Extended Tagging in Requirements Engineering

Günther Fliedl, Christian Kop, Heinrich C. Mayr,
Jürgen Vöhringer, Georg Weber, Christian Winkler

University of Klagenfurt

Abstract. In this paper standard tagging mechanisms are discussed and partly criticized. We propose a semantically and morphosyntactically enriched mechanism in which many of the shortcomings of standard POS-taggers are eliminated. Therefore rule based disambiguation steps are presented. They include mainly a specification of contextually motivated verbal subcategories. We need this fine grained system for better preprocessing of requirements texts which have to be highly explicit and non ambiguous. This linguistic preprocessor can support an interpretation tool used to extract semantic concepts and relations for the requirements engineering step in software development.

1 Introduction

Classical tagging approaches use standardized (POS) tag sets. Such kind of standardized tagging (e.g., Brilltagger [1], TnT [2], Q-Tag [11] Treetagger [8], [10] etc.), however, show weakness in the following three aspects:

- Tags like 'VAINF' provide only basic categorial and morphological information;
- Ambiguity cannot be made explicit;
- Chunking and identification of multiple tokens is not possible.

To avoid such deficiencies, we developed a system called NIBA[1]-Tag which allows tagging of German with an extended tagset, and inheritance of morphosyntactic and morphosemantic features. Morphosemantic tagging in our sense is labeling words by morpho-syntactically relevant semantic classifiers (sem-tags like 'tvag2', 'eV', 'indef', 'poss', etc.; see Appendix 1 for a rough comparison with the Treebank [7] STTS [S*99]), It has proved to be an efficient method for extracting different types of linguistically motivated information coded in text. The XML-Tagger-output for the German PP (=P2) ***Bei Eintreffen des Auftrags*** in Table 1 shows how the tagging result is structured.

As can be seen in Table 1, our tagging system has the following linguistic competence:

[1] NIBA is a German acronym for Natural language based Requirements Engineering. The project NIBA is supported by the "Klaus Tschira Stiftung Heidelberg".

Table 1. Categories and features generated by NIBA-TAG

Bei (on)	*Eintreffen (arriving)*	*eines (of-the)*	*Auftrags (order)*
lowerCase="**bei**"	lowerCase="**eintreffen**"	lowerCase="**eines**"	lowerCase="**auftrags**"
id="**100713**"	id="**352487**"	id="**976515**"	id="**413424**"
Category = **p0**	Category = **v0**	Category = **det0**	Category = **n0**
type="**temp**"	referTo="**eintreffen**"	type="**temp**"	referTo="**Auftrag**"
	referid="**8264**"		referid="**18541**"
	base-form="**eintreffen**"		base-form="**Auftrag**"
	ps3_sg_praes_ind="**trifft**"	num="**sg**"	numerus="**sg**"
		kas="**gen**"	kasus="**gen**"
		gen="**masc**"	genus="**masc**"
	partikel="**ein**"		
	verbclass="**eV**"		
	verbclass-number="**2**"		
	pp="**eingetroffen**"		

1. The assignment of POS-tags enriched with morpho-syntactic features (e.g., Category = p0/v0/n0);
2. Morpho-semantically motivated subclass tags for verbs (e.g., verbclass = eV)
3. Identification of unknown words through assignment of affix-rules (e.g., kas = gen of *Auftrags*);
4. Extended lemmatizing (e.g., referTo = *eintreffen*).

This competence is used in the requirements engineering process. The rest of the paper is structured as follows. In chapter 2 some verb classes with high frequency are discussed. In chapter 3 we will represent the tagging rules. In chapter 4 we argue for a multilevel interpretation process during requirements engineering.

2 Verbclasses Used by NIBA-Tag

Since a merely morphological classification of verbtags as commonly practised by most taggers is not suitable for many tasks in the field of requirements engineering, a subclassification of the respective tags with respect to the verb arguments is necessary.

In our system German verbs are categorized with respect to the 26 verb classes, which are divided into 12 main-verb classes [3]. In the following we list definitions and examples of the top six most frequent tags for verb classes:

Verbclass 2 – **eV**: Ergative Verbs trigger non-agentive subjects (e.g., *sterben – to die*).

Verbclass 3 – **iV**: Intransitive verbs which need only one argument (the subject) (e.g., *schlafen – to sleep*)

Verbclass 6 – **psychV**: bivalent verbs with a psychologically motivated goal argument. No passivation is possible for that kind of verbs. (e.g., *ärgern – to annoy*)

Verbclass 7 – **tVag/2** verbs select two arguments: an agent role and a thema role. These verbs allow passivation; (e.g., *lesen – to read*).

Verbclass 7.2 – **tVag2pp**: Transitive Verbs with prepositional object. (e.g., *hineinschneiden – cut into sth.*)

Verbclass 8 – **tV3**: Ditransitive Verbs with an agentive subject and 2 objects (e.g., *Der Kollege gab mir einen Rat - the colleague gave me an advice*).

Verbclass 11 – **tV2**: Bivalent verbs with a non agentive subject and a thematic object (e.g., *Der Schwamm absorbiert die Flüssigkeit – the sponge absorbs the liquid)*.

3 Substantial Characteristics of the Tagging System

The system can be characterized through the following features and components [4]:

– A fine grained categorization system for open and closed German word classes. Lexicon categories are subdivided into semantically motivated subclasses, labelled with attributes. See Appendix 1;
– A fully functional lexical database with an integrated rule component for the optional generation of full German word forms;
– Export – and import functionality providing the import of simple structured excel sheets and export to XML, which can be converted to a tagger specific lexicon LeXML, a Berkeley DB);
– An extended rule based Perl tagger, including different types of context rules.

Subsequently, much effort had to be spent in the definition of context rules facilitating the disambiguation during the tagging process [5].

The following rule-types have been developed:

1. Feature filtering and generation rules;
2. Categorization rules;
3. Conversion rules;
4. Disambiguation and deletion rules;
5. Priorisation rules;
6. Chunk rules.

Feature filtering and generation rules Many features of lexical units are context-dependent; e.g.,

Bei[loc] → *Bei* [temp]/[Verbal noun]
e.g., *Bei Eintreffen des Auftrags (on arrival of the order)*
[Aux0] → [pass]/[past participle of tvag2, tv3]

Categorization rules Categorization rules are part of the lexicon component, initialized through inflectional expansion of basic word forms for the purpose of generating paradigms. NTMS based part-of-speech tags divide words into morphosyntactically and semantically motivated categories, based on how they can be combined to build up sentences.

Conversion rules Conversion rules are typical local context rules. They operate on lexically generated categorizations to transform them into different word classes, e.g., *eintreffen* (*to arrive*) v0 [eV]-> n0. In the XML-output, the original categorial grid remains visible.

Disambiguation and deletion rules They delete contextually not acceptable attribute values, e.g., the accusative in the context of '*des*'. Attributes which cannot be disambiguated, are being deleted likewise.

Priorisation rules During the tagging process attribute values are counted, whereas those with high frequency are priorized.

Chunk-rules Chunk rules refer to the results of the basic tagging process building phrasal categories from lexical categories, mainly N3 (NPs) and P2 (PPs). NPs are identified as expansions of nouns.

4 The NIBA Tag within Requirements Engineering

Niba Tag is currently used in the Requirements Engineering Project NIBA. In Software Development, Requirements Engineering is an important phase with the objective to find out the structural, functional and dynamic aspects of the software [9]. Extracting of functional, structural and behavioural aspects is thus one of the main tasks during this step. For the purpose of modeling requirements a modeling language called KCPM (Klagenfurt Conceptual Predesign Model) with a small set of modeling notions (*thing type, connection types, operation types, conditions*) is used to describe aspects of software. The notions are presented in glossaries. Thing types and connection types cover the structural aspects of software development. Operation types and conditions focus on the functional and behavioural aspect [6].

To extract these concepts out of the tagging results, two additional tools are necessary. A NIBA Dynamic Interpreter and a NIBA Static Interpreter were developed. This paper focuses on the Dynamic Interpreter since it also covers some structural aspects.

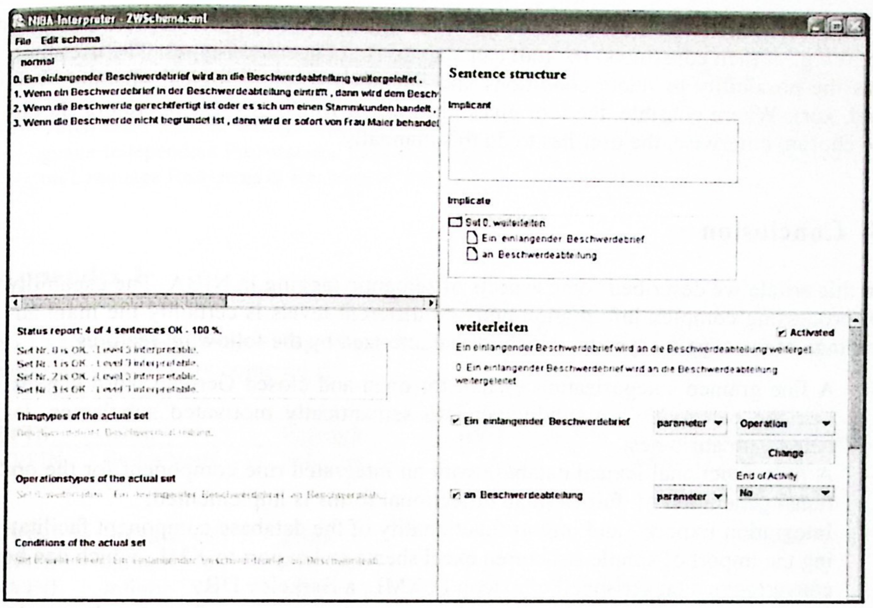
Before the interpreter can extract the concept it has to do some core linguistic work. The interpreter must assign syntactic functions and semantic roles to those word groups which are identified as syntactic phrases by NIBA-TAG. This is quite a difficult task for German sentences because of (morpho-) syntactic problems in German (free word order etc.). The interpreter makes linguistically motivated default decisions based on verb class related predicate argument structures (see Section 2). The assignments of the syntactic functions and semantic roles are then used for the interpretation process. The results are shown in the right upper and lower part of the window. The interpretation process presupposes decisions about the interpretability of the tagged sentences. Currently a first and simple model of four levels (level 0 – level 3) was introduced:

Level 0: No interpretation of the sentence is possible at all. This means that the sentence is written in such a way that the interpreter cannot find any structure within the sentence which can be used for interpretation (see level 1).

Level 1: At least one of the two parts *implication* and/or *condition* is found.

Level 2: Verbs are found, but cannot be associated with arguments.

Level 3: Candidates for verb arguments are recognized for the verb (if there is only one in the sentence) or for at least one verb (if there are more of them in the sentence).



0. Ein einlangender Beschwerdebrief wird an die Beschwerdeabteilung weitergeleitet .
1. Wenn ein Beschwerdebrief in der Beschwerdeabteilung eintrifft , dann wird dem Besch
2. Wenn die Beschwerde gerechtfertigt ist oder es sich um einen Stammkunden handelt ,
3. Wenn die Beschwerde nicht begrundet ist , dann wird er sofort von Frau Maier behandel

Sentence structure

Implicant

Implicate

Set 0. weiterleiten
Ein einlangender Beschwerdebrief
an Beschwerdeabteilung

Status report: 4 of 4 sentences OK - 100 %.

Set Nr. 0 is OK. - Level 3 interpretable.
Set Nr. 1 is OK. - Level 3 interpretable.
Set Nr. 2 is OK. - Level 3 interpretable.
Set Nr. 3 is OK. - Level 3 interpretable.

Thingtypes of the actual set

[illegible]

Operationstypes of the actual set

[illegible]

Conditions of the actual set

[illegible]

weiterleiten

✓ Activated

Ein einlangender Beschwerdebrief wird an die Beschwerdeabteilung weitergel

0 Ein einlangender Beschwerdebrief wird an die Beschwerdeabteilung weitergeleitet

✓ Ein einlangender Beschwerdebrief parameter Operation

End of Activity

✓ an Beschwerdeabteilung parameter No

Fig. 1 Main window of the interpreter

If a sentence can be interpreted on level 3, it is possible to derive KCPM notions. This can be controlled in the right lower corner of the window. For each verb and its arguments in the implication section (upper right part) the end user receives a default interpretation. If the verb is an agentive verb then it is mapped to an operation-type. The noun which is the syntactic subject of the sentence is mapped to the acting actor. The nouns which could be the syntactic objects are mapped to parameters. If the verb is not an agentive verb then it is interpreted as a condition. All the arguments of the verbs in the sentence are listed as candidates for involved thing-types. Those arguments which are possible thing types are filtered out based on a default filtering mechanism. If the sentence "*a person owns a car*" is taken as a condition, then both "*person*" and "*car*" are candidates for the involved thing-type but only one of these nouns is chosen as an involved thing-type (*person*). The rest of the sentence "*owns a car*" is then treated as the property of that thing-type necessary to fulfill a condition.

All interpreter results are understood to be "default", i.e., the user can always overrule default decisions. For example, the user can change the type of each thing-type (parameter, calling actor, acting actor) if he thinks that the default assumption is not appropriate.

Furthermore, the tool currently distinguishes between sentences (sentence parts) that are useful for interpretation and sentences which are not ("*fill-ins*"). In fact, the tool assumes that every sentence should be interpreted. However, there is a check box "Activated" which is "on" by default. If the user disables this check box, then the interpretation result will not be transferred into in the final schema.

In some cases (where the sentence fits with some given implicational sentence patterns e.g., if/then constructs) the tool can also derive cooperation-types. The user then has the possibility to relate conditions and operation-types to logical operators (or, and, xor). Where possible, the tool gives hints about which of these operators should be chosen, otherwise, the user has to do this manually.

5 Conclusion

In this article we described some aspects of semantic tagging in NIBA. The capability of processing complex information units on different levels is certainly the main advantage of our tagging system, which is characterized by the following features:

- A fine grained categorization system for open and closed German word classes. Lexicon categories are subdivided into semantically motivated subclasses, labeled with attributes;
- A fully functional lexical database with an integrated rule component for the optional generation of full German inflectional forms is implemented;
- Integration Export – and import functionality of the database component facilitating the import of simple structured excel sheets and export to XML, which can be converted to a tagger specific lexicon LeXML, a Berkeley DB;
- The linguistic tasks done by the tools presented above are a first step during the computer supported extraction process of concepts for software development.

References

1. Brill, E.: A simple rule-based part of speech tagger In: Proceedings of the Third Conference on Applied Natural Language Processing, ACL, 1992.
2. Brants, T.: TnT - A Statistical Part-of-Speech Tagger. In: Proc. of the 6th Applied Natural Language Processing Conference ANLP-2000. Seattle, pp. 224–231
3. Fliedl, G.: Natürlichkeitstheoretische Morphosyntax, Aspekte der Theorie und Implementierung. Habilitationsschrift, Gunter Narr Verlag, Tübingen, 1999.
4. Fliedl, G.; Kop, Ch.; Mayerthaler, W.; Mayr, H.C.; Winkler, Ch.; Weber, G.; Salbrechter, A.: Semantic Tagging and Chunk-Parsing in Dynamic Modeling. In: Meziane F.; Metais E.; (eds.) Proceedings of the 9th International Conference on Applications of Natural Language Processing and Information Systems, NLDB2004, Salford UK, Springer LNCS 3316 pp. 421 – 426.
5. Fliedl, G., Weber, G.: Niba-Tag - A Tool for Analyzing and Preparing German Texts. In: Zanasi, A.; Brebbia C.A.; Ebecken, N.F.F.; Melli, P. (Ed.): Data Mining 2002 Bologna: Wittpress September 2002 (Management Information Systems, Vol 6), pp. 331–337.
6. Kop C., Mayr H.C.: An Interlingua based Approach to Derive State Charts form Natural Language Requirements In: Hamza M.H. (Ed.): Proceedings of 7th IASTED International Conference on Software Engineering and Applications, Acta Press, 2003, pp. 538–543.
7. Marcus, M.; Santorini, B. and Marcienkiewicz, M.: Building a large annotated corpus of English: the Penn Treebank., Computational Linguistics, 1993.
8. Schmid, H.: Probabilistic Part-of-Speech Tagging using Decision Trees. www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger1.pdf, 1994; www.ifi.unizh.ch/stff/siclemat/man/SchillerTeufel99STTS.pdf.

9. Schach, Stephen R.: An introduction to object-oriented analysis and design with UML and the unified process, McGraw Hill, Boston, Mass., 2004.
10. Schiller, A.: Teufel, S.; Stöckert Ch.; Thilen Ch.: Guidelines für das Tagging deutscher Textcorpora mit STTS.
11. Tufis, Dan; Mason, Oliver.: Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. Proceedings of the First International Conference on Language Resources & Evaluation (LREC), 1998, pp. 589–596.

Appendix 1

STTS (Stuttgart-Tübingen Tagset) vs. NIBA-Tagset and Feature-System (only some example-verbtags are compared):

STTS	Gloss	Example	NIBA tagset		Attribute values
VVFIN	finites Verb, voll	[ich] lese	tVag/2	Transitive	[ps1], [sg], [ind], [pres]
VVINF	Infinitiv, voll	gehen	iV	Intransitive	[inf], [stat1]
VVINF	Infinitiv, voll	ankommen	eV	Ergative	[inf], [stat1]
VVINF	Infinitiv, voll	trinken	tVag/2	Transitive	[inf], [stat1]
VVIZU	Infinitiv mit *zu*, voll	auszuatmen	iV	Intransitive	[inf], [stat2]
VVIZU	Infinitiv mit *zu*, voll	anzukommen	eV	Ergative	[inf], [stat2]
VVIZU	Infinitiv mit *zu*, voll	loszulassen	tVag/2	Transitive	[inf], [stat2]
VVPP	Partizip Perfekt, voll	gegangen, gelesen			[stat3]
VAFIN	finites Verb, aux	[du] bist, [wir] werden	AUX		
VAIMP	Imperativ, aux	sei [ruhig !]	Vcop		[imp]
VAINF	Infinitiv, aux	werden, sein	Vcop		[inf], [stat1]
VAPP	Partizip Perfekt, aux	gewesen			[inf], [stat3]
VMFIN	finites Verb, modal	dürfen	AUX		[mod], [ps1], [pl]
VMINF	Infinitiv, modal	wollen	AUX		[mod], [inf]
VMPP	Partizip Perfekt, modal	gekonnt, [er hat gehen] können			