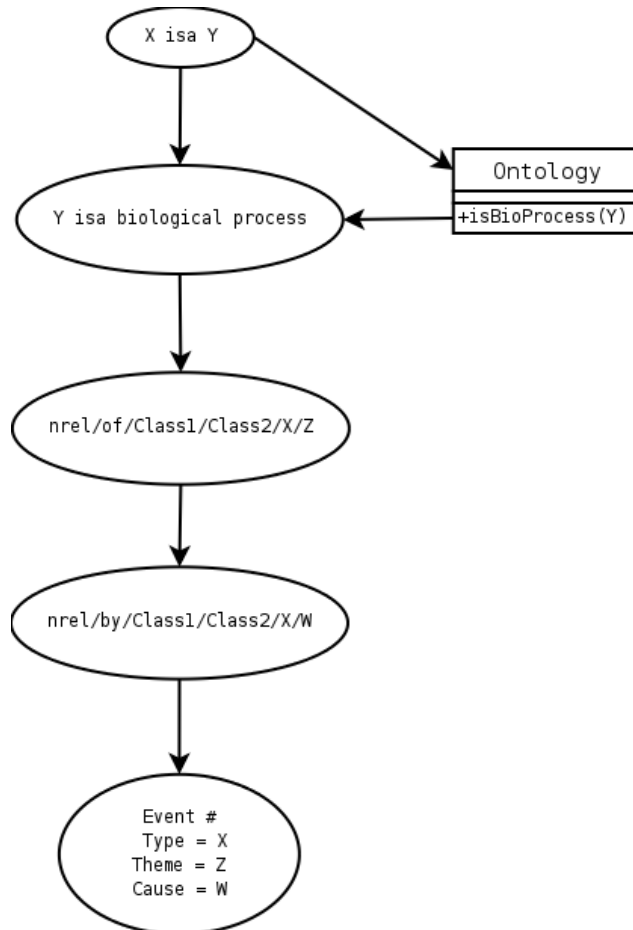


**Figure 4.5:** Pattern for Event 11,13

### 4.4.7 Example 7

**Sentence**
*HS-40 behaves as an authentic enhancer for high-level zeta 2 globin promoter activity*

**GENIA**

- Event 15

    - Type: Positive_regulation
    - Theme: zeta 2 globin promoter
    - Cause: HS-40

**GeneTUC**

- hs_40 isa gene

- sk(26)isa enhancer

- adj/authentic/sk(26)/real

- sk(27)isa activity

- adj/great/sk(27)/real

- adj/level/sk(27)/real

- adj/zeta_2_globin_promoter/sk(27)/real

- nrel/for/enhancer/thing/sk(26)/sk(27)

- behave/hs_40/sk(28)

- srel/as/thing/sk(26)/sk(28)

- event/real/sk(28)

| GE | TQL | |
|---|---|---|
| Event 15 | | |
| Type: Positive_regulation | sk(26)isa enhancer | 1 |
| | sk(27)isa activity | 2 |
| Theme: zeta 2 globin promoter | adj/zeta_2_globin_promoter/sk(27)/real | 3 |
| Cause: HS-40 | behave/hs_40/sk(28) | 4 |
| | event/real/sk(28) | 5 |

**Table 4.9:** Matrix for example 7

**Equality matrix**
A stepwise comparison is shown in Table 4.9, and a Final State Automata illustrates the recognition steps in Figure 4.6.

1. the presence of an enhancer establishes an event (defined in ontology)

2. the presence of an activity establishes an event (defined in ontology)

3. the sort of activity, related by sk(27)

4. "behave" indicates cause when related to an event via sk(28)

5. the event anchor for sk(28)
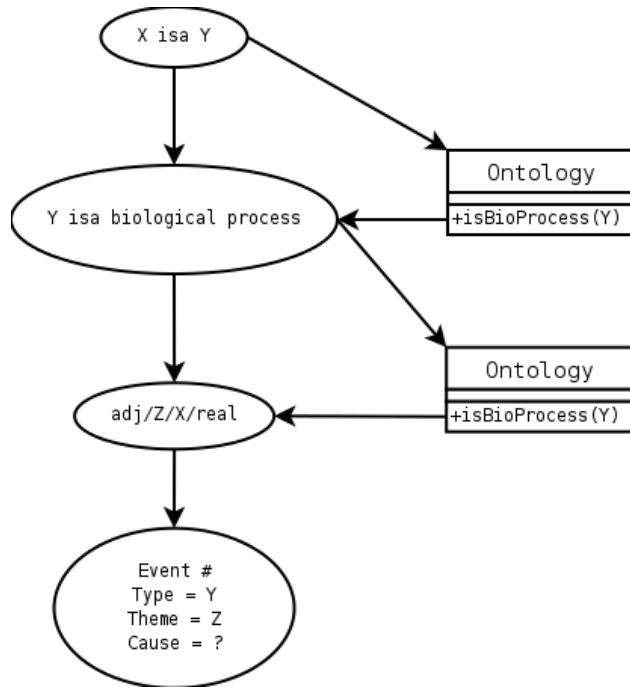


**Figure 4.6:** Pattern for Event 15

# Chapter 5

# Implementation

This chapter contains discussions, decisions and descriptions of implementation-related matters. Three prototypes were constructed during the project, and one was carried out to a full program. The major problems are presented, as well as their solutions.

Some remarks on the event-annotated corpus from GENIA can be found in Appendix A.3.

## 5.1 Prototyping event extraction

We were quite uncertain of how and where the event recognition logic should be implemented. There are (at least) three different possibilities, each with very distinct pros and cons. Prolog is the native language of GeneTUC. Perl is the language other programs accompanying GeneTUC usually have been implemented in. Modern Object Oriented languages (in this case: Java) provides a greater amount of flexibility. Each alternative has been prototyped, and the results are summarised below. Other languages, like Perl or Ruby, would also have been interesting to try, but time forbids this. Code examples of each prototype is included in Appendix A.2.

### 5.1.1 Prolog

Since GeneTUC is written entirely in Prolog, it would seem like the most elegant solution to implement event recognition logic in the existing system. Prolog is a quite difficult language to use, as it employs the logical programming paradigm. This paradigm enables powerful text-recognition or pattern-recognition to be implemented as stateful rules. A well known example of this feature is implementation of natural language grammar.

Likewise is it possible to construct a "grammar" for the TQL-code we wish to process. This grammar would consist of several rules for patterns we wish to identify in the TQL-code. It will be satisfied in an depth first-wise manner. A rule may "bind" some value to a variable, and may be used in subsequent patterns. If this binding produces patterns that has no candidate, Prolog will automatically backtrack the binding, and try other possibilities. Indeed an elegant method, but alas, difficult to implement.

The implementations is a module that can be plugged into GeneTUC as is, and in general terms tries to bind a variable containing TQL to series of event recognition rules.

One of the major problems we encountered were accessibility of data. Often when programs are implemented in Prolog, data is only "in scope" locally. That is, the execution "discovers" the data, prints them to standard out, and carries on with the execution. Unless that data are required at later stages in the execution, they are not bound to any variable, and thus not accessible. Implementing this binding later is at least difficult. It would require all related rules to be redefined to carry the variable on. (An alternative is to use globals, but this is a malpractice which should be avoided, since it breaks the depth-first property of the paradigm.)

This problem was also significant while producing the extraction patterns. For most patterns, 2-3 traversals of the TQL code were enough to extract the key data. But some patterns required more traversals, and intermediate construction of new patterns. This required each traversal to be fed it's own variable with original TQL, which iteratively would have to be "handed down" from the initial rule call. This complicated matters exponentially, and lead to suggestion of other programming languages.

Prolog was thus abandoned because it was to complicated to implement new rules and additional pattern in existing rules. Example code included in Appendix A.2.1, and output from the prototype is shown in Table 5.1

```
E: phorbol ester reduces constitutive nuclear nf kappa b and inhibits hiv-1.


% Semantic Evaluation Structure


[
 event(reduce,phorbol_ester,nf_kappa_b),
 event(inhibit,phorbol_ester,hiv-1)
]
```

**Table 5.1:** Output from the Prolog prototype

### 5.1.2 Perl

Perl is widely known for it's text-processing abilities, and has already been used extensively by Rune Sætre to process the output from GeneTUC. The straightforward accessibility of data provided by the while(<>) -statement[1], allows for rapid development of prototypes for text processing. The regular expression pattern matching functionality is perfect for finding predefined patterns in TQL code, as well as extracting TQL from GeneTUC's output.

The con of not using Prolog, is that all of the knowledge GeneTUC already has, cannot be employed in the extraction process. Such knowledge could be to find the root-form of a verb, if a word actually *is* a verb, what class it belongs to in the ontology etc. Consequently, TQL-code has to be self-contained with all the data we need. Because TQL is used by other TUC-projects than GeneTUC, e.g. BussTUC, BUSTER, TELEBUSTER, TELETUC, the TQL cannot be changed.

A Perl-specific problem is that writing OO-code is a nasty affair. In this case, objects are used to contain data in the way it is produced by GeneTUC (and GENIA). See Figure 5.1 for an overview of the abstraction. As shown, there are quite a number of classes, and most of them exist primarily to hold data.

---

[1]takes a line of input from either stdin or a specified file

The Perl-prototype were able to extract those patterns that were identified in the previous chapter. Developing new patterns proved difficult, as well as tracing execution of rules for debugging purposes. The difficulties experienced can be attributed to Perl being syntactically compact.

Example code from this program is included in Appendix A.2.2, and a typical output is shown in Table 5.2.

```
E: these results are in striking contrast to the increase in nuclear nf kappa
skolem[397] = result
skolem[398] = contrast
skolem[399] = increase
skolem[401] = importance
skolem[402] = monocyte

-- event: 111
 type = positive_regulation
 theme = increase
 cause = esters

-- event: 112
 type = importance
 theme = regulation

-- event: 113
 type = importance
 theme = hiv_1_
```

**Table 5.2:** Output from the Perl prototype

### 5.1.3 Java

Java is more "user friendly"' to work with than Prolog and Perl - except for certain usages, such as Natural Language Processing. Pattern recognition and NLP have some common ground, e.g. NLP is sort of stateful pattern recognition on syntax. There are two useful features in Java that allows us a modus operandi close to Perl, and thus the functional paradigm that is preferred for text processing:

- **java.util.regex.***
  The regular expression package in Java, consisting of Pattern and Matcher, providing full-fledged regex, including matching groups.

- **for ( Class c : Collection<Class> )**
  The new syntax for for-loops in Java 1.5 (5.0) allows for simple traversal of Collections, i.e. ArrayLists. Using new generics allows us to retrieve objects from Collections without having to cast them, resulting in prettier code and less runtime exceptions. This makes traversal of lingual expressions easier.

Implementing pattern-matching rules in Java required a whole lot more lines of code than for the other prototypes. This is attributed to Java being more verbose, and having less syntax for expressing intent implicitly. On the other hand, rules become easier to write and understand.

Both Prolog and Perl -programs were close to ∼150 LOC[2], while the Java-prototype consisted of ∼1000 LOC (having approximately the same functionality). The advantage of Java is clear at this point: Implementing further rules that partially is dependant on previous rules is only slightly more difficult than implementing the first rules. Besides, tracking changes and impact on existing rules is also feasible. Getting and storing data captured by the rules is simple by using objects as data containers.

Example code from the Java prototype can be found in Appendix A.2.3, and example output is shown in Table 5.3.

```
<Event: 1:2, type: binding activity, theme: t._factor_nf_kappa_b, cause: consensus_sequence>
<Event: 2:2, type: enhance, theme: activity, cause: interleukin_2_>
<Event: 2:6, type: coding, theme: nf_kappa_b_transcription_factor, cause: null>
<Event: 1:1, type: translocation, theme: null, cause: null>
<Event: 1:4, type: inhibition, theme: null, cause: null>
```

**Table 5.3:** Output from the Java prototype

### 5.1.4 Problems encountered

This section deals with problems encountered in the full-scale implementation in Java.

**Missing composite nouns**

When an expression is parsed by GeneTUC, it is first tested against the dictionary (semantic database) to determine if it is a complex noun. Such nouns have special meaning that one is unable to derive from the constituting words. I.e. "fast gun" should not be interpreted "a gun that is fast", but rather "a person that earns his living by offering assassination services". If there are no such definition, GeneTUC parses the expression with a composition of adjective + noun.

This is linguistically correct, but it is of less semantic information value. The problem becomes even more difficult in biochemistry; i.e. *"binding activity"*, which certainly can be interpreted as *"a activity that is binding something"*. Whereas the composite nouns belong to classes that can be used to identify event-associated nouns (*"some activity"*), those that are not predefined is much harder to decide for. Should a *"harsh activity"* initiate an event? Additional, nouns may have multiple adjectives, of which only a subset may constitute the composites, i.e. *".. cells influence constitutive or induced NF-kappa B translocation"*, where *"cells"*, *"influence"*, *"constitutive"*, *"induced"* and *"nf_kappa_b"* all are parsed as adjectives to *"translocation"*! Should we maintain a list of appropriate adjectives, or nouns considered as adjectives in a setting, which we are to allow as theme for an event?

Probably not, so we decide to identify events that are explicitly expressed. But then interestingly, it seems as if in many such cases, the rightmost adjective (most tightly bound, in this case nf_kappa_b) is the one sought for event theme. Investigation should be undertaken if event-extraction from TQL proves to be useful.

---

[2]Lines Of Code

**Cascading events**

Often, events build up in layers where one event serve as the cause for the next. If we are unable to identify the root-event, related events becomes even harder to find, since there will be sort of a backreference that we do not know of. Because of this, we should bias our focus a little towards identifying root-events. In addition, we should consider this problem when analysing the results.

**Ambiguous interpretation**

TQL code describes one interpretation of a sentence, heavily based on the way the sentence was parsed. However, it often occurs somewhat ambiguous sentences that is interpreted in an "unfortunate" way, rendering the extraction of events difficult. Again, missing composite nouns may result in this problem.

**Lack of abstraction**

Even though our initial test suggested so, the abstraction provided by GeneTUC is not yet adequate. See Table 5.4 for TQL of the two examples: "p50-coding exist" and "coding of p50 exist". These two simple sentences "should" produce the same TQL code. However, their interpretations have important differences: P50 is interpreted as a noun and an adjective. While P50 certainly is a (proper) noun, interpreting it as an adjective is in some cases grammatically correct.

Another problem exist with usage of nouns as adjectives. Consider the example *"p50 gene coding"*. Now there are two (noun-)adjectives, of which "gene" bind most tightly. It is certainly correct to apply the most tightly connected adjective to make the interpretation "coding of gene". But it is "coding of p50" that is the interesting information. GeneTUC should somehow be able to represent this distinction.

**Multiple themes**

Some interactions, i.e. bindings, has two "actors" or "themes". Consider this (real) sentence, where actual nouns have been replaced by letters A to F: *"A acts on B by enhancing binding activity of C to it's D in the E of the F"*. GeneTUC sort of uses narrow scoping and sees *"binding activity of C to it's D"*. This leads to an event of type binding, of C by D. However, a binding should not have a cause. This contradicts earlier interpretations of the nrel/of and nrel/by -relations, and poses a problem. A way of solving this would be to encode a special treatment of such relations in the context of "binding". As previously mentioned, we try to avoid encoding semantic knowledge in the evaluation system, and such events will therefore cause a loss of score. (GENIA suggest this to be interpreted as an event of type binding, of C and F (!))

**Synchronising events**

Some sentences may contain a lot of events that is based on a single "root"-event. When only some of these events are extracted, we are faced with a problem regarding what events to compare. Think of this problem in the terms of "least common substring" á la events.

```
E: Coding of p50 exist.

..................................
% TQL:

sk(1)isa coding
p50 isa protein
nrel/of/coding/thing/sk(1)/p50
exist/sk(1)/sk(2)
event/real/sk(2)
..................................


E: p50-coding exist.

..................................
% TQL:

sk(4)isa coding
adj/p50/sk(4)/real
exist/sk(4)/sk(5)
event/real/sk(5)
..................................
```

**Table 5.4:** Lack of abstraction

```
<event id=''E2''>
 <type class=''Binding''/>
 <theme idref=''T12''/>
 <theme idref=''T14''/>
 ...
```

**Table 5.5:** Multiple themes

Unfortunately, we are not able to solve the problem even using a polynomial method, since our "substring" may be incorrect.

## 5.2 Extraction method

To clarify exactly how "event extraction" works, this section contains a short summary.

Lingual expressions which use different words and syntactic constructs, may still have the same intended meaning. This meaning can be represented in event logic, and the differences are thus removed to some extent. By observing how typical expressions that contain one of more events are represented in relational logic, it is possible to construct patterns that capture events.

We have hand-crafted rules in Java that is using regular expressions to define such patterns. By also implementing an array of constraints that has to be satisfied for each capture to complete, we are able to filter the interesting biological events from those of no significance.

Currently the effort has been spent on identifying the events, so those constraints which has to be satisfied to "initiate" an event are rather strict. There are three overall event-initiating statements:

1. something isa interesting_process

2. interesting_verb/subject/object/id

3. event/world/id

The constraints applies to *interesting_process* and *interesting_verb*, which must be of some biological character, i.e. *activation* or *bind*. These "clue-words" have been extracted from a small set of event-annotated abstracts by GENIA. The event-statement is a method in GeneTUC of treating modality, but can also be used to track biological events.

When an event has been initiated, a larger set of rules is iterated over each set of TQL-statements, in pursuit of candidates to fill the "theme" and "cause" slots. "Type" is usually implicitly given by the clueword that produced the initial match.

See also Figure 5.3.

## 5.3  Event evaluation

Having implemented successful event recognition logic in Java, subsequent evaluation will also have to be implemented in the same language (to access the data contained in Java objects). Considering that this evaluation task is a lot more straightforward programming than event extraction, Java is preferable from all points of view. The next section describes how our results were evaluated. Then the numeric scoring method is presented. The problems mentioned in Section 5.1.4 are treated as outlined in Section 5.3.3.

### 5.3.1  Evaluation method

An illustration of the method is shown in Figure 5.2. To evaluate the events in a fair way, it is important to emphasise what we really are testing. The focus of this project is testing understanding of parsed text. This requires us to filter out all of those sentences that did not parse. Comprehension of sentences are the responsibility of grammar and dictionary. These factors are basic, and easy to test alone. The first step of evaluation is thus to eliminate sentences that are outside our scope.

This leaves us with a collection of Abstracts containing (interesting) Sentences containing Events. Although not all of these events are recognised by GeneTUC, they all have to be registered to achieve correct recall score.

Next, the events from GENIA that do not have a corresponding GeneTUC event are removed. This is a preemptive action to simplify comparison. It leaves with two arrays of events, where pairs corresponds to each other. Now, we can evaluate the content of each event.

The content is affecting precision scores. There are two "levels" on which we can measure precision:

- **event**
  For those events recognised, how many of them are have all parameters correct? This number will be referred to as "event precision". *(type && theme && cause)*

- **event attribute**
  For all attributes in recognised (and possibly incorrect) events, how many of them are correct? This number will be referred to as "attribute precision" *(type || theme || cause)*

Primarily, the most interesting precision if that of complete events. Only these can be used for practical applications, as discussed in Chapter 7. However, this score is also most affected by lack of event extraction rules. It is clear that such rules should not only be crafted manually. This matter is also discussed in Chapter 7.

Isolated scores for type, theme and cause may be useful to point out strengths and weaknesses in the represented semantic knowledge. One could also measure conditional precision scoring, i.e. percentage of correct theme given correct type, or correct cause given correct type and theme. Eventually, an Evaluation-object containing all the scores is created and stored for usage in the grand total.

(The scores may be subject to last minute change, as they are to be used in an upcoming article.)

### 5.3.2 Scoring

Recall, precision and F-score were first presented in Section 2.3. We briefly recall them:

- **Precision: TP / (TP + FP) = (Correct guesses) / (All guesses)**

  - Event Precision: TP of events / (TP + FP of events)
  - Attribute Precision: TP of attributes / (TP + FP of attributes)

- **Recall: TP / (TP + FN) = (Correct events) / (All actual events)**

- **F-score: 2 * P * R / (P + R)**

B-weighted F-score is not used in the evaluation. The abbreviations mean:

- **TP = True Positive (correct mapping)**

- **FP = False Positive (errorous mapping)**

- **NP = Not Positive (no event)**

- **FN = False Negative (no mapping)**

### 5.3.3  Treating problems

**Missing composite nouns and recognising synonyms**

When nouns are defined differently in GeneTUC and GENIA, their lexical representation differ. There may be events that are semantically similar, and yet lexically different; Thus, impossible to evaluate by comparing two strings lexically. A list of synonyms (and antonyms) should be created, so that the user (optionally) could answer questions of the similarity of events. These answers must of course be retained between sessions, be exchangeable between different users, and possible to construct by other programs than the Evaluation-program (i.e. an ontology-parser).

**Interpretation-related problems**

If the logic is incorrect or ambiguous, no events should be extracted. Such condition is difficult to determine. Occurrence of errorous events will penalise us by reducing the precision score. Spurious events could be discovered by evaluating a carefully crafted set of logic statements and annotations (supposed to score 100%).

**Multiple themes**

This problem is solved by a workaround. It does not affect the scoring. Events with multiple themes does not have cause. To avoid overwriting previously extracted themes with new themes, we rather store a second theme as cause.

**Synchronising events and cascading events**

A problem occurs when the number of events that have been extracted differs from the number of events in the gold file. This either means that we have failed to recognise an event, or identified a non-existent event. The latter are more common than the former. This problem was also encountered in the previous project (described in [Søv05]). It will be dealt with by having some guidelines:

- During evaluation, one should evaluate as small units as possible, e.g. (events in) sentences, and not (events in) abstracts.

- Event extraction should be optimistic, so that:

- When two events do not match, one should retry comparison with another event, found by using some heuristic.

- And finally, one should be able to "synchronise" events manually, by using two text files[3].

A simple heuristic were implemented, that would "skip ahead" to find a more appropriate event to compare with. But only if it leaves enough GENIA events to compare to the remaining GeneTUC events. A "match" is defined by two events having equal types. This improved the type precision on the training set by 11%.

---

[3]Manual synchronisation was successful in [Søv05]
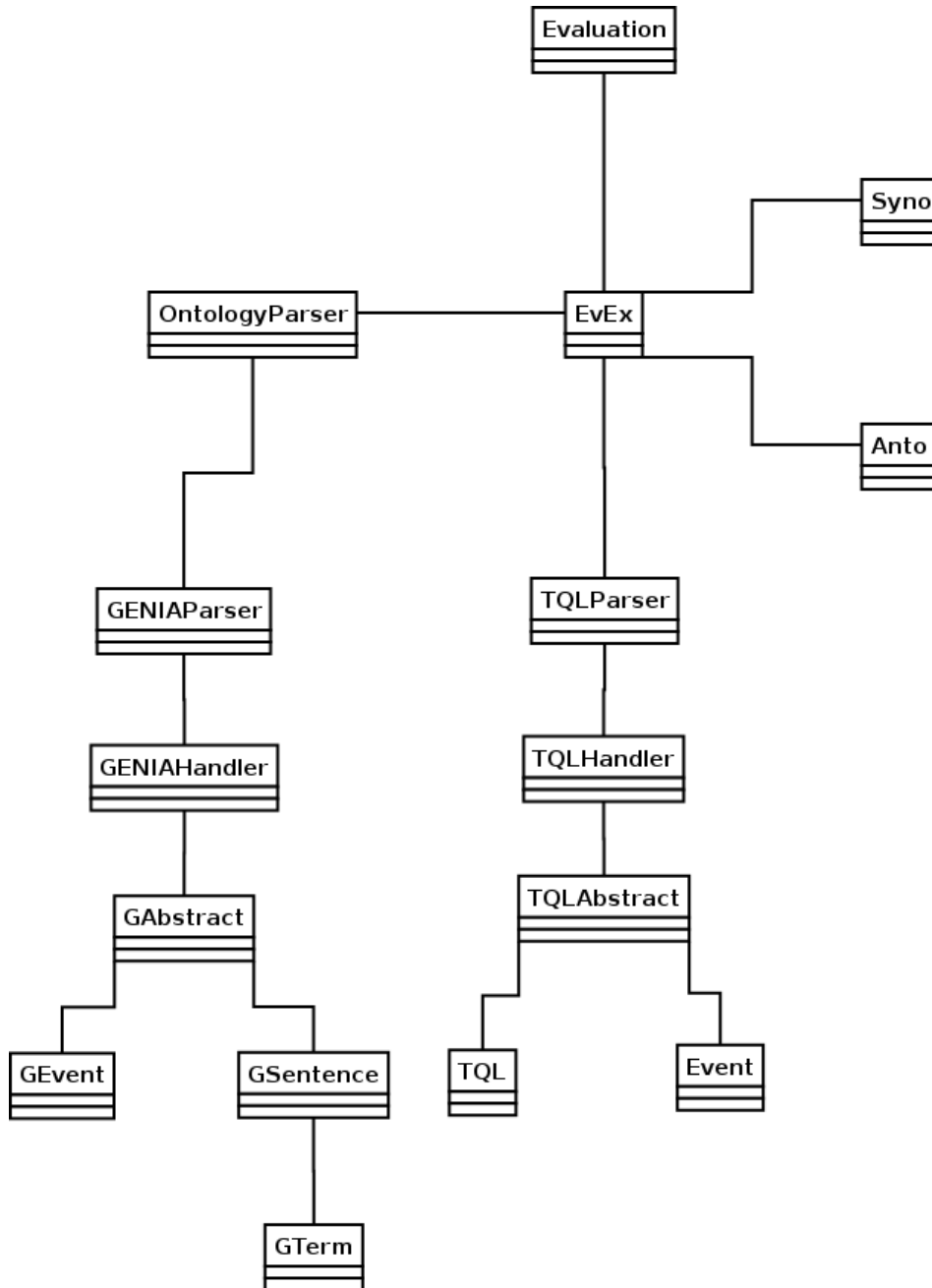
**Figure 5.1:** UML class view
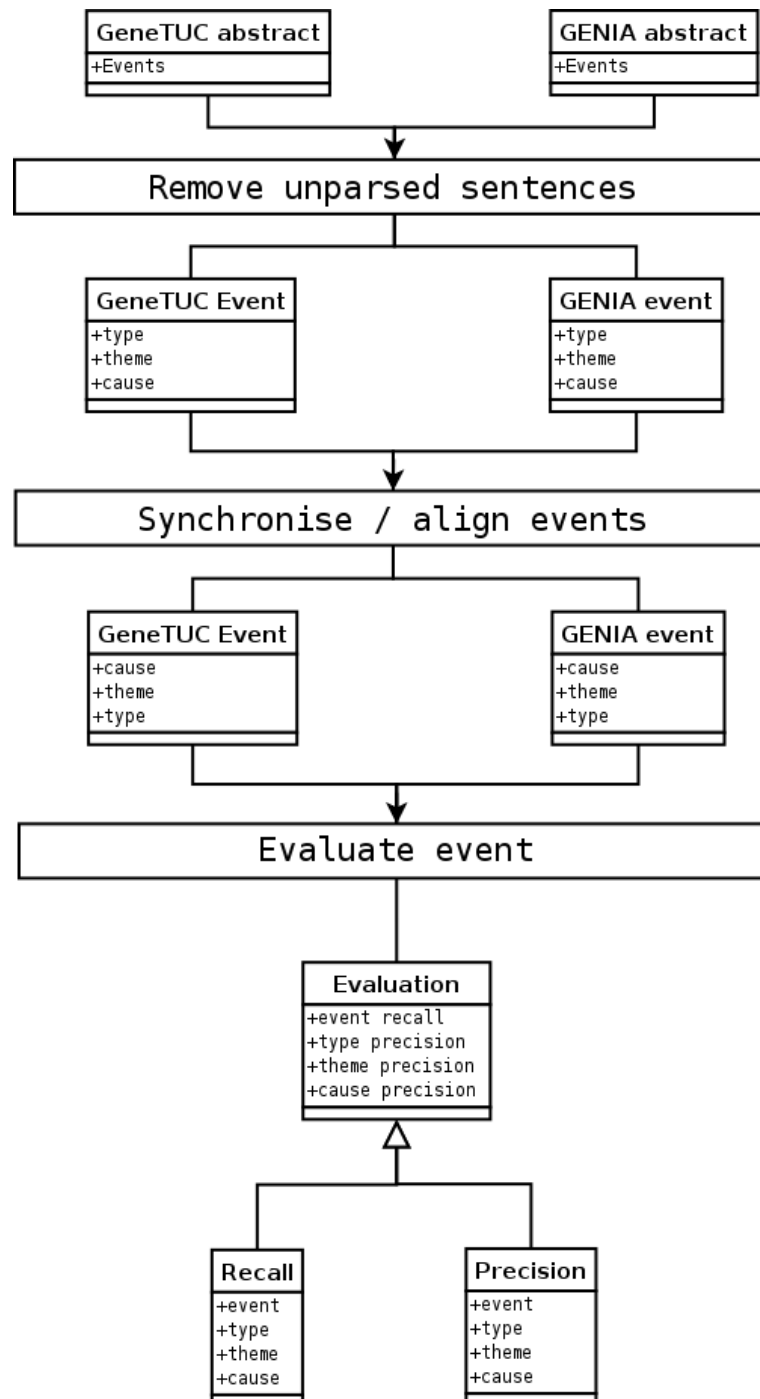
**Figure 5.2:** Abstract overview of evaluation

```
  ========= EXTRACTION RESULTS =========
Total abstracts in file;      TQL: 16,        GE: 16  (100%)
Events in all abstracts;      TQL: 44,        GE: 393 (11%)
Sentences in all abstracts;   TQL: 73,        GE: 143 (51%)


     === WIDE EVALUATION RESULTS ===
Recall of events:             44/169  (26%)
Type precision:               17/169  (10%)
Theme precision:              6/169   (3%)
Cause precision:              9/169   (5%)


     === NARROW EVALUATION RESULTS ===
Recall of events:             44/44   (100%)
Type precision:               17/44   (38%)
Theme precision:              6/44    (13%)
Cause precision:              9/44    (20%)
```

**Table 5.6:** Evaluation results (illustrative)
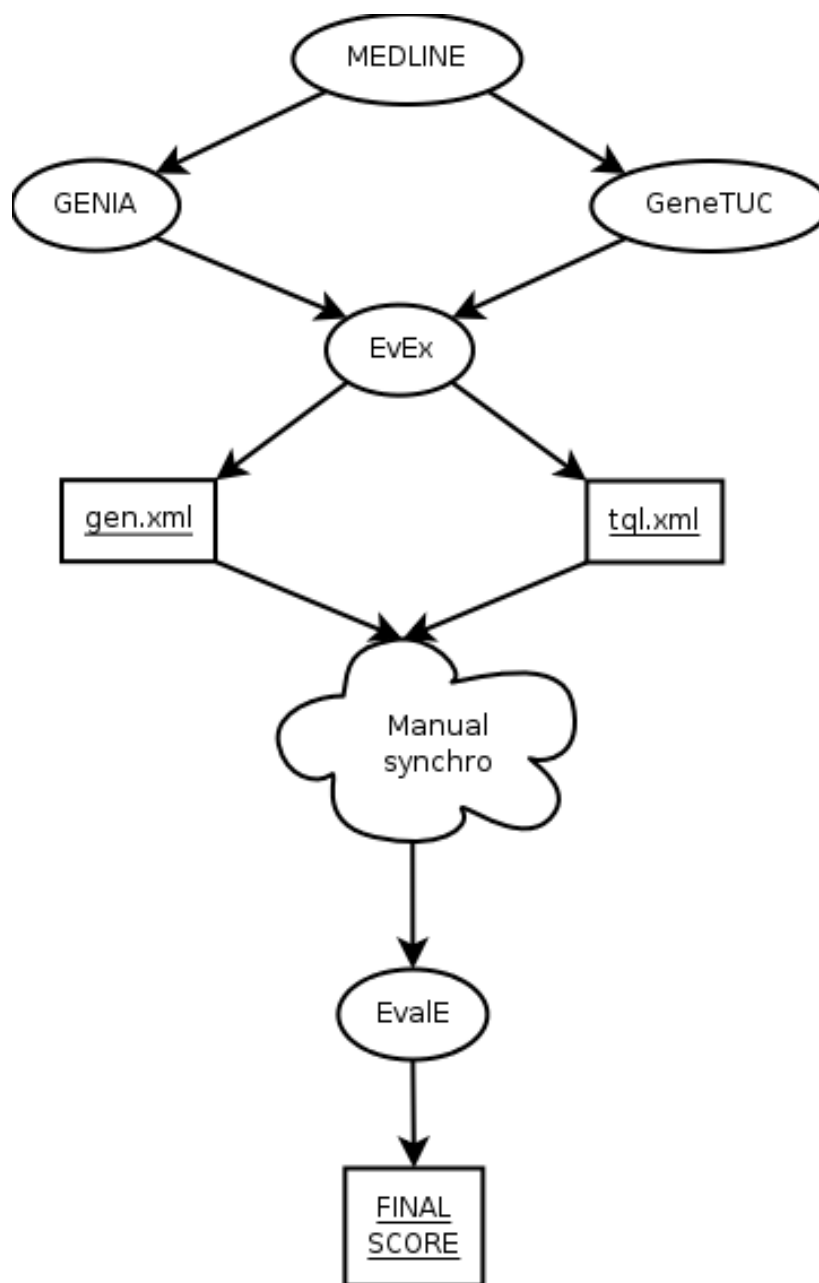
**Figure 5.3:** Overview of solution

# Chapter 6

# Results

This chapter contains the results from event extraction and event evaluation performed by the software described in Chapter 5. The extraction logic was trained (i.e. taught) with the training data, then evaluated against the test data. Two sources of data has been used: GENIA (for annotated abstracts) and PUBMED (for plain text abstracts). GENIA provided 19 annotated abstracts, from which we randomly selected 16 (8 for testing and 8 for training). The PUBMED identification number for these abstracts are listed in Table 6.1 and Table 6.2.

We expect that these results will show if it is feasible to extract biological events from GeneTUC by using hand crafted mapping rules. At the least, a recall of $\sim$30% (one event per sentence) should be expected. Precision in training should amount to an average of $\sim$30% as well. If the extraction is feasible, the average precision in testing should not be significantly less.

The results are discussed in Chapter 7.

| Training |
|----------|
| 1419905  |
| 1431113  |
| 1464736  |
| 1482376  |
| 1493333  |
| 1502202  |
| 1505523  |
| 1527846  |

**Table 6.1:** Training abstracts

| Testing |
|---------|
| 1527859 |
| 1531086 |
| 1533884 |
| 1583734 |
| 1618911 |
| 1653056 |
| 1655897 |
| 1668145 |

**Table 6.2:** Testing abstracts

## 6.1 Training data

Our training data consisted of 8 abstracts, where 35 (44%) out of 79 sentences were parsed successfully. In these 35 sentences, we were able to identify 62 events (67%). 92 were identified by GENIA. (Table 6.3)

10 (16%) of our 62 events were identified with correct type, theme and cause. (Table 6.4)

Overall, 62 (33%) out of 186 attributes were correctly identified, of which 25 (40%) were "type", 14 (22%) were "theme" and 23 (37%) were "cause". (Table 6.4)

These figures results in a F-score of 0.44220003 using attribute precision average, and 0.25831324 using event precision. (Table 6.5)

### 6.1.1 Processing statistics

| Processing statistics | | | |
|---|---|---|---|
| | % | TQL | GE |
| Total abstracts in file; | (100%) | 8 | 8 |
| Events in all sentences; | (29%) | 62 | 212 |
| Events in parsed sentences; | (67%) | 62 | 92 |
| Total number of sentences; | (44%) | 35 | 79 |

**Table 6.3:** Processing statistics, training data

### 6.1.2 Precision and recall

| Precision and recall | | |
|---|---|---|
| Event precision: | (16%) | 10/62 |
| Attribute precision (type): | (40%) | 25/62 |
| Attribute precision (theme): | (22%) | 14/62 |
| Attribute precision (cause): | (37%) | 23/62 |
| Average precision: | (33%) | 62/3x62 |
| Event recall: | (67%) | 62/92 |

**Table 6.4:** Precision and recall, training data

### 6.1.3 F-score

| F-score for | |
|---|---|
| Harmonic F-score (attribute): | 0.44220003 |
| Harmonic F-score (event): | 0.25831324 |

**Table 6.5:** F-score, training data

## 6.2 Test data

Our test data consisted of 8 previously unseen abstracts, where 37 (57%) out of 64 sentences were parsed successfully. In these 37 sentences, we were able to identify 22 (28%) events. 76 were identified by GENIA. (Table 6.6)

3 (13%) of our 22 events were identified with correct type, theme and cause. (Table 6.7)

Overall, 18 (27%) out of 66 attributes were correctly identified, of which 8 (36%) were "type", 3 (13%) were "theme" and 7 (31%) were "cause". (Table 6.7)

These figures results in a F-score of 0.2749091 using attribute precision average, and 0.17756097 using event precision. (Table 6.8)

### 6.2.1 Processing statistics

| Processing statistics | | | |
|---|---|---|---|
| | % | TQL | GE |
| Total abstracts in file; | (100%) | 8 | 8 |
| Events in all sentences; | (12%) | 22 | 181 |
| Events parsed sentences; | (28%) | 22 | 76 |
| Total number of sentences; | (57%) | 37 | 64 |

**Table 6.6:** Processing statistics, test data

### 6.2.2 Precision and recall

| Precision and recall | | |
|---|---|---|
| Event precision: | (13%) | 3/22 |
| Attribute precision (type): | (36%) | 8/22 |
| Attribute precision (theme): | (13%) | 3/22 |
| Attribute precision (cause): | (31%) | 7/22 |
| Average precision: | (27%) | 18/3x22 |
| Event recall: | (28%) | 22/76 |

**Table 6.7:** Precision and recall, test data

### 6.2.3 F-score

| F-score | |
|---|---|
| Harmonic F-score (attribute): | 0.2749091 |
| Harmonic F-score (event): | 0.17756097 |

**Table 6.8:** F-score, test data

## 6.3 Comparison

Table 6.9 compares the key results. The difference in numbers of parsed sentences is significant, but the direct effect has been ruled out. Precision decreased slightly from training to test. Recall were significantly reduced.

| Attribute | Training | Test |
|---|---:|---:|
| Parsed sentences | 35/79 (44%) | 37/64 (57%) |
| Event precision | 10/62 (16%) | 3/22 (13%) |
| Average attr. precision | 62/186 (33%) | 18/66 (27%) |
| Event recall | 62/92 (67%) | 22/76 (28%) |

**Table 6.9:** Comparison table, both data sets

# Chapter 7

# Discussion

This chapter will provide a discussion on the choices that have been made and results that have been obtained during the project. The first section (7.1) provides a review of the research period. Section 7.2 reviews the implementation of prototypes, extraction- and evaluation software. Results are discussed in Section 7.3. Some of the content in this chapter has been described earlier, but is now discussed in a retrospective view.

## 7.1  Background research

As previously mentioned, this project is a sequel to an autumn-project in 2005. The work in this project were preparatory, but it was not possible to refactor the result to fit this project. It produced two papers: [SSAT05] and [Søv05]. Apparently, the most valuable heritage was the knowledge of how GeneTUC is built and can be built on. Tacit knowledge on evaluation of parsed text also helped avoid some of the pitfalls discovered earlier.

Having a good understanding of syntax, it was logical to dive into the complex world of semantics. Thus this project started with a throughout study of [dS98], miscellaneous papers and websites on semantics. This way not directly necessary, but having a deeper understanding on why things are as they are always seems like a good idea. At least when one are about to decide which semantic representation is most appropriate for our use.

This theory dig surfaced with three alternatives to be evaluated. The most theoretical and basic of those, Predicate Argument-Structures, is treated in a lot of scholar texts. The other two - Gene Ontology Annotation and GENIA events - are still subject to research. Most information on them had to be found in research articles. Extracting theoretical knowledge from such articles were labourous, but at the same time very rewarding. At this point, the background study on semantics paid off.

After having decided on using GENIA Events (which will be discussed later in this chapter), a study of events were carried out. Semantic Events is currently a hot topic within NLP research communities. Proper background knowledge from linguistics seemed not only appropriate, but also necessary. It proved that events had been around for a long time, but most effort had been on capturing complete semantics in an event based structure. GENIA and other current projects focus on representing *the interesting* semantic information. By reducing the facts to be represented, one also runs the risk of creating a too-specific system. Making too many presumptions and covering a too small domain could render a NLP system useless. Fortunately,

the decision on whether or not a statement is important is left to biologists - which seem to be content with events.

Extraction of protein-protein interactions is judged to be very important in molecular biology. It is believed that determining all the interactions a protein participtes in is an important stepping stone. This justifies focusing on such a narrow domain.

The connection between "traditional" linguistic event theory and cutting-edge GENIA events are not quite clear. On one hand, GENIA focuses on a narrow domain: *"covering the NFkB pathway"* (read: interactions of a particular protein). By registering clue words (typically verbs or verb phrases) in the text, one is able to derive the class of an event (e.g. activation). Traditional event theory argues that events should not be bounded by syntax, but treats classification of events as task that is to be carried out with abstract tools, e.g. filling slots in an grammar sequence and classifying an event "continuous and bounded". On the other hand, both approaches seek to represent events approximately the same way. Thus, it is not straightforward to explain GENIA events theoretically. Luckily, they are very intuitive.

In total, the background research may have been involved a lot of unproductive studies, but it is difficult to argue that some of the topics could have been left out. Because it was important to verify that events really were the best choice, or - if not - what the alternatives were. In this tradeoff, having covered a topic a little too broadly is better than having covered it a little too deeply - one could always dig a little deeper if it is deemed necessary.

## 7.2 Extraction and evaluation

While the background research may have been somewhat abstract, the tasks to be carried out were very concrete. The focus is now shifting from *what* to do, to *how* to do it.

Prototype development was an effective way of discovering what the difficulties of different approaches was. Since these programs were intended to be used only a few times by the developer, the user interface was not of great importance. This enabled focus on developing the logic core first, and then tying in usability functions later. A weakness in the prototyping process was that we had not decided what the terminal conditions should be. All programs were developed with the superficial goal "to construct something that demonstrates event extraction in this paradigm/language". It would probably have been better to have tangible goals of "extracting these X events", and then estimate the effort needed to extend the system to have full-scale extraction abilities. Lack of such goals lead to a too broad focus on trying to solve too many problems at the same time, and thus none were actually solved.

### 7.2.1 Prolog

Prolog were the most complicated case, and spending much time on developing such prototype was probably not justifiable. Experiences from the prior project indicated that implementation of representation mapping in Prolog were feasible. However, that project mapped a finite set to a finite set. This project turned out to be about mapping a finite set of predicates *and* an *in*finite set of arguments into a set of events. Declaring mapping functions were thus not as easy as first expected. In conjunction with the difficulties with data access and traversal of data (described in Chapter 5), the problem became complex and difficult to divide into solvable sub-problems.

The prototype were finalised when a very basic recognition logic had been implemented. It were able to identify a very small subset of the existing events, but expanding this set would require too much effort. Rules becomes too complicated to identify manually and represent explicitly. An eventual Prolog-approach, should thus consider solving the problem by other means than pattern rules.

### 7.2.2 Perl

Perl was a long leap: From the logic- to the functional paradigm. Although it had proved to be appropriate in other applications related to GeneTUC, prototyping suggested that we would run into much of the same difficulties as with Prolog. Not any like those related to accessing data, but general difficulties with keeping rules up to date and implementing new rules. Complications with tracing program execution and debugging were a major influence in the decision of terminating the prototype early. Admittably, the problems have to be attributed to the skills of the programmer. An experienced Perl-programmer would have been able to implement a great deal more rules. However, the problems suggest that there eventually will be infeasible to insert more rules and still keep track of execution.

The prototype were finalised with approximately the same functionality as the Prolog prototype, although it had taken less than half the effort. Since we now have to employ external programs for extraction, it may be a good idea to consider multiple alternatives. Taking the full step to using high-level languages certainly introduce overhead, but at the same time possibilities - such as creating a nice GUI that may be used by biologists or other curators to assist evaluation, providing feedback to the extraction system, improving rules etc., in fact a complete software suite offering both supervised and unsupervised learning of extraction rules. But there are more pressing matters to attend to in development of GeneTUC, and such suite will not be really useful until basic matters are solved.

### 7.2.3 Java

Java were first suggested as a wrapper for GeneTUC. There are packages for Java that enables interfacing with Prolog. Such program could ease the manual labour of creating a dump-file for GeneTUCs results, filter this for TQL, manually invoke an extraction program, and so on. However, the one that were tested - Jasper - had very narrow interface functionality. It executed unification of a single predicate. Executing GeneTUC in this fashion, would have required rewriting of the main program. A revolutionary turn of this project would not be appropriate with respect to risk and available resources.

Thus, a prototype were implemented that could read XML-separated TQL-code, extract events by pattern matching and ontology querying, all of which is described in Chapter 5. Implementing further rules and evaluating them were possible since the rules became very expressive in Java code. Extraction were greatly simplified when the evaluation software also had been implemented. With this module functional, one could change or implement a rule, and observe the change in precision, recall and f-score.

One matter still remained. The rules had to be figured out manually. If the case is that biologists should be able to curate rules, they would have to be represented in another way (than program code). If one were to create a graphical representation of the TQL code, e.g. as a semantic net, biologist could use a pointing device to mark up certain connections that relate to an event. The decisions could then be generalised, and automatically interpreted as rules.

Another option would be to employ machine learning methods to learn event patterns. An elaboration of this idea follows later. It is important to keep the two ideas separate. One would learn *rules for extraction* (without domain knowledge), or one could learn *rules for identifying biological event patterns in event logic* (using domain knowledge).

### 7.2.4 Conclusively

Conclusively, an overall distorting factor in prototype development have been uncertainty of what the product were to be used for. Event extraction for the sake of being able to produce events is most valuable in the future, both for evaluation and corpus/knowledge-sharing. On the other hand, having indicating evidence of the current state of GeneTUC is very valuable for usage right now, considering upcoming deadline for submitting the paper (Appendix B).

As for now, the events are created for short term use. The events produced can be said to have been "normalised" by GeneTUC. That is - they have been represented in a generic way. Next, we should investigate further on how the other events can be discovered. This report will provide useful input to later projects trying to fulfil long-term goals of events in GeneTUC.

## 7.3 Discussion of Results

The results were obtained by first "training" the system using one data set, and then "testing" on a different one. The process of training was not the automatic one usually applied in machine learning, but rather manual "teaching" of the system. It would have been preferable to use an automatic approach, so that we could test the data using cross validation, i.e. training the system using different parts of the dataset, and testing against other parts, averaging the results.

By using this (possibly) too simple test, one have to consider how it affects the statistics. Results are listed in Chapter 6. Detailed statistics of training and test data includes a section of "processing statistics". These are a summary of overall results. "Sentences in all abstracts" represent the parsing success of GeneTUC. 44% success with training data and the slightly higher 57% success of test-data. The difference is incidental, and most likely caused by the small number of abstracts used.

This difference *may* lead to a bias in the results. However, not because the total count of sentences differ. Those sentences that GeneTUC were unable to parse are removed from further evaluation and so is the corresponding sentences and events from GENIA. It is clear that complicated sentences often contains more events than simple sentences, and often such events that are repeated with different granularity. Such complicated sentences are less likely to be parsed by GeneTUC, and one could thus say that GeneTUC "favours" simple sentences - containing fewer events - and thus renders those events too significant. See Section 8.4.3 for a discussion of why GeneTUC are unable to parse some sentences.

A strong indicator for these data being too sparse is revealed by the total number of events identified by GENIA in those sentences parsed by GeneTUC. For the training data, GENIA had identified 62 events. However, the testing data contained only 22 events. There are no reasonable explanation for this skew, except that some abstracts may be "harder" to parse because of the authors language. Thus should a larger number of abstracts be used for testing. Unfortunately, there are only 19 event-annotated abstracts available from GENIA.

### 7.3.1 Event precision

Event precision is the fraction of events that are similar or equivalent in all attributes. The number of events that are 100% equal is much lower, but application of a synonym-dictionary justifies this comparison. The result is low, but steady (16% in training, 13% in testing). It is likely that event precision will increase fast when we are able to define more precise extraction rules. Until then, this score suffers from imprecise rules used in the extraction of an event, where one of the attributes are errorous.

### 7.3.2 Attribute precision

Attribute precision is given for each attribute in an event (type, theme and cause). Type scores 40% vs. 36%, theme 22% vs. 13%, and cause 37% vs. 31% (train vs. test). On average the score is 33% in training and 27% in testing. The decrease observed between training and testing for every attribute is natural due to the unseen event instances in the test set. It could be argued that the training results initially should have been close to 100%. This question will be treated in paragraph 7.3.4.

The attributes are extracted one-by-one, and there should be no covariation between them. From these results we conclude that our rules for identification of type, theme and cause are covering the base cases, but further steps have to be taken to increase precision. It is noteworthy that extraction of the theme-attribute was harder than extraction of type and cause. This may be explained by considering cause as "subject" and theme as "object". It is usually only one subject in a sentence, but there may be more than one object.

### 7.3.3 Event recall

Event recall scored 67% vs. 28% - a relatively drastic decrease considering that the precision scores were so similar. The low score of test data could be caused by the equivalent difference in actual events in those sentences - 62 vs. 22. It is clear that this difference should reduce our total number, but strange that this should decrease our percentage. A possible explanation is that many of these 22 events are closely connected, and thereby resulting in a larger fraction of "connected" events - which are harder to extract. A larger test set would have normalised this fraction.

### 7.3.4 Training precision and recall

Training precision and recall should arguably have been close to 100%, given adequate data. It is obvious that our training results were not close to that good. This was caused by two factors. First, creating rules manually was harder than expected. In addition, the training data did not become available until mid-May. Until then, 10 example events from a research article were the only data. If those 19 abstracts (containing ∼600 events) had been available earlier, a more machine learning-related approach would have been chosen.

### 7.3.5 F-score

F-score, respectively 0.44220003 and 0.2749091, is a relative measure, computed as a harmonic average between precision and recall. These two numbers alone tells us that the training set performed better than the test set, and that both of these had rather mediocre performance. They can be used to compare our results with other projects.

### 7.3.6 BioCreAtIvE

BioCreAtIvE 2004, task 2.2 was very similar to our event extraction, i.e. *functional annotation of gene products*. Or in other words markup of ontology terms with appropriate hash keys. The results displayed by various participants are close to those we achieved: up to 30% accuracy [BLKV05]. In BioCreAtIvE 2006, there will be a task even more similar to our extraction: *extract protein-protein interactions from free text*. (This is a subtask in task 3[1]).

Participation in this contest will be discussed in Section 8.4, Future Work.

## 7.4 Limitations and potential

It is a clear weakness that cross validation is not available. Repeated training/teaching is not possible with the current system. However, it must be pointed out that this system never was intended to be a extraction tool for events. It was intended to execute a mapping from TQL to events. As long as this mapping is incomplete, the events produced are un-representative for the knowledge of GeneTUC. If a machine learning paradigm is used, the extraction process contains knowledge itself, and is thus not what we try to achieve. (But it would have been attractive to develop none the less.)

Event extraction can be learned by a program and work in symbiosis with GeneTUC to produce e.g. Gene Ontology annotations and participate in BioCreAtIvE. Possibly can the scheme also be reversed, if events specified a priori can be used to assist parsing - and ultimately produce the semantic definitions that are needed by GeneTUC to parse the text perfectly by itself.

Events is a very general representation, and it is thought of as a possible cross-project platform for exchanging semantic data and evaluating systems against gold-standards. Such a corpus- and knowledge sharing would be of great assist to several research groups, especially those dealing with extraction of matters similar to protein-protein interactions.

The greatest poteintial right now, is participation in the BioCreAtIvE contest, which is discuessed in Section 8.4.2.

---

[1]http://biocreative.sourceforge.net/biocreative_2_ppi.html

# Chapter 8

# Conclusion

## 8.1  Aim

This thesis has investigated different semantic representations for measuring the degree of correct understanding in a NLP system (GeneTUC), and implemented an approach using biological events from the GENIA project. Close to every NLP project use their own format for representing knowledge. Measuring the quality of this knowledge presents a challenge, since a mapping into a common format has to be present.

Representations may have different bias, and thus represent different features of the language. Mapping from one format to another is thus a non-trivial matter. But if a mapping can be found, it would enable these two projects to share corpora, which is considered a major advantage in NLP research.

## 8.2  Result

Three formats have been examined: Predicate Argument-Structures, Gene Ontology annotations and GENIA events. Predicate Argument-Structures were found to be too oriented towards organising syntax structures without providing a real understanding of the represented text. On the other hand, Gene Ontology annotations were too high-level semantically oriented, requiring additional knowledge to be implemented in the mapping to produce interesting results. GENIA events was found to be suitable since they represent approximately the same information that can be found in TQL (knowledge language produced by GeneTUC).

A number of prototypes were implemented, and Java was selected as the most optimal language to implement mapping logic. Manual mapping-rules were identified, implemented and evaluated using a training-set of 8 abstracts. It was found that manual construction of such rules is infeasible, and that an automated machine learning approach would have been better. If the goal is to evaluate the NLP-system, such approach would have to guarantee that the learning phase only achieved knowledge on the extraction process - not the facts in the text and general domain knowledge. The matter is complicated, because GENIA events is biased towards representing biological interaction, whereas common lingual events have no such bias. Events are is considered interesting regardless of how they are produced.

We tested the resulting system on 8 previously unseen abstracts, and achieved a F-score of 0.1775

(13% precision, 28% recall) for complete and perfectly extracted events (having all attributes correct). In addition, we had an F-score of 0.2749 (27% precision, 28% recall) for extraction of event attributes (average correctness of all attributes regardless of event).

The results suggest that events may not be a completely adequate measure for semantic knowledge. Primarily because it is uuncertain if it is possible to map one-to-one between TQL and events. Secondarily because (GENIA) events are focused on protein-protein interactions only.

The events we are able to extract now, are comparable to acceptable results from the previous BioCreAtIvE contest. Some adapting and training with the (not yet released) BioCreAtIvE corpus will show if we truly will be able to participate.

## 8.3 Contributions to the field

The results indicate that event extraction from TQL is possible, but may be substantially improved by having an extraction system with additional domain knowledge. Such a system may also be trained to extract events having other domain focus than biological events.

Further development of GeneTUC may be assisted using events as a measure of semantic correctness. (As a tool to determine how a change to grammar or dictionary impacts the correctness of produced semantics.) It would however require certainty of correct results. We are not able to provide such guarantee until there exists a complete mapping.

GeneTUC and the event extraction software will be tuned to participate in the BioCreAtIvE contest. This year, there is a special task for *"Extraction of protein-protein interactions from text"*. The problem is characterised as *"one of the most pressing biological problems"*. Participation will be a great opportunity to demonstrate the versatility of GeneTUC.

## 8.4 Future work

This thesis has directly influenced three other projects: A paper to be submitted to Computational Linguistics, commented in Section 8.4.1. And participation in the BioCreAtIvE contest, described in Section 8.4.2. The future work on GeneTUC affected by this project is described in Section 8.4.3.

### 8.4.1 Computational Linguistics Special Issue

Computational Linguistics is a journal published by the Association for Computational Linguistics - *"THE international scientific and professional society for people working on problems involving natural language and computation"*. This[1] is a special issue on Semantic Role Labeling (SRL). We believe that biological events and SRL have many attributes in common, and are interchangeable to some extent.

We are thus to submit a paper which describes GeneTUC and the effort of this project. The paper is yet but a draft, and has been attached in Appendix B.

---

[1]http://www.lsi.upc.es/ carreras/srlcl.html

### 8.4.2 BioCreAtIvE 2006

BioCreAtIvE 2006 is about to release their tasks and training data for the next run. The most interesting task is discovery of *Protein-Protein Interaction* (PPI), which is described briefly at the BioCreAtIvE website[2]:

*The study of protein interactions is one of the most pressing biological problems. Characterising protein interaction partners is crucial to understanding not only the functional role of individual proteins but also the organisation of entire biological processes. (. . . )*

*Because the molecular biology literature provides detailed descriptions of protein interaction experiments specifying the individual interaction partners, as well as the corresponding interaction types, it has been exploited as a resource to derive protein interaction records for interaction databases. Due to the rapid growth of the biomedical literature and the increasing number of newly discovered proteins, it is becoming difficult for the interaction database curators to keep up with the literature by manually detecting and curating protein interaction information.*

The earlier mentioned BioCreAtIvE 2004 task (functional annotation with GO codes) were inappropriate because it involved usage of Gene Ontology. GeneTUC applies a different ontology created by GENIA. Crossreferencing these is thought to be a formidable task. However, the 2006 PPI task requires annotation of proteins using UniProt master identificator. It should be fairly easy to map the common name of a protein to this identificator using a database- or Google API.

More specific, the PPI task consists of 4 subtasks, where our primary aim is number two:

1. **Detection of protein interaction papers**
   The subtask is concerned with sorting papers. It is not necessary to discover protein interactions in non-biological texts. Thus, it is common to select a set of texts which are likely to contain interaction pairs, and discard the rest.

   Such task is outside GeneTUC's domain. Yan Hua Chen[3], Research Fellow at NTNU, will join the team with focus on this task.

2. **Protein interaction extraction sub-task**
   The subtask is to locate protein-protein interaction pairs in full text articles, and provide the pairs with corresponding gene mention symbols. GeneTUC and the event extraction program presented in this report will cooperatively solve the task.

3. **Best protein interaction description sentence detection**
   The subtask is to provide textual evidence of the extracted interaction pair. With some modification, the event extraction software will be able to provide the sentence in which the extracted event occurred.

4. **Protein interaction experiment detection sub-task**
   The subtask is to annotate the protein interaction pairs with the experimental method they were discovered with, according to a "controlled vocabulary". We have not yet determined if we are to participate in this subtask.

The result of our participation in BioCreAtIvE can be found in the proceedings and/or journal articles from BioCreAtIvE, due to be published September 2007.

---

[2] http://biocreative.sourceforge.net/
[3] yanhua@idi.ntnu.no

### 8.4.3 GeneTUC

GeneTUC is currently able to parse about 50% of the sentences it is presented (from the biology-domain). There are several reasons for why this number is not higher.

**Case and punctuation**

GeneTUC is currently ignoring the case of letters, treating them all as lowercase. This will lead to ambiguity in some situations, e.g. when mentioning protein names in (case-sensitive) shorthand notation. Adding case sensitivity would thus have a positive effect. Punctuation is ignored, except period, which are treated as a sentence delimiter. Ignoring e.g. commas may lead to ambiguity. Treating the periods in "Ph.D." as sentence delimiters leads to incomprehensible sentences. Punctuation would therefore have to be treated sooner or later.

**Syntax**

Facts may be stated in different ways, which sometimes affects the way they are represented in the knowledge base; And thus the way they can be retrieved. GeneTUC has until now been focused on parsing abstracts. Using full-text articles would probably lead to discovery of the same statement in different variations, and thus the differences would be captured.

**Semantics**

Protein names, experiment methods and other peculiar biological terms are not standardised in the way common English are. Different projects may use different names for the same thing. Projects like Gene Ontology is working on standardising such terms. Discovery and learning of a non-standardised terms in GeneTUC may lead to spurious entries in the knowledge base, and thus negatively impact the credibility of the knowledge base. It is important to adapt GeneTUC to e.g. Gene Ontology standards as soon as possible. A step in the correct direction, is participation in BioCreAtIvE, where all proteins have to be identified by their UniProt term.

# Bibliography

[Amb]     Tore Amble. Tuc tutorial. selje.idi.ntnu.no:/home/a/17/busstuc/GENETUC2/tuc_tutorial.txt.

[Amb01]   Tore     Amble.          *BussTUC.*          Team     Trafikk,     2001.
          http://www.idi.ntnu.no/ tagore/busstuc/.

[Amb04]   Tore Amble. *The Understanding Computer.* NTNU, 2004.

[And00a]  Anders Andenæs. *GeneTUC.* NTNU, 2000.

[And00b]  Anders Andenæs. *GeneTUC - An NLP System for Biomedical Texts.* NTNU, 2000.

[BB96]    Benua and Borer.  The passive/anti-passive alternation.  In *Paper presented at
          GLOW, Athens*, 1996.

[BLKV05]  Christian Blaschke, Eduardo Andres Leon, Martin Krallinger, and Alfonso Valencia.
          Evalutaion of biocreative assessment of task 2. *BMC BioInformatics*, 6, 2005.

[Bor94]   Hagit Borer.  The projection of arguments.  *Functional Projection, University of
          Massachusetts Occasional Papers*, 17, 1994.

[Bor96]   Hagit Borer. Passive without theta grids. *Morphological Interfaces*, 1996.

[CFP06]   Call for papers: Special issue of computational linguistics on semantic role labeling,
          2006. http://www.lsi.upc.edu/ carreras/srlcl.html.

[Con05]   Shared tasks - semantic role labeling, 2005. http://www.lsi.upc.es/ srlconll/.

[Dav67]   Donald Davidson. *The logical form of action sentences.* University of Pittsburgh
          Press, 1967.

[dS98]    Henriëtte de Swart. *Introduction to Natural Language Semantics.* CSLI Publications,
          1998.

[E.76]    Bach E., editor. *An extension of classical transformational grammar.* Michigan State
          University, 1976. Problems of Linguistic Metatheory.

[ea97]    J.R. Hobbs et. al. Fastus: A cascaded finite-state transducer for extracting informa-
          tion from natural-language text. In E. Roche and Y. Schabes, editors, *Finite-State
          Devices for Natural Language Processing*, pages 383–406, 1997.

[ea00]    Michael Ashburner et. al. Gene ontology: tool for the unification of biology. *Nature
          Genetics*, 25:25–29, 2000.

[ea05]    Gabor Melli et. al. Description of squash. In *Proceedings of the Document Under-
          standing Conference*, 2005.

[ESP⁺04]    Alphonse E, Aubin Sophie, Bessieres P, Bisson G, Hamon T, Lagarrigue S,
            Nazarenko A, Manine A, Nedellec C, Vetah M, Poibeau T, and Weissenbacher D.
            Event-based information extraction for the biomedical domain: the caderige project.
            *Joint Workshop on Natural Language Processing in Biomedicine and its applications*,
            pages 43–49, 2004.

[FK79]      W. Nelson Francis and Henry Kucera. *Manual of information to accompany A
            Standard Corpus of Present-Day.* Department of Linguistics, Brown University,
            1979. http://khnt.hit.uib.no/icame/manuals/brown/INDEX.HTM.

[GEN]       GENIA. http://www-tsujii.is.s.u-tokyo.ac.jp/jw-tmnlpo/Kim.pdf.

[GJ02]      Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Compu-
            tational Linguistics*, 28:245–288, 2002.

[Hig85]     James Higginbotham. On semantics. *Linguistic Inquiry*, 16:547–593, 1985.

[JM00]      Daniel Jurafsky and James H. Martin. *Speech and Language Processing.* Prentice-
            Hall, 2000.

[Kra89]     Angelika Kratzer. Stage-level and individual-level predicates. *NSF Report*, 1989.

[Mar93]     Mitchell P. Marcus. Building a large annotated corpus of english: The penn treebank.
            *Computational Linguistics*, 19(2):313–330, 1993.

[NH04]      Srini Narayanan and Sanda Harabagiu. Question answering based on semantic struc-
            tures. In *International Conference on Computational Linguistics (COLING 2004)*,
            2004.

[Pus91]     James Pustejovsky. The syntax of event structure. *Cognition*, 41:47–81, 1991.

[Ros99]     Sara Thomas Rosen. The syntactic representation of linguistic events. *GLOT In-
            ternational*, 4:3–11, 1999.

[RR98]      Elizabeth Ritter and Sara Thomas Rosen. Delimiting events in syntax. In W. Geuder
            and M. Butts, editors, *The projection of arguments: Lexical and Syntactic Con-
            straints.* Standford: Center for the Study of Language and Information, 1998.

[RR00]      Elizabeth Ritter and Sara Thomas Rosen. Event structure and ergativity. In
            J. Pustejovsky and C. Tenny, editors, *Events as grammatical objects*, pages 187–
            238. Standford: Center for the Study of Language and Information, 2000.

[SC]        Satoshi Sekine and Michael John Collins. Evalb - bracket scoring program.
            http://nlp.cs.nyu.edu/evalb/.

[SHWA03]    Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. Using
            predicate-argument structures for information extraction. In *Proceedings of hte 41st
            annual meeting of the association for computational linguistics*, pages 8–15, 2003.

[SSA]       Rune Sætre, Harald Søvik, and Tore Amble. Genetuc: Event extraction from relation
            logic. to be published in Computational Linguistics: Special Issue on Semantic Role
            Labeling.

[SSAT05]    Rune Sætre, Harald Søvik, Tore Amble, and Yoshimasa Tsuruoka. Genetuc, genia
            and google: Natural language understanding in molecular biology literature. *Special
            Issue of LNCS Transactions on Computational Systems Biology*, 2005.

[Ste99]      Mark Stevenson. A corpus-based approach to deriving lexical mappings. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 285–286, 1999.

[Sæt02a]     Rune Sætre. *GeneTUC v2 - A Biolinguistic Project, Next Generation*. NTNU, 2002.

[Sæt02b]     Rune Sætre. *Natural Language Understanding - Automatic Information Extraction (IE) from Biomedical Texts*. NTNU, 2002.

[Sæt06]      Rune Sætre. *GeneTUC: Natural Language Understanding in Medical Text*. PhD thesis, NTNU, 2006. http://www.idi.ntnu.no/ satre/genetuc/genetuc.pdf.

[Søv05]      Harald Søvik. Genetuc2: Transformation of internal parse structures to penn treebank-compatible format. Master's thesis, Norwegian University of Technology and Science, 2005.

[Tat]        Yuka Tateisi. Genia project home page. http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/.

[TOiT04]     Yuka Tateisi, Tomoko Ohta, and Jun ichi Tsujii. Annotation of predicate-argument structure on molecular biology text. In *Proceedings of IJCNLP-04 Workshop*, 2004.

[TOTT99]     Yuka Tateisi, Tomoko Ohta, Takako Takai, and Jon'ichi Tsujuu. An ontology for biological reaction events. *Genome Informatics*, 10:298–299, 1999.

[Tra92]      Lisa Travis. Inner aspect and the structure of vp. *Cahiers Linguistique de l'UQAM*, 1:130–146, 1992.

[Tra94]      Lisa Travis. Event phrase and a theory of functional categories. In P. Koskinen, editor, *Proceedings of the 1994 Annual Conference of the Canadian Linguistic Association*, Toronto, 1994. Toronto Working Papers in Linguistics.

[Tra96]      Lisa Travis. The syntax of achievements. In Katherine Crosswhite, editor, *AFLA III*. UCLA, 1996.

[Tra00]      Lisa Travis. Event structure in syntax. *Stanford: Center for the Study of Language and Information*, pages 145–186, 2000.

[tre]        treebank@unagi.cis.upenn.edu. The penn treebank project. http://www.cis.upenn.edu/ treebank/.

[WCS94]      Jong-Nae Wang, Jing-Shin Chang, and Keh-Yih Su. An automated treebank conversion algorithm for corpus sharing. *ACL*, pages 248–254, 1994.

[WEB]        Wikipedia, the free encyclopedia. http://www.wikipedia.org/.

[WSC04]      Tuangthong Wattarujeekrit, Parantu K. Shah, and Nigel Collier. Predicate-argument structures for event extraction in molecular biology. *BMC Bioinformatics*, 5, 2004.

[XP04]       Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, 2004. In conjunction with ACL'04, Barcelona.

[Yak]        Akane Yakushiji. Automatic construction of biomedical information extraction rules as predicate-argument structure patterns. GENIA project.

[YTM01]      Akane Yakushiji, Yuka Tateisi, and Yusuke Miyao. Event extraction from biomedical papers using a full parser. In *In Proceedings of Pacific Symposium on Biocomputing*, pages 408–419, 2001.

# Appendix A

# Report appendix

## A.1 Glossary

**Biolinguistics** - A term coined by Tore Amble to refer to the intersection between linguistics, computer science and biology.

**Event** - A semantic or syntactic-semantic representation of the meaning content of a sentence.

**Event logic** - see Relation logic

**Semantic representation** - may refer to any data structure that tries to capture lingual *meaning* into a formal expression. The capture may be complete (i.e. logic) or domain specific and partial (i.e. Gene Ontology).

**PA-structures** - Predicate Argument-structures (sometimes abbreviated PAS or PASs) is an organised list of syntactic parameters to e.g. a head verb.

**Relation logic** - A descriptive name for the logic produced by GeneTUC (TQL).

**SRL** - Semantic Role Labeling, the concept of analysing syntax with semantic understanding.

**TUC** - The Understanding Computer

**TQL** - TUC Query Language

# A.2 Example code

This section contains code examples from the prototypes that have been constructed. Each example is a "snapshot" from the development. The code is not commented or documented. A version of the final software with appropriate comments, javadoc and documentations can be found as a digital attachment.

## A.2.1 Prolog

```
%% FILE semev.pl
%% CREATED HS-060216

%% events can be of these pattern
% something induces something
% enhancement of something by something
% something binds to something
% something did not affect (event)
% something behaved as an enhancer

%%%
% In case the sentence did not parse, we should stop all execution of this file.
%%
semev(_,error).

%%%
% Main procedure of this file
%%
semev(L,TQL) :-
   value(trace,O),      % user defineable:
   O >= 2,              % minimum trace level for which SES is printed
   write('\n% Semantic Evaluation Structure \n'),
   L = _,
   % Termlist =_,
   % sentence(L,Termlist),
   extract(TQL,TQL,Events),
   write(Events),
   nl.

%% Control structure for traversal of TQL

% H|T - regular list recursion
% E - the sk(Number) of an event
% Event the event()-tuple to be returned

extract([List], [TQL], Event) :-
   extract(List, TQL, Event).

extract(List, TQL, Event) :-
   events(List, TQL, Event).
```

```prolog
events([H|T], TQL, Events) :-
   event(H,TQL,EventHead),
   events(T,TQL,EventList),
   eventlist(EventHead,EventList,Events).

events([], _TQL, []).

eventlist(EventHead,EventList,Events) :-
   var(EventHead) -> Events = EventList ; Events = [EventHead|EventList].

%% end control structure



%% recognition logic

event(Verb/Cause/Theme/sk(E),TQL,event(VerbTranslated,TrueTheme,TrueCause)) :-
   atom(Verb),
   test(v_templ(Verb)),
   findevent1(E,TQL),
   translverb(Verb,VerbTranslated),
   resolveskolem1(TQL,Cause,TrueCause),
   resolveskolem1(TQL,Theme,TrueTheme).


findevent1(E,[H|T]) :-
   findevent2(E,H) ;
   findevent1(E,T).

findevent2(E,event/real/sk(E)).

event(sk(E)isa Buzzword,TQL,event(Buzzclass,Rel1,Rel2)) :-
   buzz(Buzzword,Buzzclass),
   nrel(TQL,Buzzword,E,Relitem),
   resolveskolem1(TQL,Relitem,Rel1),
   nrel(TQL,Rel1,_,Rel2).

event(Process isa process,TQL,event(Process,SrelSubject)) :-
   findsrel1(TQL,SrelSubject) ;
   SrelSubject = nosrel.

findsrel1([H|T],SrelSubject) :-
   findsrel2(H,SrelSubject) ;
   findsrel1(T,SrelSubject).

findsrel2(srel/of/thing/SrelSubject/sk(_E),SrelSubject).

event(Verb/Subj/sk(E),TQL,event(Verb,Subj,Obj)) :-
   atom(Verb),
   test(v_templ(Verb)),
   findsrel3(TQL,E,Obj).
```

```
findsrel3([H|T],E,Obj) :-
   findsrel4(H,E,Obj) ;
   findsrel3(T,E,Obj).

findsrel4(srel/to/thing/Obj/sk(E),E,Obj).

event(_,_,_).  %% ignore lines unrecognizeable

%% support logic

nrel([H|T],Buzzword,E,Relitem) :-
   nrelhead(H,Buzzword,E,Relitem);
   nrel(T,Buzzword,E,Relitem).

nrelhead(nrel/_Of/Buzzword/thing/sk(E)/Relitem,Buzzword,E,Relitem).

% extra layer for constraints on buzzwords
buzz(Buzzword,Buzzclass) :-
   buzzclass(Buzzword),
   buzzlist(Buzzword,Buzzclass).

buzzlist(enhancement,positive_regulation).
buzzlist(induction,positive_regulation).
buzzlist(Buzzword,Buzzword).

buzzclass(Buzzword) :-
   subclass0(Buzzword,abstract);
   subclass0(Buzzword,activity);
   subclass0(Buzzword,process).

translverb(induce, positive_regulation).
translverb(Verb, Verb).

resolveskolem1([H|T],Skolemn,Class) :-
   resolveskolem2(H,Skolemn,Class);
   resolveskolem1(T,Skolemn,Class).

resolveskolem1([],Skolemn,Skolemn).

resolveskolem2(sk(Sk)isa Class,sk(Sk),Class).
```

### A.2.2   Perl

```perl
#!/usr/bin/perl

# Harald Søvik, harals@stud.ntnu.no
# CREATED  060207 10:55.
# MODIFIED 060213 14:45.


#
# Purpose: Extract and create events from TUC dump
#

my $disable_substitutions = 0;

my %substitutions = (
    "induce" => "positive_regulation",
    "expression" => "gene_expression"
);

my $state = 0;
my @block;
my @skolem;
my $eventcounter = 1;

while (<>) {

    if ($_ =~ /^\.{72}$/){
        # we have reached the end of some output.

        if($state == 1){
            # the delimiter belonged to a tql-section - start processing
            processBlock();
            processTQL();

            # unset block to prepare a new tqlblock
            @block = ();
        }

        # indicate that we are outside a section
        $state = 0;

    } elsif (($_ =~ /^% TQL: *$/) or ($state == 1)){
        # we are entering or are already inside a tql-block
        $state = 1;

        if($_ =~ /^% TQL: *$/){
            # ignore TQL start marker
        } elsif ($_ =~ /^\s*$/){
            # ignore lines solemnly consisting of whitespace
        } else {
            # add line to block
            @block = (@block, $_);
```

```perl
        }

    } elsif ($_ =~ /E: (.*)$/){
        print "--- \nE: $1\n";

        if($1 =~ /^\\.title.*$/){
            # if start of abstract, unset skolem array
            @skolem = ();
        }
    } else {

        # print "tql line not recognized\n";

    }
}


###
# this function should determine what the event evaluates to, and normalize the event code
##
sub processBlock(){
    foreach $line (@block){
        if($line =~ /\((.+)\)=>false/){
            @block = ();
            @block = split(/,/, $1);
            @block = (@block, "false");
        }
    }



}


###
# when called, the block should consist of one TQL-statement per line
##
sub processTQL() {
    foreach $line (@block) {
        # first, extract all skolem
        if($line =~ /^sk\((\d+)\).?isa.?(\w+)$/){
            print "skolem[$1] = $2\n";
            $skolem[$1] = $2;


        }
    }

    # extract events
    foreach $line (@block){
        # verb/skolem/skolem/skolem
        if($line =~ /^(\w+)\/sk\((\d+)\)\/sk\((\d+)\)\/sk\((\d+)\)$/){
            print "  \n-- event: $eventcounter
                    \n type = ".sW($1)."
                    \n theme = $skolem[$2]
                    \n cause = $skolem[$3]
```

```perl
                \n";
      $eventcounter++;
  # verb/w/w/sk
  } elsif($line =~ /^(\w+)\/(\w+)\/(\w+)\/sk\((\d+)\)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($1)."
                \n theme = $3
                \n cause = $2
                \n";
      $eventcounter++;
  # nrel/of/w/w/sk/w
  } elsif($line =~ /^nrel\/of\/(\w*?)\/(\w*?)\/sk\((\d+)\)\/(\w*?)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($skolem[$3])."
                \n theme = $4
                \n";
      $eventcounter++;
  # nrel/of/w/w/w/w
  } elsif($line =~ /^nrel\/of\/(\w*?)\/(\w*?)\/(\w+)\/(\w*?)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($3)."
                \n theme = $4
                \n";
      $eventcounter++;
  # nref/for/w/thing/w/w
  } elsif($line =~ /^nrel\/for\/(\w*?)\/(\w*?)\/(\w+)\/(\w*?)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($3)."
                \n theme = $4
                \n cause = $1
                \n";
      $eventcounter++;
  # w/w/sk/sk
  } elsif($line =~ /^(\w*?)\/(\w*?)\/sk\((\d*?)\)\/sk\((\d*?)\)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($1)."
                \n theme = $skolem[$3]
                \n cause = $2
                \n";
      $eventcounter++;
  # w/sk/sk
  } elsif($line =~ /^(\w*?)\/sk\((\d*?)\)\/sk\((\d*?)\)$/){
     print "  \n-- event: $eventcounter
                \n type = ".sW($1)."
                \n theme = $skolem[$2] and $skolem[$3]
                \n";
      $eventcounter++;
  } elsif($line =~ /^false$/){
     print "  \n-- event: $eventcounter
                \n type = negation
                \n";
      $eventcounter++;
```

87

```
        } else {
            # line contains what ?
            # print "unknown\n";
        }
    }
}


sub sW {
 my ($word) = @_;
    if($disable_substitutions){
        return $word;
    } elsif($retSub = $substitutions{$word}){
        return $retSub;
    } else {
        return $word;
    }
}

print "done\n";
```

### A.2.3  Java

```java
import java.util.ArrayList;
import java.util.regex.*;
import java.util.HashMap;



public class TQL {

    public OntologyParser op;
    public TQLAbstract parent_abstract;
    public ArrayList<String> statements = new ArrayList<String>();
    public ArrayList<Event> events = new ArrayList<Event>();
    public HashMap<String, String> nouns = new HashMap<String, String>();
    public int input;
    public int block;
    public int eventcounter = 0;

    public final boolean debug_tql = false;




    public TQL(int input, int block, TQLAbstract par, OntologyParser op){

        this.input = input;
        this.block = block;
        this.parent_abstract = par;
        this.op = op;
    }


    public void addLine(String line){
        statements.add(line);
    }

    public ArrayList getLines(){
        return statements;
    }

    public void hello(){
        printEvents();
    }

    public ArrayList getEvents(){
        return events;
    }

    public void printEvents(){
        for(Event i : events){
```

```java
            i.printEvent();
        }
    }


    public synchronized int getNextEvID(){
            eventcounter++;
            return eventcounter;
    }


    public void extractEvents(){

        // PUT INITIATING RULE CALLS HERE

        findNoun(); // rule 1
        findVerb(); // rule 2

    }


    public void findNoun(){ // RULE 1: if noun isa activity -> activity isa event
        for(String s : statements){
            identifyNoun(s);
        }
    }


    public void findVerb(){ // induce/lipopolysaccharide/phosphorylation/sk(1)
        for(String s : statements){
            identifyTransVerb(s);
        }

        for(String s : statements){
            identifyITVerb(s);
        }
    }

    public void identifyNoun(String s){
        Pattern p_1 = Pattern.compile("^([\\w()]+)\\s?isa\\s(\\w*)$");
        Matcher m = p_1.matcher(s);

        while ( m.find() ) {
            String g1 = m.group(1).trim();
            String g2 = m.group(2).trim();

            nouns.put(g1, g2);

            if( op.isBuzzNoun(g2) /* || op.isOfClass(g2,"Biological_process") */ ) {
                // identified a process etc - create event and find theme denoted by induce/lipopol
                if ( debug_tql ) System.out.println("MATCH NOUN: "+g2+" in "+s+" with regexp "+p_1.

                Event e = new Event(this);
                e.setType(sk(g1));
                e.setLitType(g1);
                events.add(e);
```

```
                parent_abstract.addEvent(e);
                findNREL(e);
                identifyAdj(e);
            }
        }
    }


    public void identifyTransVerb(String s){
        Pattern p_4 = Pattern.compile("^(\\w+)/([\\w()]+)/([\\w()]+)/sk\\(\\d+\\)$");
        Matcher m = p_4.matcher(s);


        while ( m.find() ) {


            String g1 = m.group(1).trim();
            String g2 = m.group(2).trim();
            String g3 = m.group(3).trim();

            if( op.isInOntology( g1 ) ) {

                if ( debug_tql ) System.out.println("MATCH TR-VERB: "+s+" with regexp "+p_4.

                Event e = new Event(this, g1);
                e.setType(g1);
                e.setCause(sk(g2));
                e.setLitCause(g2);
                e.setTheme(sk(g3));
                e.setLitTheme(g3);
                events.add(e);
                parent_abstract.addEvent(e);
            }
        }
    }


    public void identifyITVerb(String s){

        Pattern p_5 = Pattern.compile("^(\\w+)/([\\w()]+)/sk\\(\\d+\\)$");
        Matcher m = p_5.matcher(s);

        while ( m.find() ) {

            String g1 = m.group(1).trim();
            String g2 = m.group(2).trim();

            if( op.isBuzzVerb( g1 ) ) {

                if ( debug_tql ) System.out.println("MATCH IT-VERB: "+s+" with regexp "+p_5.

                Event e = new Event(this, g1);
```

```java
                e.setTheme(sk(g2));
                e.setLitTheme(g2);
                events.add(e);
                parent_abstract.addEvent(e);
            }
        }
    }


    // NLP-feature
    public String prolog(String s){
        if( s == null ) {
            return "(\\w+)";
        } else {
            return s;
        }
    }


    public String sk(String s){
            Pattern sk = Pattern.compile("^sk\\(\\(\\d+\\)\\)$");
            Matcher msk = sk.matcher(s);
            if ( msk.find() ) {
                String nounclass = nouns.get( msk.group() );
                return nounclass;
            } else {
                return s;
            }
    }



    public void findNREL(Event e){
        identifyNREL(e);
    }

    public void identifyNREL(Event e){
        // dersom referanse er en skolem må vi escape parenteser
        String ref = e.getLitType();
        Pattern p_2 = Pattern.compile("sk\\((\\d+)\\)");
        Matcher l = p_2.matcher(ref);
        if ( l.find() ) {
            ref = "sk\\("+l.group(1)+"\\)";
        }

        Pattern p_3 = Pattern.compile("^nrel/of/(\\w+)/(\\w+)/"+ref+"/([\\w()]+)$");

        for(String s : statements) {
            Matcher m = p_3.matcher(s);
            if ( m.find() ) {
                if ( debug_tql ) System.out.println("MATCH NREL: "+s+" with regexp "+p_3.pattern())
                e.setTheme(sk(m.group(3).trim()));
                e.setLitTheme(m.group(3).trim());
            }
        }
```

```java
        Pattern p_4 = Pattern.compile("^nrel/by/(\\w+)/(\\w+)/"+ref+"/([\\w()]+)$");

        for(String s : statements) {
           Matcher n = p_4.matcher(s);
           if ( n.find() ) {
              if ( debug_tql ) System.out.println("MATCH NREL: "+s+" with regexp "+p_4.pat
              e.setCause(sk(n.group(3).trim()));
              e.setLitCause(n.group(3).trim());
           }
        }


        Pattern p_5 = Pattern.compile("^nrel/for/(\\w+)/(\\w+)/([\\w()]+)/"+ref+"$");

        for(String s : statements) {
           Matcher n = p_5.matcher(s);
           if ( n.find() ) {
              if ( debug_tql ) System.out.println("MATCH NREL: "+s+" with regexp "+p_5.pat
              e.setCause(sk(n.group(3).trim()));
              e.setLitCause(n.group(3).trim());
           }
        }


        Pattern p_6 = Pattern.compile("^nrel/to/(\\w+)/(\\w+)/"+ref+"/([\\w()]+)$");

        for(String s : statements) {
           Matcher n = p_6.matcher(s);
           if ( n.find() ) {
              if ( debug_tql ) System.out.println("MATCH NREL: "+s+" with regexp "+p_6.pat
              e.setCause(sk(n.group(3).trim()));
              e.setLitCause(n.group(3).trim());
           }
        }

     }

     public void identifyAdj(Event e){

        String skolem = escapeRegEx(e.getLitType());

        Pattern p_7 = Pattern.compile("^adj/(\\w+)/"+skolem+"/real$");

        for(String s : statements) {
           Matcher n = p_7.matcher(s);
           while ( n.find() ) {
              String adj = n.group(1).trim();
              if ( op.isBuzzAdjective(adj) ) {
                 if ( debug_tql ) System.out.println("MATCH ADJ: "+s+" with regexp "+p_7.p
                 e.type_extended = adj.concat(" "+e.type);
              }
           }
        }
```

```
    }


    public String escapeRegEx(String s){

        String result = "";

        for ( int i = 0 ; i < s.length() ; i++ ) {
            String c = s.substring(i,i+1);
            if ( c.equals("(") ) {
                result = result.concat("\\(");
            } else if ( c.equals(")") ) {
                result = result.concat("\\)");
            } else {
                result = result.concat(c);
            }
        }
        return result;
    }

}
```

## A.3 Possible improvementes to GENIA events

Some of the problems encountered while working with the event-annotated corpus are rooted in the annotation itself. They are mentioned here to aid GENIA in their effort.

### A.3.1 References to events

Distinguishing events from nouns is not straightforward.

When an event is connected to another event through theme or cause, it is represented exactly the same way as a noun would have been represented. This will cause problems for all nouns that match the pattern E
d+, i.e. E16, which in some cases may be a reference to the bacteriophage Lactococcus phage bIL170. In addition, it is neccessary to parse the idref-value to determine where to look up the reference. Example in Table A.1.

```
Noun:
<theme idref=''T13''/>


Event:
<theme idref=''E13''/>
```

**Table A.1:** Ambigious references

### A.3.2 Duplicate syntax in ontology

To specify that a class is a subclass of something, both of the syntaxes shown in Table A.2 is used. This can be sort of confusing, and is not necessary unless there exist an explicit reason to do so.

```
<owl:Class rdf:ID=''...''>
 <rdfs:subClassOf rdf:resource=''Regulation''/>
</owl:Class>

<rdfs:subClassOf>
 <owl:Class rdf:ID=''Regulation''/>
</rdfs:subClassOf>
```

**Table A.2:** Redundant syntax

### A.3.3 Multiple themes

These are represented as theme-tags at the same level, and causes a need for retaining the previously parsed tag in the xml-handler. It could be useful to contain these in a special <multitheme>-tag. See Table 5.5

# Appendix B

# Papers

This section contains articles that has been co-authored by the author of this report.

*GeneTUC, GENIA and Google: Natural Language Understanding in Molecular Biology Literature* was written during autumn 2005, and was the result of a project that can be considered preparatory to this thesis. It has been publised in Special Issue of LNCS Transactions on Computational Systems Biology[SSAT05].

*GeneTUC and Event Extraction in biomedical texts* is a derivative work of this thesis. The version included here is only a draft version.

# 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%.

**Keywords:** Biomedical Literature Data Mining, Google API, GENIA

## 1 Introduction

In recent years, the interest in developing effective tools for Natural Language Processing (NLP) tasks in biomedical literature has been increasing. There is a practical need to effectively curate, organize and retrieve information automatically from textual sources, and most of these sources have already been indexed by the worlds largest search engines, such as Google and Yahoo. With the recent release of Application Programming Interfaces (APIs) to these search enginges, a world of new possibilites for practical applications has presented itself.

### 1.1 GeneTUC, GProt and Bioogle

This paper describes how such new applications, like Bioogle [11] and GProt [10], that is built on top of the Google API, can be used to enhance already existing applications, like GeneTUC [8] which is a Natural Language Understanding

(NLU) system. GProt[3] provides a way of automatically updating protein and gene databases by extracting information from the (biomedical research) literature. The general idea behind this type of approaches is described in [16, 15, 13]. Most of this biomedical literature is already indexed in MEDLINE, and therefore also by Google and other major search engines. Databases that can be enriched with this kind of automatically extracted information are "Entrez Gene", Uni-Gene, Swiss-Prot and other protein DBs, pluss Gene Ontology which contains semantic labels. Bioogle[4] is a simpler system that just uses Google to determine the semantic class of a word, like "CCK is a protein", so that it can be added to the semantic hierarchy (or dictionary) in a correct way. See [7] for more details and a description of how to access the online versions of these programs.

## 1.2 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 [3], and a more recent "IE in Biology" survey is given in [9]. 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 [4], *manually* developed IE rules [17], protein name classifiers trained on *manually* annotated training corpora [1], and classifiers trained on *automatically* annotated training corpora [14].

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 [5]. 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 corpora is used, namely the GENIA Tree Bank (GTB) corpus [12], first to train and then later test how well the GeneTUC parser performes compared to other available parsers in this domain.

## 1.3 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 continuously being development at the Norwegian University of Science and Technology. Section 3 will explain in more detail how TUC, and especially GeneTUC, works.

---

[3] http://furu.idi.ntnu.no:9080/gprot/
[4] http://furu.idi.ntnu.no:9080/bioogle/

## 1.4 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 [12]. 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. The Subsection 2.2 has more details of this

## 2.1 MEDLINE

Medline[5] 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 licences for PubMed/MEDLINE for free to anyone interested. When it was downloaded in September 2004, the academic package includeded 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[6] 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 tokenisation, which is handled by another module, namely the lexical analysis module, in GeneTUC.

---

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

## 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 (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 excisting KB.
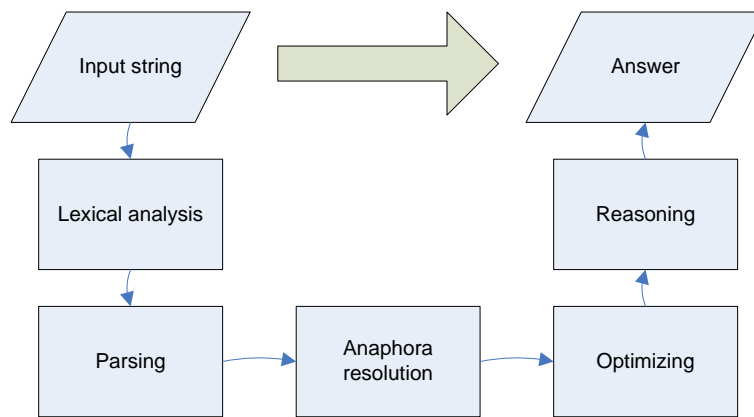


**Fig. 1.** Data flow in the TUC System

*Example 1.*
$$\frac{\begin{array}{c} \text{Statement : "CCK activates Gastrin."} \\ \text{Update to KB : activate(cck,gastrin).} \end{array}}{\begin{array}{c} \text{Question : "What activates Gastrin?"} \\ \text{Answer : "CCK"} \end{array}}$$

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 the 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 [2] 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 Questionm, 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 |
| . . . |

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 same grammar as in Example 3. This rule is context sensitive, meaning that *ing*-verbs like "resulting" can only be substituted by 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 successfull 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 phenomena, which is hard to parse even for humans, is *garden path sentences* [6].

# 4  Methods

The main goal of this research was to evaluate the GeneTUC parser on the GENIA corpus. Since GeneTUC and GENIA was not made using any common grammar standard, a lot of modifications in GeneTUC were needed. These adaptions 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 tokenisation as in the GENIA gold file. This was only successfull in around 20% of the sentences, so we reduced the test set to only include sentence that were similarily 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[7] relations (class concepts), and *not* as isa[8] relations (proper names).

---

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

3. *Duplicate Entries.* Changing plural forms into singular forms often leads to duplicate entries in the dictionary, since the singular form
4. *Short Ambigious 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 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 i 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 Gene-TUC 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
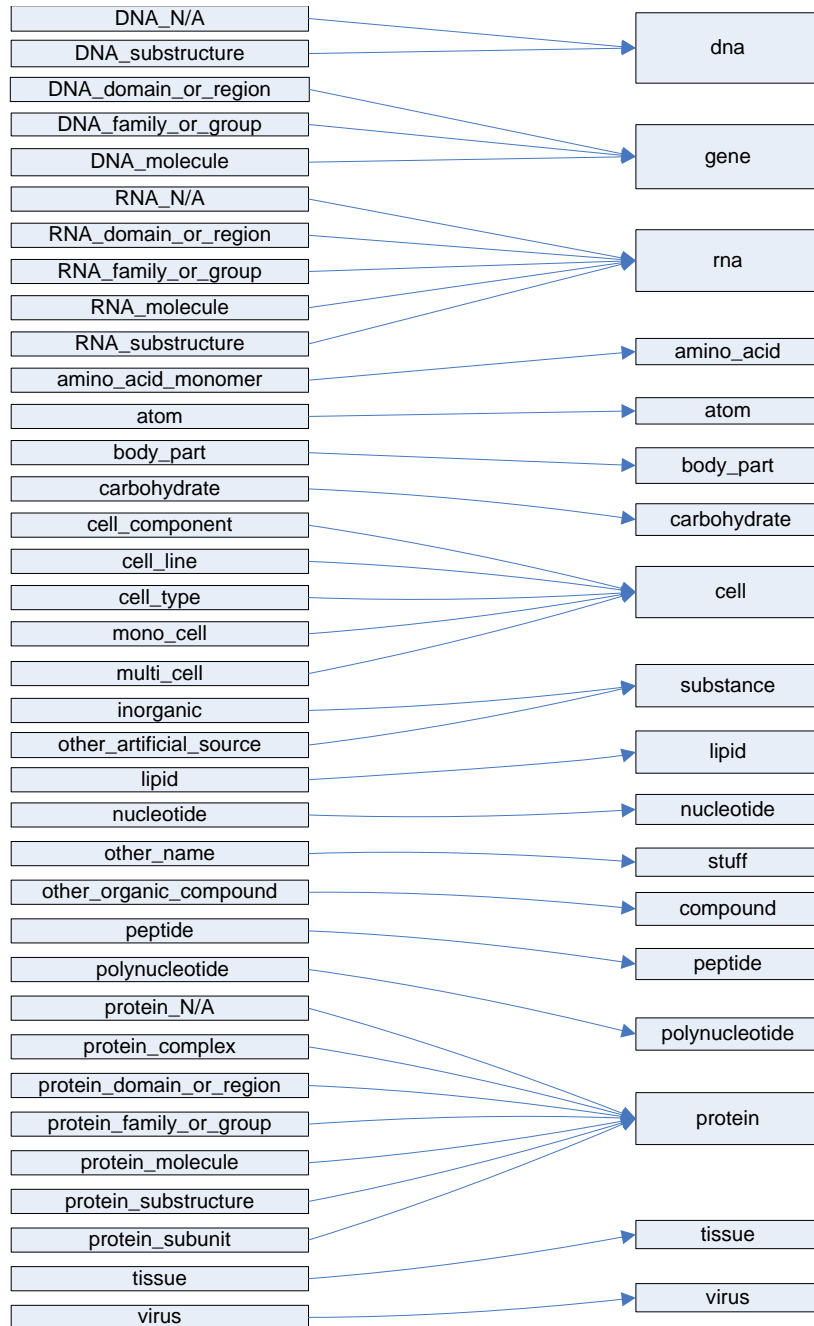
**Fig. 2.** Conversion from GENIA to GeneTUC Ontology

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 parsetrees, 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.

The creation of the grammar is currently done 100% manually. It is a slow and longlasting 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 Tokenisation

EvalB[9] 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 characters listed in Example 7.

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

Also, single tokens (like IL-2) are sometimes turned into two seperate 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 tokenised and stored in XML format, the correct tokenisation 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 tokenised version of GTB, instead of the plaintext version. The problem then, is that we have to circumvent TUC's own tokenisation module (lexical analysis), and this might introduce problems in the later moduls, for example because of ignored characters that were previously handled by the lexical module. Another example of problems introduced if the original tokenisation 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 tokenisation problem is to make a new plaintext version of the text, from the tokenised xml-version. In the new plain text version,

---

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

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 tokenisation different from the "golden" tokenisation renders the parse incomparable; Those sentences where the number of golden tokens and test tokens are inequal leads to an *error*. Likewise do those where the golden token and test token are characterwise 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* are those brackets covering the correct tokens, and has correct *label*. The matches enables measurement of:

- Recall (the ratio between matched brackets and total brackets in gold file)
- Precision (the ratio between matching bracketing and number of brackets in test file)
- Crossing (the ratio between those brackets in the test file exceeding the span of the matching bracket in the gold file, and the total number of brackets in test file)
- Tagging accuracy (the ratio of correct labelled tokens to 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 gold file.

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

| Meassurement | Dictionary | |
|---|---|---|
| Description | **Original** | **+GENIA** |
| Number of sentences: | 2591 | 2591 |
| Successful parses: | 318 | 1531 |
| Successrate: | 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 the EvalB

| Meassurement | 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 Related Work

# 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 anaphorical 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 tokenisation and optimisation of the text before passing it on to the parser. We could perhaps have used the already tokenised 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 seperate 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.

Tokenisation is usually done before, and seperate 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 this is an old way of thinking, and 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.

# Appendix A

## Acknowledgements

# 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 (Forthcoming)*, 2004.
2. Michael A. Covington. *Natural Language Processing for Prolog Programmers*. Prentice-Hall, Englewood Cliffs, New Jersey, 1994.
3. J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, January 1996.
4. 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.
5. 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.
6. 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.
7. Tuan D. Pham, Hong Yan, and Denis I. Crane. *Advanced Computational Methods for Biocomputing and Bioimaging*, chapter WebProt: Online Mining and Annotation of Biomedical Literature using Google. Nova Science Publishers, New York, USA, 2006.
8. Rune Sætre. GeneTUC, A Biolinguistic Project. (Master Project) Norwegian University of Science and Technology, Norway, June 2002.
9. Rune Sætre. Natural Language Processing of Gene Information. Master's thesis, Norwegian University of Science and Technology, Norway and CIS/LMU Munchen, Germany, April 2003.
10. 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.
11. Rune Sætre, Amund Tveit, Tonje Strœmmen Steigedal, and Astrid Lægreid. Semantic Annotation of Biomedical Literature using Google. In Dr. Marina

Gavrilova, Dr. Youngsong Mun, Dr. David Taniar, Dr. Osvaldo Gervasi, Dr. Kenneth Tan, and Dr. Vipin Kumar, editors, *Proceedings of the International Workshop on Data Mining and Bioinformatics (DMBIO2005)*, Lecture Notes in Computer Science (LNCS) (Forthcoming), Singapore, May 2005. Springer-Verlag Heidelberg.

12. Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. Syntax annotation for the genia corpus. In *Proceedings of the IJCNLP 2005*, Korea, October 2005.

13. Jun'ichi Tsujii and Limsoon Wong. Natural Language Processing and Information Extraction in Biology. In *Proceedings of the Pacific Symposium on Biocomputing 2001*, pages 372–373, 2001.

14. Amund Tveit, Rune Sætre, Tonje S. Steigedal, and Astrid Lægreid. ProtChew: Automatic Extraction of Protein Names from . In *Proceedings of the International Workshop on Biomedical Data Engineering (BMDE 2005, in conjunction with ICDE 2005)*, Tokyo, Japan, April 2005. IEEE Press (Forthcoming).

15. Limsoon Wong. A Protein Interaction Extraction System. In *Proceedings of the Pacific Symposium on Biocomputing 2001*, pages 520–530, 2001.

16. Limsoon Wong. Gaps in Text-based Knowledge Discovery for Biology. *Drug Discovery Today*, 7(17):897–898, September 2002.

17. 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.

# Computational Linguistics - Special Issue on Semantic Role Labeling

# GeneTUC: Role Labeling of Protein-Protein Interaction using Semantic Events

Rune Sætre*
Department of Computer and
Information Science, Norwegian
University of Science and Technology

Harald Søvik**
Department of Computer and
Information Science, Norwegian
University of Science and Technology

Tore Amble†
Department of Computer and
Information Science, Norwegian
University of Science and Technology

*With increasing amounts of biomedical literature, there is also an increasing need for information extraction systems to support biomedical researchers. GeneTUC has been developed to be able to read biological text and answer simple questions about it 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. In this paper, we present an evaluation of GeneTUC, using a small part of the GENIA corpus that is already manually annotated with semantic events.*

## 1. Introduction

In recent years, the interest in developing effective tools for natural language processing (NLP) tasks in biomedical literature has been increasing. There is a practical need to effectively curate, organize and retrieve information automatically from textual sources, and most of these sources have already been indexed by the worlds largest search engines, such as Google and Yahoo. With the recent release of Application Programming Interfaces (APIs) to these search enginges, a world of new possibilites for practical applications has presented itself.

---

* satre@idi.ntnu.no
** harals@idi.ntnu.no
† toreamb@idi.ntnu.no

### 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 (**?**), and a more recent "IE in Biology" survey is given in (**?**). 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 (**?**), *manually* developed IE rules (**?**), protein name classifiers trained on *manually* annotated training corpora (**?**), and classifiers trained on *automatically* annotated training corpora (**?**).

A new emerging approach to medical IE is the heavy use of corpora. The workload can then be shifted from the extremely timecomsuming manual grammar construction to the somewhat easier and more teamwork oriented corpus/treebank building (**?**). 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 corpora is used, namely the GENIA event annotated corpus (), first to train and then later test how well GeneTUC is able to abstract the different ways to express protein-protein interactions.

### 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 continuously being 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 paper is organized as follows. Section **??** describes the materials and programs that were used, section 3 explains in detail how GeneTUC works, section **??** presents our approach, section **??** presents the empirical results, section **??** describes other related work, section **??** contains a discussion of the results, and finally the conclusion and future work are presented in section **??**.

### 2. Material

The main goal is to extract events from the text. Events in this context relate to some form of interaction between proteins and/or gene products. To construct such a system, we needed a text that already had been annotated with events. Such text is tentatively being provided by GENIA: Event annotated abstracts from MEDLINE. In total, GENIA were able to provide 19 annotated abstracts, from which we randomly selected 16 (8 for testing and 8 for training).

| Abstracts (MEDLINE ID) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Train: | 1419905 | 1431113 | 1464736 | 1482376 | 1493333 | 1502202 | 1505523 | 1527846 |
| Test: | 1527859 | 1531086 | 1533884 | 1583734 | 1618911 | 1653056 | 1655897 | 1668145 |

**Table 1**
MEDLINE abstracts

## 2.1 MEDLINE

Medline[1] 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 licences for PubMed/MEDLINE for free to anyone interested. The academic package includeded a local copy of 6.7 million abstracts, out of the 12.6 million entries that were available on their web interface, as of September 2004.

## 2.2 GENIA events

It was decided to use GENIA event annotated corpus for training and testing of Gene-TUC. This corpus has not yet been officially released, but as a courtesy, GENIA provided us with an example corpus consisting of 19 annotated abstracts. Each sentence in these abstracts have been manually inspected and annotated by a biologist. Each biologically interesting term have been highlighted with a markup consisting of identificator, lexical string and semantic class. Each sentence also has an array of events that have been extracted and linked to terms in the sentence. An event is classified by the parameter *type*, (e.g. activation). The parameter *theme* corresponds to the patient of an event (e.g. gastrin), and the optional *cause* refers to the agent (e.g. cck). An event also has a *clue*, which is a text passage that justifies the event. The clue contains certain keywords: *clueType*, which is a word or passage that acts as a trigger to the event, *linkTheme* and *linkCause* is the mentioned patient and agent, *clueLoc* is an environmental location of the event, and *clueTime* is a temporal parameter related to the event.

A subset of the attributes in an event were found to be appropriate for automatic comparison: type, theme and cause.

## 3. GeneTUC

GeneTUC is on the way to be a full-fledged Question Answering system, but the coverage is still low. Figure **??** 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 second order logic form called Tuc Query Language (TQL) and then stored in the Knowledge Base (KB) of the system. If a later input string is a question, it will be coded in the same way using TQL, but either the subject or one of the objects in the sentence will be

---

1 http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

wildcards to be matched against the excisting KB. Example 1 shows an example of how this works.

**Example 1**

| Statement | : | "CCK activates 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 the the question is stated in a similar grammatical form as the factual statement. This means it will be an advantage to have the same fact stored many times in the system, which is usually the case when MEDLINE is used as the source of facts, since authors tend to repeat themselves, and also many authors are likely to write about the same facts.

### 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 │ ; semicolon │ ? question mark │ ! exclamiation 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.

**3.2.1 Categorial Grammar.** TUC is based on a Categorial Grammar which allows *gaps* in the sentence. This mechanism is very easy to use when parsing sentences like in this example.

**Example 2**

Input: What activates Gastrin?

| Grammar: | | |
|---|---|---|
| Statement | → | NounPhrase VerbPhrase |
| Question | → | [What] Statement\NounPhrase |
| VerbPhrase | → | Verb NounPhrase |
| | ... | |

**Example 3**

Input:

A gene signal resulting in protein production occurs.

A gene signal that results in production of proteins occurs.

Grammar:

| | | |
|---|---|---|
| Statement | → | NounPhrase RelativeClause VerbPhrase |
| RelativeClause | → | [that] Statement\NounPhrase |
| | . . . | |

**Example 4**

Input:

Results of preliminary studies, which we have conducted,

suggest that use of this agent is useful.

Grammar:

| | | |
|---|---|---|
| NounPhrase | → | Det Nominal RelativeClause |
| RelClause | → | RelativePron Statement/NounPhrase |
| | . . . | |

Example 2 shows how a *What-Question* can be parsed using the existing grammar rules for *statements*. It states that *What-questions* consist 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 with the actual NP in a corresponding fact statement, like the one given in Example 1, and give a correct answer to the natural language query. The use of Forward (\) and Backward (/) application also reduces the number of grammar rules needed, since every new rule for statements implicitly creates corresponding new rules for questions.

Example 3 shows Forward application in a Relative Clause, where the missing NP in the Relative Clause can be found by inspecting the surrounding NP. In Example 4 the use of Backward application is shown. In GeneTUC, Backward application also includes Inward application, so "S/NP" means that the NP can be missing anywhere in the Statement.

### 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 setnence, this part of the sentence is *locked* 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 differen span of words to produce a different and successfull NP. This kind of backtracking can be very computational expensive, especially with highly ambigious 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 phenomena, which is hard to parse even for humans, is *garden path sentences* (**?**).

ENTITY1 recognizes and **activates** ENTITY2.
ENTITY1 can **activate** ENTITY2 through a region in its carboxy terminus.
ENTITY2 are **activated** by ENTITY1a and ENTITY1b.
ENTITY2 **activated** by ENTITY1 are not well characterized.
The herpes virus encodes a functional ENTITY1 that **activates** human ENTITY2.
ENTITY1 can functionally cooperate to synergistically **activate** ENTITY2. The ENTITY1
play key roles by **activating** ENTITY2.

**Table 2**
Variations of E1 activates E2 in text

```
...
activate/ENTITY1/ENTITY2/sk(1)
...
```

**Table 3**
Variations of E1 activates E2 in TQL

### 3.4 TUC Query Language (TQL)

GeneTUC produces a knowledge representation language called TQL, which is a vari-
ant of relation logic. The transformation into logic removes a lot of complexity from the
language, and thereby eases the burden of finding extraction patterns. The KRL consists
of 8 predicates which is able to represent any expression in natural language.

### 4. Method

A method for extraction of semantic events from GeneTUC is described in detail in
(**?**). The approach for identification of these events was developed specific for evalua-
tion againt GENIA events. The extraction engine presented here is using hand-crafted
extraction rules created from sparse data. This may seem like heading in the opposite
direction of most other efforts exhibited today, but this method shows promising results
none the less.

It is founded on a novel approach for event extraction - event logic generated by
the deep parsing NLU-system GeneTUC. This additional layer of abstraction justifies
the use of manual work and the sparse-data approach. In initial experiments, a recall of
25%, with 15% precision was achieved.

Lingual expressions which use different words and syntactic constructs, may still
have the same intended meaning. This meaning can be represented in relational logic,
and the differences are thus removed to some extent. By observing how typical expres-
sions that contain one of more events are represented in relational logic, it is possible to
construct patterns that capture events.

The fact that these 7 representations is reduced to a single representation in TQL
spurs the idea that the GeneTUC system is able to abstract away these variations and
directly give access to an event structure in logic that can be used for event extraction
purpose.

However, events can be expressed in additional ways that are not as easily identi-
fied::

ENTITY1-activation of ENTITY2.

---

**Table 4**
Variations of E1 activates E2 in text

These structures in logic were manually identified, using the the very sparse data that is shown in (**?**).

### 4.1 Implementation

We have hand-crafted rules in Java that is using regular expressions to define such patterns. By also implementing an array of constraints that has to be satisfied for each capture to complete, we are able to filter the interesting biological events from those of no significance.

Currently the effort has been spent on identifying the events, so those constraints which has to be satisfied to "initate" an event are rather strict. There are three overall event-initiating statements:

1. something isa interesting_process

2. interesting_verb/subject/object/id

3. event/world/id

The constraints applies to *interesting_process* and *interesting_verb*, which must be of some biological character, i.e. *activation* or *bind*. These "clue-words" have been extracted from a small set of event-annotated abstracts by GENIA. The event-statement is a method in GeneTUC of treating modality, but can also be used to track biological events.

When an event has been initiated, a larger set of rules is iterated over each set of TQL-statements, in pursuit of candidates to fill the "theme" and "cause" slots. "Type" is usually implicitly given by the clueword that produced the initial match.

### 4.2 Problems

Granularity of events proved to be our major problem. Whereas the TQL easily allows identification of a single event within a sentence, the same event with other granularities were far more problematic to identify. Set dependencies also presented a problem, due to a complex logic representation. Both of these problems were solved as good as possible by manual exhaustive programming.

Enumeration (and separation) of events in the sentences presented another problem. Whereas some sentences has multiple events, other have none at all, and in extreme cases, events may span several sentences. This problem had to be procured by automatically inserting "dummy"-events after sentences where GeneTUC discovered fewer events than listed in the GENIA corpus.

The comparision of two events were supported by manual intervention at some points. I.e. the GENIA term binding is represented as binding activity in GeneTUC. To ensure correct and fair scoring, such choices decided by the analyst were stored in a database for later usage. In this way, the task of evaluating multiple events were tractable for a single student on a single afternoon.

## 5. Results

This chapter contains a summary of the results from event extraction and event evaluation. These functions were first trained, then tested. Cross validation of the results were unfortunately not possible, since repeated manual construction of extraction rules is infeasible.

### 5.1 Training data

| Processing statistics | | | |
|---|---|---|---|
| | TQL | GE | % |
| Total abstracts in file; | 8 | 8 | (100%) |
| Events in all abstracts; | 62 | 212 | (29%) |
| Sentences in all abstracts; | 35 | 79 | (44%) |
| Precision and recall | | | |
| Event precision: | 10/62 | | (16%) |
| Attribute precision (type): | 25/62 | | (40%) |
| Attribute precision (theme): | 14/62 | | (22%) |
| Attribute precision (cause): | 23/62 | | (37%) |
| Average precision: | 62/3x62 | | (33%) |
| Event recall: | 62/92 | | (67%) |
| F-score for | | | |
| Harmonic F-score (attribute): | | | 0.44220003 |
| B-weighted F-score (attribute): | | | 0.41835700 |
| | | | |
| Harmonic F-score (event): | | | 0.25831324 |
| B-weighted F-score (event): | | | 0.20283974 |

### 5.2 Test data

| Processing statistics | | | |
|---|---|---|---|
| | TQL | GE | % |
| Total abstracts in file; | 8 | 8 | (100%) |
| Events in all abstracts; | 24 | 181 | (12%) |
| Sentences in all abstracts; | 37 | 64 | (57%) |
| Precision and recall | | | |
| Event precision: | 3/22 | | (13%) |
| Attribute precision (type): | 8/22 | | (36%) |
| Attribute precision (theme): | 3/22 | | (13%) |
| Attribute precision (cause): | 7/22 | | (31%) |
| Average precision: | 18/3x22 | | (27%) |
| Event recall: | 22/76 | | (28%) |
| F-score | | | |
| Harmonic F-score (attribute): | | | 0.2749091 |
| B-weighted F-score (attribute): | | | 0.5007418 |
| | | | |
| Harmonic F-score (event): | | | 0.17756097 |
| B-weighted F-score (event): | | | 0.24109791 |

**5.3 Comparison**

| Attribute | Training | Test |
|---|---|---|
| Sentences in all abstracts | 35/79 (44%) | 37/64 (57%) |
| Event precision | 10/62 (16%) | 3/22 (13%) |
| Average attr. precision | 62/186 (33%) | 18/66 (27%) |
| Event recall | 62/92 (67%) | 22/76 (28%) |

**6. Discussion**

**7. Conclusion**

# Appendix C

# Software documentation

This documentation covers standard user opererations for the software package EvEx (Event Extraction), a pattern recognition program for GeneTUC meant to extract semantic events from relation logic. This software is intended for academic usage, and not end-user distribution. The documentation is relatively superficial, and focuses on demonstrating basic functionality in the program.

This documentation is part of a larger package, consisting of

- software documentation

- application programmers interface documentation

- software bytecode

- software source code

- test data

- report (thesis)

## C.1   Achieving the software

All of the items above is included in a ZIP-file, which is accessible from DAIM[1] at http://daim.idi.ntnu.no or http://folk.ntnu.no/∼harals/thesis.zip

The report should also be accessible alone from http://folk.ntnu.no/∼harals/thesis.pdf

## C.2   Requirements

This software could be run on any computer that you get it to run on. However, correctness is guaranteed only if the following requirements are met.

---

[1]Digital Arkivering og Innlevering av Masteroppgaver

## C.2.1   Operating system

Using a 32-bit computer running GNU Linux is recommanded.

64-bit systems could run the program in 32-bit compatible-mode using the -d32 option. Running the system on MS Windows platforms and Solaris should be possible, but have not been tested.

## C.2.2   Hardware requirements

No special hardware specifications is needed to run the program.

## C.2.3   Software requirements

The software is written for Java 1.5.0 (also versioned 5.0), and will not run cleanly on earlier versions. (This is because the code employs generics and syntax that was introduced in the 1.5 release.)

Except for the native classes of Sun Java, an additional package is required: *org.apache.commons.collections.map*. This package has been included in the program directory.

There are also some reccomendations that would ease the user experience. As the software is intended to be used with GeneTUC, it would be practical to have GeneTUC on the same computer. This introduces several other requirements:

- SunOS / Solaris

- SICStus 3.11 or newer

- Perl 5.8

# C.3   Installing the program

The ZIP-file contains a directory, where there are 4 subdirectories. The program requires no further install than unpacking to a convinient location on your hard drive.

- **EvEx**
  Contains the source code and binaries (java byte code).

- **doc**
  Contains the report with appendixes.

- **javadoc**
  Contains the javadoc in HTML-format. Intended for developers only.

- **data**
  Contains output from GeneTUC, extracted TQL from this output, event annotated corpus from GENIA and distilled XML events from GENIA and GeneTUC.

## C.4 Executing the program

### C.4.1 Example run

To quickly get an impression of the program, execute this command in the EvEx directory:

```
java EvEx ../data/genia20/train8.gen ../data/tql20/train8.tql
```

### C.4.2 Required parameters

Correct syntax is defined as:

```
java EvEx
            [--debug {all,tql,ge,ontology}]
            [--manual]
            [--tqlxml file]
            [--genxml file]
            goldfile.xml
            testfile.xml
```

The essential parameters is the two files that contain data for testing: goldfile and testfile.

**(goldfile)**

The goldfile contains a golden corpus that has been annotated with GENIA events, as defined by the GENIA Document Type Definition. It provides a perfect solution for event extraction.

**(testfile)**

The testfile is a file produced by GeneTUC. It contains TQL-code that has been delimited with XML. The sequence of sentences must be the same as in the gold file.

### C.4.3 Optional parameters

There are a small number of additional options that can be specified at the command line.

**–debug {all,tql,ge,ontology}**

Enables program trace from different program modules:

- tql - parsing of testfile and event extraction logic

- ge - parsing of goldfile and creation of GENIA event objects

- ontology - parsing of the ontologyfile

- all - all of the above

**–manual**

This switch enables interaction in the evalutaion. When it is impossible for EvEx to determine if two events is synonymous, the user is queried. Such question could be i.e. if "binding" and "binding activity" should be considered equal. *0* or *n* returns a negative answer, and *1* or *y* returns positive. The default is to return negative.

Both negative and positive answers are stored and used in subsequent evaluations.

**–tqlxml (file)**

Enables output of "distilled" events to (file). These events were those discovered in the TQL code. See C.4.4.

**–genxml (file)**

Enables output of "distilled" events to (file). These events were those parsed from GENIA annotated corpus. See C.4.4.

## C.4.4   EvalE

EvalE is a much simpler program that works by simpler principles. It takes two paths as arguments: distilled gold and distilled test-files. These files has been filled with "dummy" events, so that by the start of each sentence in corpus, the events are "in synch".

The meaning of this program is is enable manual curation of events, to acheive more correct scores. A typical usage is to delete excess events from GeneTUC or GENIA, to compare only those events that are representing the same. This impacts precision/recall score, so numbers from this program should never be compared to other numbers.

Example:

```
java EvalE gen.xml tql.xml
```

## C.4.5   tqlextract.pl

Since GeneTUC produces a dump-file containing a whole lot more than TQL, this small program has been provided to filter the TQL-statements to a single file. In addition, it provides delimitation of abstracts by inserting XML-tags (based on comments in the input to TUC).

Example:

```
perl tqlextract.perl dumpfile > tqlfile.xml
```

## C.5 Creating own tests

This section provides a short tutorial on how to create your own test sets:

### C.5.1 Prepare data

The command examples are pseudobash.

1. Create a list of abstracts:
   $LIST = 'ls -1 abstracts/*.txt | cut -d '.' -f 1'

2. Concatenate them, and insert a comment before each abstract:
   for $i in $LIST; do echo "#KEEP MEDLINE ID $i >abs.e; cat abstracts/$i.e > abs.txt
   end

3. Prepare the same event-annotated abstracts:
   for $i in $LIST; do cat genia/$i.xml > gen.xml; end

### C.5.2 Run the abstracts trough GeneTUC

1. cd GENETUC2

2. ./genetuc.sav

3. ?- run.

4. E: \tell dump

5. E: \r abs

6. E: \told

7. E: \^ (quit to prompt)

8. perl tqlextract.perl dump > dump.tql

Now, dump.tql should contain an XML-skeleton with TQL.

### C.5.3 Extract and evaluate events

1. java EvEx gen.xml dump.tql

## C.6 Troubleshooting

Keep in mind that this software not is intended for distribution to end user, but rather as a developers tool for producing an evaluation of GeneTUC.

## C.6.1 XML-related problems

If the program does not evaluate the results, or evaluation is flawed, it is most likely because of an error in the XML files. EvEx uses SAXParser, which rejects XML that does not correspond to the W3C-definition. (See http://www.w3.org/XML/)

The files should thus be manually expected, and corrected.

## C.6.2 Execution-related problems

If the program does not execute, check that your version of the required programs corresponds to the requirements. If you have Java SDK, recompile *.java.