Semantic Enriching of Natural Language Texts with Automatic Thematic Role Annotation

Sven J. Körner and Mathias Landhäußer

Karlsruhe Institute of Technology (KIT),
sven.koerner@kit.edu, lama@ipd.uni-karlsruhe.de
WWW home page: http://svn.ipd.uni-karlsruhe.de/trac/mx/

Abstract. This paper proposes an approach which utilizes natural language processing (NLP) and ontology knowledge to automatically denote the implicit semantics of textual requirements. Requirements documents include the syntax of natural language but not the semantics. Semantics are usually interpreted by the human user. In earlier work Gelhausen and Tichy showed that SAL$_{E}$ MX automatically creates UML domain models from (semantically) annotated textual specifications [1]. This manual annotation process is very time consuming and can only be carried out by annotation experts. We automate semantic annotation so that SAL$_{E}$ MX can be completely automated. With our approach, the analyst receives the domain model of a requirements specification in a very fast and easy manner. Using these concepts is the first step into farther automation of requirements engineering and software development.

1 Introduction

Requirements engineering (RE) starts with the elicitation of the stakeholders' requirements, includes the management of the various user viewpoints and later leads to requirements analysis. Requirements analysis is often done by building domain models to visualize the processes. Domain models are used for *Model Driven Architecture* (MDA) [2], [1]. The requirements analyst uses these domain models to verify and rectify the stakeholders' input. Usually highly trained analysts build domain models manually. This process is as vital as time consuming in software development. So far, the analyst has little tool support [1] for his use-cases. A recent survey [3] shows that many practitioners yearn for improvements.

In 2007, Gelhausen and Tichy [1] showed how UML domain models can be created automatically from text that is enriched with semantic information. The models that are created in the SAL$_{E}$ MX [4] process are complete and exhaustive, especially compared to the average quality of a human modeler. Their work shows that there is a direct connection of natural language and its corresponding UML model representations. The automatic model creation uses the implicit semantics of a phrase. The semantics is denoted manually via textual annotations. This

[1] Many tools support the later stages of the software development process, e.g. CASE tools.

makes semantic information computer processable. The problem is that semantic annotation is very time-consuming. The idea is to accelerate RE with automatic semantic annotation. The enivisioned toolkit enables the average requirements analyst to create domain models rapidly.

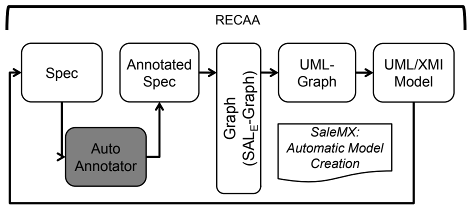


Fig. 1. AutoAnnotator Supports the Automatic Model Creation from Requirements

Our tool AutoAnnotator (see Fig. 1) offers exactly this possibility and helps to put RE where it needs to be: next to the software development processes that have been proven for many years. To decrease the high error rates, RE needs to be defined more clearly and to be less dependent on the human factor. Our solution integrates into a larger scope software solution RECAA [4] that covers requirements engineering from requirements elicitation to implementation and back. The analyst profits from automatic model creation from natural language specifications while documenting his understanding in cooperation with the stakeholder. RECAA maintains the connection between textual specifications and their model representation in both directions, and AutoAnnotator is an important part of this software solution.

The outline of this paper is as follows: Section 2 covers related work and the idea of automatic model creation. Section 3 describes our processing. Section 4 shows how we use semantic tools like ontologies to determine more complex structures and to verify the findings from NLP against world knowledge. Section 5 wraps up with a summary of our findings and the outlook to future work.

2 Related Work

Natural language is the main type of information in RE [3]. Its automatic processing is therefore especially interesting. Natural language was and will remain the main form of requirements documentation [5]. After elicitation, requirements are transformed into models that give a more formal representation of the described software system. These models are usually not intended for use with the client but with the software architects and the programming team. The average client cannot understand these models. As a result, the analyst usually maintains two models: one (semi) formal model for the development team, and one informal description in natural language for the client. These models have to be kept in sync during requirements evolution.The client signs a contract based on the model he understands. Dawson and Swatman argue in [6], that the mapping between informal and formal models is ad hoc and often results in divergent models. This strongly suggests to fill the gap between textual specifications and its models.

But requirements analysts cannot be replaced by NLP software, as Ryan argues in [7]. Most NLP tools are based on statistical approaches and have even greater error margins then human analysts. He highlights that NLP does not enable a machine to *understand* text, but it allows for a text's systematic transformation.

In 1997, Moreno [8] set the foundation for model extraction. Juzgado [9] et al. explain that a systematic procedure to dissect and process natural language information is strongly needed. They hint to the disadvantages of manual tasks which dominate the RE process until today. They postulate that this procedure must be independent from the analyst and his individual skills. In 2000, Harmain [10] developed CM-Builder, a NLP tool which generates an object oriented model from textual specifications. Additionally, Gildea and Jurafsky [11] showed in 2002 that statistical models can tag a sentence's semantics with a precision of 65% and a recall of 61%. Montes et al. describe in [12] a method of generating an object-oriented conceptual model (like UML class diagrams) from natural language text. Hasegawa [13] et al. describe a tool that extracts requirements models (abstract models of the system) from natural language texts. Instead of using only NLP, they perform text mining tasks on multiple documents to extract relevant words (nouns, verbs, adjectives etc.), assuming that important and correct concepts of the domain are contained in multiple distributed documents. Kof [14] reports that using NLP approaches is indeed feasible and worthwhile for larger documents (i.e. 80 pages and more). In [15] he shows that NLP is mature enough to be used in RE.

3 Combining NLP Tools - The Processing Pipeline

In this section, we explain how AutoAnnotator derives semantic information from syntactic sentence structures automatically. First we need to describe how Sal$_\text{E}$ mx [1] extracts UML domain models from natural language text using annotations.

3.1 How Sal$_\text{e}$ mx Works

Thematic roles [1] can be used to extract domain models from natural language text. As an example we use the sentences `Chillies are very hot vegetables. Mike Tyson likes green chillies. Last week, he ate five of them.` Using the syntax of Sal$_\text{E}$ [2], we need to tag the elements [3] with thematic roles. In Sal$_\text{E}$, elements containing more than one word are connected using a _ and

[2] Semantic Annotation Language for English [1].

[3] Elements can be atomic or combined parts of a sentence that represent a semantic entity and can therefore represent thematic roles.

Table 1. Linguistic Structures of SAL$_{\text{E}}$ (excerpt).

Linguistic Structure	**Explanation**
AG *agens*	An acting person or thing executing an action
PAT *patiens*	Person or thing affected by an action
ACT (+AG +PAT) *actus*	An action, executed by AG on PAT
STAT (+AG +PAT) *status*	A relation between AG and PAT
FIN (+FIC) *fingens* and *fictum*	The FIN plays the role of/acts like/is a FIC
TEMP (+ACT) *tempus*	A time specification TEMP for an ACT

obsolete elements are omitted using `#`. Furthermore, we prefix multiplicities with `*` and attributes with `$`. SAL$_{\text{E}}$ contains 67 thematic roles [4] based on the works of Fillmore and others [16],[17],[18]. For our example, the roles shown in Tab. 1 are sufficient. Manually annotating the text with SAL$_{\text{E}}$ results in the following:

```
1  [ Chillies|FIN #are $very $hot vegetables|FIC ].
2  [ Mike_Tyson|AG likes|STAT $green chillies|PAT ].
3  [ $Last week|TEMP, he|AG ate|ACT *five #of them|PAT ].
4  [ @he|EQD @Mike_Tyson|EQK ]. [ @them|EQD @Chillies|EQK ].
5  [ @chillies|EQD @Chillies|EQK ].
```

The thematic role *fingens* (`FIN`) is used to denote a person or thing that is playing a role; vice versa, *fictum* (`FIC`) is the role played by somebody or something. The word `are` is encoded in the *fingens/fictum* relationship and thus can be omitted. `very` and `hot` are attributes – the former attributing `hot`, the latter attributing `vegetables`.

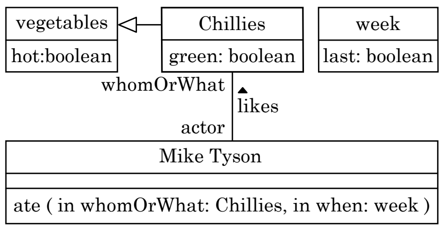
In the second sentence, the role *agens* (`AG`) tells the system that the according element is the active (not in a grammatical sense!) entity of the phrase. The *agens* in our case is `Mike_Tyson` which has been created by concatenating `Mike` and `Tyson` since it is an element consisting of two words. The role *actus* (`ACT`) is used for actions like *walk from A to B* while *status* (`STAT`) is used for general statements or relations like *A owns B*. Since `like` is a general statement, it is a *status*. Last but not least, we have `chillies` which is "the thing affected" by the *status* of Mike; therefore it is the *patiens* (`PAT`).

`TEMP` is a time, a date, or a "period". It modifies the roles it is used with in conjunction. Here `week` modifies *actus* and `last` is an attribute of `week`. `he` is the *agens* in the third phrase, performing the action `ate`. `them` is the thing being affected by the action of the *agens*, therefore it is the *patiens*. `five` is a multiplicity, determining the number of `them`. `of` is omitted.

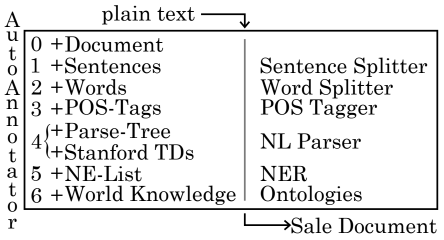
Knowing that `he` refers to `Mike_Tyson` and `them` refers to `chillies`, the analyst includes the assertions listed in line 4. SAL$_{\text{E}}$ MX replaces the element tagged with `EQD` [4] with a reference to the element tagged with `EQK` [5]. To preserve

[4] `EQD` is an acronym for "equal drop". The element is marked for replacement.

[5] `EQK` is an acronym for "equal keep". The element replaces one or more `EQD` elements.



(a) The UML class diagram generated with SAL_E MX.



(b) The Processing Pipeline of AUTOANNOTATOR.

the same case, we replace `chillies` with `Chillies` in line 5 [6]. With this input, SAL$_E$ MX generates an UML class diagram as shown in Fig. 2(a).

3.2 Automating Annotation

To achieve a streamlined process and a holistic information extraction, we combine several NLP tools and check the results against "digital common sense", i.e. world knowledge from an ontology (see Sect. 4). Our process is outlined in Fig. 2(b). It starts with the plain text. Every stage of the pipeline adds or verifies some information.

First, the text [7] is converted into an internal data structure (0). It contains the plain text aside the additional information gathered during the conversion into a graph structure.

After loading the text and splitting it into chunks (steps 0, 1, 2), it is processed with a part-of-speech (POS) tagger (3), a statistical parser (4), and a named entity recognizer (5). All tools used are from the Stanford NLP Group [19]. Afterwards, the document contains the following information:

(PENN-like) POS tags as described in [20]:

```
Chillies/NNS are/VBP very/RB hot/JJ vegetables/NNS ./.
Mike/NNP Tyson/NNP likes/VBZ green/JJ chillies/NNS ./.
Last/JJ week/NN ,/, he/PRP ate/VBD five/CD of/IN them/PRP ./.
```

Stanford Typed Dependencies (SD) as described in [21]:

```
nsubj(vegetables-5, Chillies-1), cop(vegetables-5, are-2),
advmod(hot-4, very-3), amod(vegetables-5, hot-4), ...
```

Named Entities The list of named entities contains only Mike Tyson.

[6] Normalization could render assertions like this unnecessary. The model extraction of SAL$_E$ MX is not yet capable of using this additional information.

[7] The text should comply to some rules: Since the described process is text-only, it should not contain images or rely on information given in images or illustrations. At the moment we cannot compute enumerations. They should be replaced beforehand.

Using the POS Tags and The Stanford Dependencies one can derive, that

- `Mike` and `Tyson` should be concatenated because they are in the same noun phrase (`NNP`), that consists only of them. On top of that, there is a named entity 'Mike Tyson'.
- `likes` is a verb having the subject `Mike Tyson` and the (direct, non-passive) object `chillies`. Since we do not know, if `like` is an action or a state, we can only tag it with `METHODROLE`[8]. Mike Tyson will be tagged with *agens* and chillies with *patiens*.
- `green` modifies `chillies`, and is not a number; thus it will be marked as attribute.

Similar deductions can be made for the first and the third sentence:

```
Chillies|FIN #are $very $hot vegetables|FIC.
Mike_Tyson|AG likes|METHODROLE $green chillies|PAT.
$Last week|TEMPROLE, he|AG ate|METHODROLE *five #of them|PAT.
```

Comparing the AutoAnnotator output with the manual annotation in Sect. 3.1, we realize that we need additional information to make the rest of the annotation decisions (6).

4 Semantic Information Enriching with Ontologies

We use two different knowlegde bases to gain the missing information: WordNet [22] and Cyc [23]. These ontologies are built upon concepts (of a domain) and relationships between these concepts and can be used to answer queries.

First we determine a word's base form with WordNet. Only containing open-class words (nouns, verbs, adjectives and adverbs), WordNet has simply four POS tags. The POS tags we discovered in step (3) restrict the search space when querying WordNet, as e.g. `ate` is not only included as verb with the base form `eat` but also as noun `Ate`[9]. The PENN tags allow us to parametrize the query. Since we do know, that `ate` has the PENN tag `VBD` and therefore is a (past tense) verb, WordNet does not produce the goddess as a result to our query.

The Cyc ontology is one of the most exhaustive and compelling collections of structured computable world knowledge. Cyc offers a vast collection of assertions between the ontological representation of many real world objects. Cyc delivers additional semantics to the problems discovered in Sect. 3. Let's revisit the second sentence of our example: We have found that the verb `like` is some kind of action or relationship between two entities. When asked about `like`, Cyc answers:

```
Predicate:  likesRoleInEventType
   isa:     FirstOrderCollectionPredicate, TernaryPredicate
```

[8] Our role system allows inheritance. `METHODROLE` is the (abstract) parent of *actus* and *status*. Using `METHODROLE`, we (internally) mark the element to be processed later.

[9] Ate is recorded as the Greek *goddess of criminal rashness and its punishment*.

```
Collection: TernaryPredicate
   genls:   Predicate, TernaryRelation
```

Cyc shows that `like` is predicate type of word (a *TernaryPredicate* to be precise). The collection of all *TernaryPredicates* itself has a generalization *Predicate*. Predicates are modeled as relations [24] and therefore the thematic role that has to be assigned is *status*.

5 Summary

Using sentence grammar structures to determine the correct semantics of a sentence seems feasible with our approach. We use popular NLP tools for the preprocessing of natural language texts. Even though AutoAnnotator is still work in progress, we have run a small qualitative case study using the technical specification of the WHOIS Protocol (IETF RFC 3912). The results suggest that the proposed approach is indeed capable of deriving the semantic tags of SaLEMX. Still there are some difficulties, which have to be addressed in future development.

First of all, subphrases are not yet handled correctly leading to confusing results. Errors of the pipelined NLP tools are not yet addressed adequately. Assigning a confidence value to each tool could improve results when information conflicts. On top of these future improvements, we plan to extend AutoAnnotator with an interactive dialog tool. This allows the analyst to steer the analysis process. We expect this interactive component to be used to resolve obvious mistakes the algorithms make as part of a feedback loop in the annotation process. Together with an instant UML diagram building process, the analyst could identify and correct the derived semantics on the fly.

Eventually, our process improves the annotation process with a speedup which we are currently evaluating. Only if the analyst is faster and receives the same quality models than in the manual process, automatic model creation can help support and improve the software development process.

References

1. Gelhausen, T., Tichy, W.F.: Thematic role based generation of UML models from real world requirements. In: First IEEE International Conference on Semantic Computing (ICSC 2007). Volume 0., Irvine, CA, USA, IEEE Computer Society (September 2007) 282–289
2. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1 (June 2003)
3. Mich, L., Franch, M., Inverardi, P.N.: Market research for requirements analysis using linguistic tools. Requirements Engineering **9**(1) (February 2004) 40–56
4. Körner, S.J., Derre, B., Gelhausen, T., Landhäußer, M.: RECAA – the Requirements Engineering Complete Automation Approach [Online].
5. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: Proc. Future of Software Engineering FOSE '07. (May 2007) 285–303
6. Dawson, L., Swatman, P.A.: The use of object-oriented models in requirements engineering: a field study. In: ICIS. (1999) 260–273

7. Ryan, K.: The role of natural language in requirements engineering. In: Proceedings of IEEE International Symposium on Requirements Engineering, IEEE (Jan 1993) 240–242
8. Moreno, A.M., van de Riet, R.: Justification of the equivalence between linguistic and conceptual patterns for the object model (1997)
9. Juzgado, N.J., Moreno, A.M., López, M.: How to use linguistic instruments for object-oriented analysis. IEEE Software **17**(3) (2000)
10. Harmain, H.M., Gaizauskas, R.J.: CM-Builder: An automated NL-based CASE tool. In: ASE. (2000) 45–54
11. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Computational Linguistics **28**(3) (September 2002) 245–288
12. Montes, A., Pacheco, H., Estrada, H., Pastor, O.: Conceptual model generation from requirements model: A natural language processing approach. In Kapetanios, E., Sugumaran, V., Spiliopoulou, M., eds.: NLDB. Volume 5039 of Lecture Notes in Computer Science., Springer (2008) 325–326
13. Hasegawa, R., Kitamura, M., Kaiya, H., Saeki, M.: Extracting conceptual graphs from Japanese documents for software requirements modeling. In Kirchberg, M., Link, S., eds.: APCCM. Volume 96 of CRPIT., Australian Computer Society (2009) 87–96
14. Kof, L.: Natural language procesing for requirements engineering: Applicability to large requirements documents. In Russo, A., Garcez, A., Menzies, T., eds.: Automated Software Engineering, Proceedings of the Workshops, Linz, Austria (September 2004) In conjunction with the 19th IEEE Internationl Conference on Automated Software Engineering.
15. Kof, L.: Natural language processing: Mature enough for requirements documents analysis? In Montoyo, A., Muñoz, R., Métais, E., eds.: NLDB. Volume 3513 of Lecture Notes in Computer Science., Springer (June 2005) 91–102
16. Fillmore, C.J.: Toward a modern theory of case. In Reibel, D.A., Schane, S.A., eds.: Modern Studies in English. Prentice Hall (1969) 361–375
17. Krifka, M.: Thematische Rollen (June 2005)
18. Rauh, G.: Tiefenkasus, thematische Relationen und Thetarollen. Gunter Narr Verlag, Tübingen, Germany (1988)
19. Manning, C., Jurafsky, D.: The stanford natural language processing group [Online].
20. Santorini, B.: Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). Technical Report MS-CIS-90-47, University of Pennsylvania Department of Computer and Information Science (1990)
21. de Marneffe, M.C., Manning, C.D.: The Stanford typed dependencies representation. In: COLING Workshop on Cross-framework and Cross-domain Parser Evaluation. (2008) 1–8
22. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM **38**(1) (1995) 39–41
23. Cycorp Inc.: ResearchCyc. http://research.cyc.com/ [checked 2010-02-15].
24. Körner, S.J., Gelhausen, T.: Improving automatic model creation using ontologies. In Institute, K.S., ed.: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering. (July 2008) 691–696