PROCEEDINGS OF SCIENCE

A Tool for Collaborative Rule–Based Morpheme Annotation

Dietmar Seipel[*] [a] **and Luise Borek** [b]

[a] *University of Würzburg, Department of Computer Science, Am Hubland, D – 97074 Würzburg, Germany*

[b] *Technical University of Darmstadt, German Computational Philology, Hochschulstrasse 1 (S 103/182), D – 64289 Darmstadt, Germany*

E–mail: dietmar.seipel@uni-wuerzburg.de | borek@linglit.tu-darmstadt.de

Dictionaries register basic units of languages, and they are therefore useful resources for starting research into language structures. The availability of dictionaries in electronic form has opened up new opportunities for a detailed analysis that can contribute to a deeper understanding of the registered units and the dictionary structures in general.

We are developing a morpheme annotation tool that in a first step adds a fine–grained morphological annotation to every dictionary entry and in a second step combines the information from various heterogeneous dictionaries into a comprehensive database. It holds a supplementary morpheme segmentation for each dictionary lemma. We use a set of synchronic and diachronic dictionaries of the German language as a test bed for our project. Making use of word formation rules transformed into PROLOG, we have generated a powerful system that goes beyond single applications.

The morpheme annotation tool allows for collaboratively annotating large–scale digital data. At the same time, it helps to overcome further existing problems in the field, namely cross–development and individual solutions that are kept away from the professional community. To achieve this, we have expanded the implemented rules and improved the tool in terms of usability. A documentation of our rule system will be made fully available and may serve as a matter of examination itself.

The International Symposium on Grids and Clouds and the Open Grid Forum
ISGC 2012
February 26 – March 02, 2012
Academia Sinica, Taipei, Taiwan

[*]Speaker.

PoS(ISGC 2012)017

Contents

1. Introduction

Within the collaborative research project *Variance in Language and Genome* [15], funded by the German Federal Ministry of Education and Research since 2008, we are integrating various electronic dictionaries into a metaDictionary, which will include a fine–grained annotation of morphemes [10]. The diachronic variance of the morpheme complexes is then characterized and compared to the variance in genome data in bioinformatics. The starting point of the research project is a collection of digitized synchronic and diachronic dictionaries made publicly available at the University of Trier (*Trierer Wörterbuchnetz* [14]). Additionally, we use the *Campe Dictionary* [1] as well as the *Wörterbuch der deutschen Gegenwartssprache (WDG)* [7], digitized by the Berlin–Brandenburgische Akademie der Wissenschaften. Based on the broad and representative database, the goal of the project is to develop and test methods and algorithms for detecting and understanding variance.

For building the metaDictionary, we are developing a *morpheme annotation tool*. In this paper, we extend the focus of the tool beyond its project shaped purpose. We present

newly implemented features that complement the original goals, widening the range of both our tool and the resulting metaDictionary. We put a focus on morpheme structures and implemented (word formation) rules that enable a wide range of additional functions. Instead of merely allowing to examine the distribution and combinatorics of morphemes in large–scale dictionary data, we have extended the implemented annotation rules to enable the tool to detect and analyze *word formation* structures.

This approach strongly benefits from the interaction of the involved disciplines. On the one hand, it builds on rule–based structures of language that can overlap with information structures in computer science, which allows transforming those rules, making them ready for (re)application. On the other hand, natural language processing (NLP) is faced with the unattainable challenge of dealing with large–scale language data in a strictly automated way. Naturally, these kinds of data also contain *irregular patterns* that build exceptions to the implemented rules. That is why we have chosen to keep the tool open for user interaction, at the same time enabling an easy adaptation of the rules and the flexibility towards further development.

Morphemes are the smallest meaningful units in a language, and they serve as a construction kit for all kinds of word formation. They are involved in these processes in different ways: *lexical morphemes* can be used to construct new words, *affixes* form a new word when they are added to an existing one. Using German dictionaries as our source, we have obtained a morphological segmentation with the publicly available Morfessor tool, thus enriching the data with information that had only been present implicitly. By adding extra information, e.g., a comprehensive list of affixes, we have extended the tool's functionality and created a powerful multidimensional database, whose potential we have further exploited.

Structure of the Paper. The rest of this paper is organized as follows: In Section 2 on morphological segmentation and annotation, we outline some basic concepts for the analysis of word formation and formation rules. Section 3 sketches basic programming concepts and data structures, which we use in our tool. The tool for analyzing the morphological structure of dictionary entries and annotating their parts is presented in Section 4. Section 5 explains the extended flexibility and usability. Some conclusions are given in Section 6.

2. Morphological Segmentation and Annotation

As described earlier, we have used *declarative parsing* techniques for the compilation of the metaDictionary based on a fine–grained annotation from the underlying dictionaries [10]. With dictionary input as source, it is possible to take along some default annotations, such as the *language stage*, and further specific annotations for each lemma, such as the *part of speech*, that are available in the electronic text. We further add a morphological segmentation of each lemma to the existing data making use of the Morfessor tool; incorrect segmentations can be identified by comparison with available confirmed data.

The interactive tool will provide alternative suggestions which can be approved (or further altered) by the user. Thus enriched, we have got a foundation for an automatized approach that is based on *word formation* and *systematic rules* sharing the understanding that is presented by Fleischer and Barz [3]. Richly annotated lexical morphemes qualify to pass on their information to any word formation they occur in. For our bottom–up–approach, we have supplied a *gold standard* of correctly annotated lexical morphemes that will be passed on throughout the tool. In addition, a set of affixes is integrated into the tool, that enables us to apply word formation rules and to distinguish lexical morphemes from those that are part of the word formation process. Whenever the tool has a distinct annotation available for a morpheme, it is added automatically. When there are various options due to homographs, the user will be notified and asked to confirm either of the options before the information will be saved.

Another annotater – for English, German, and Italian – that is also working with word formation rules is part of the well–known, commercial LanguageTools of Canoo Engineering [2]. In contrast, our tool is open source, and it can be extended and refined flexibly. In our project, we work with German dictionaries and German word formation rules; the annotation tool, however, can be applied to other languages as well, if we can supply the morphemes, the affixes, and the corresponding word formation rules. In this paper, we have selected some English examples to illustrate the tool; the relevant word formation rules in these examples are analogous in German and English.

2.1 Word Formation

The pre–annotated and segmented lemma lists from various German dictionaries result in a morpheme network that allows for a refined analysis. In German, new words can be created by – at least partly – making use of existing units and processes.

Retrospectively, substantial parts of the lexicon are built following specific rules. In addition, one is dealing with of a limited number of morphemes, that can be combined to form the lexicon. This contributes to its variety and ability to change. In our tool, we benefit from productive patterns of word formation and their transformation into generic annotation rules in Prolog. We have chosen the most common systematic rules for derivational and compounding processes.

As explained above, the data integrated into the tool is prepared in a way that allows the rules to work effectively, as the required key players, lexical morphemes, affixes, and word formation rules, have been given. Not only are the sets of lexical morphemes and functional affixes known, also the parts of speech of each word are pre–confirmed, since they originate from a dictionary. The combination and order of their occurrences enables the tool to detect which rules are relevant to determine what kind of word formation process has taken place; this annotation is then added as well.

Additionally, the confirmed parts of speech serve for evaluation purposes, as inconsistencies within a word formation would become obvious. Bracketing of contained units mirrors the order and structure of the underlying processes for a word formation.

2.2 Formation Rules

The majority of the rules of the annotation tool formalizes standard functional patterns of word formation. The dominating concepts in German word formation are *derivation* and *composition*. For derivational processes, the involved affixes and their effects are required. Rules for composition use patterns, which examine the part of speech of the morphemes and the compound they form. `(collar*lm*n+bone*lm*n)*mc*n` is an example of a composition. Clearly, combinations of both processes are common.

A segmentation that lemmatizes single morphemes tends to give up information that could be valuable for a statistical analysis. In order to be able to analyze the morphemes as they occur within a word, we have implemented a flexible solution, which allows the original lemma to remain unchanged; removing all annotations would leave the original word unharmed. This holds an advantage over other methods that visualize word formation processes by enlisting the units involved in their unbound conditions. However, the aim of automatically annotating dictionary input requires rules to process the written forms *as they are*. `(sky+(div*xbm(dive)+ing*da)*v*derv)*v` is an example of a derivation, that shows how the extra annotation `xbm` handles altered bound morphemes.

Moreover, a couple of supplementary rules have been implemented to enhance the efficiency and functionality, e.g., *implicit rules* and *exclusion rules*. They help to identify valid conclusions and allow for a precise adjustment as well as a slim computing. Another factor taken into account is the position a morpheme can or cannot take within a complex word. This especially applies for affixes. Here, considering the position is absolutely required. It is relatively easy to determine a suffix, when it is the last element of the text form of a word. It could, however, appear embedded, when its derivate functions as a unit within a more complex word formation, as does `less` in `((use*lm+less*suffix)+ness*suffix)*noun`. As a result, the rules have to be recursive. *Exclusion rules* help finding correct annotations automatically to keep the manual interaction load low. Since on the word level the part of speech is known already, it will never be suggested as an additional annotation: `bear*noun` will not be suggested to be a verb at the same time, even though `verb` is known to the tool as a valid annotation to that text form.

We have decided to keep the annotation process as transparent as possible; this has resulted in a *bottom–up approach*, in which only lexical morphemes pass on their annotations into more complex word formations. However, an additional adjustment is required for lexical morphemes that are not used as free forms. This applies to all verbs, as dictionaries commonly record infinitive forms leading to a default segmentation. Thus, the sub–term `X*lm*v` of `(X*lm*v+Y*ia)*v` is allowed to pass on information. In the examples above, the part of speech annotation of a lemma is given. Lexical morphemes bound in word formations, however, lack this information. Their parts of speech are added in our bottom–up approach from annotated lexical morphemes. The set of underlying word formation rules can then determine the word formation process for the annotated formation. Whenever a grammatical morpheme is involved, an *implicit rule* tags the result as *derived*, as, e.g., in `((right*lm*adj+eous*da*suffix)*adj+ness*da*suffix)*noun*derv`.

3. Declarative Programming Concepts and Data Structures

We use *declarative programming* in PROLOG for parsing, querying, and transforming the linguistic data [4, 9]. There is a variety of well–known declarative languages available for different areas: SQL for relational databases, XQuery and XSLT for XML processing, PROLOG for declarative programming, and rules for decision support systems and grammars. Their advantages are, that they are compact, rapidly programmable, clear, less error–prone, and flexibly extensible.

We have tested declarative knowledge extraction by parsing dictionary entries with definite clause grammars (DCG) on the Adelung Dictionary and – within the framework of the TEXTGRID–Project [13] – on the Campe Dictionary [1] for a fine–grained lexicographic analysis. For compiling the metaDictionary, we have split the lexemes from the electronic dictionaries into morpheme terms. We are developing a PROLOG–based tool for a declarative, rule–based control and annotation of morpheme segmentations. We can extend our approach by methods for a more precise alignment within the metaDictionary.

Mathematical structures (terms, atoms, rules) are the basic *data structure* of PROLOG. If we represent the annotated morpheme decompositions as terms, then we can very compactly and flexibly encode and extend the annotation information. Analogous to the *word formation rules* for natural languages, there are syntax rules for forming structures in the programming language PROLOG. Strings are atomic structures. For an n–ary operator $\odot$ and structures t_i, $\odot(t_1,\ldots,t_n)$ is a complex structure. An infix notation $t_1 \odot t_2$ exists for some binary structures $\odot(t_1,t_2)$: e.g., the infix term `((craft+s)+man)*noun+ship` is equivalent to the prefix term `+(*(+(+(craft,s),man),noun),ship)`. Also rules are complex structures – of the form `Head :- Body` – in infix notation. E.g., the following rule states that a morpheme complex `X` should be annotated with the word class `noun`, if its first component `A` is a noun (`has_word_class`) and its second component `B` has a text form from a given list `[ship, ...]` including `ship`.

```
has_word_class(X, noun) :-
   mc(X, A, B), has_word_class(A, noun), text_form(B, [ship, ...].
```

4. The Morpheme Annotation Tool

The challenge was to build a tool that can deal with large–scale digital data – an endeavor that formerly required an exhaustive amount of human supervision. For annotating the large numbers of dictionary entries (which can exceed 100.000 units), one needs linguistic knowledge and suitable tools from computer science: a reliable morphological analyzer, a suitable, compact knowledge representation, inference methods for the annotation rules, and a graphical user interface. The architecture of our annotation tool, which we have built using PROLOG, is shown in Figure 1.

The decomposition of complex morphemes into basic units is based on the Whole Word Morphology. The initial morpheme decompositions are derived using well–established tools such as Morfessor. Then, they are pre–annotated based on the fine–grained information

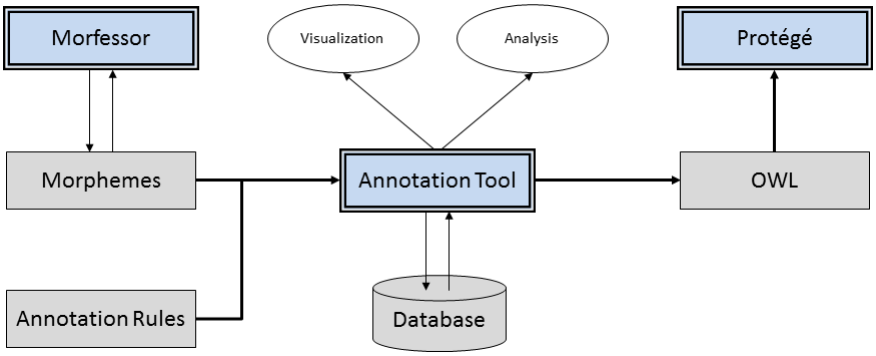

Figure 1: System Architecture.

of electronic dictionaries. We use available dictionary information to support the analysis, e.g., information on word classes and conditions of usage. The basic morphological units found can later be corrected, refined and annotated interactively with our tool. We try to extract as many annotations as possible from the underlying electronic dictionaries using our parsing techniques.

4.1 Annotated Morpheme Terms

We use a compact knowledge representation of the annotated morpheme decompositions as suitable term structures, which can be managed elegantly in PROLOG.

Using unique abbreviations, such as `lm` for lexical morpheme, `da` for derivational affix, and `le` for linking element, the morpheme term `((craft+s)+man)+ship` for the English word `craftsmanship` can be annotated as `(((craft*lm*noun+s*le)+man*lm*noun)*noun*comp+ship*da*suffix)*noun*derv`, which is visualized in Figure 2.

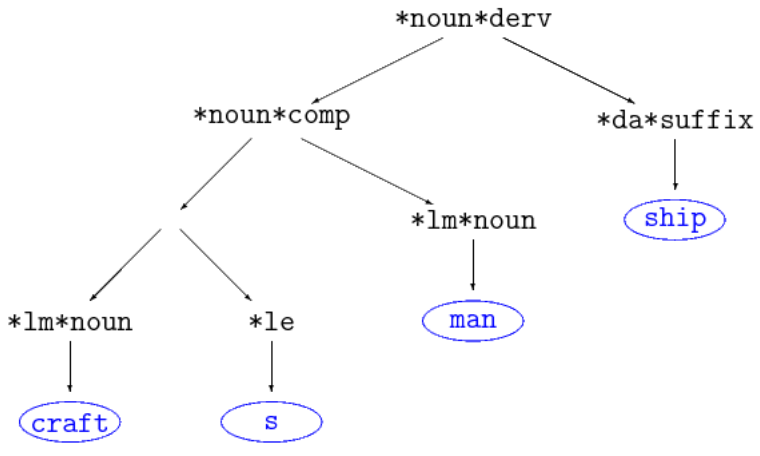

Figure 2: Annotated Morpheme Decomposition.

For persistently storing the morpheme terms and for efficiently accessing the morpheme terms based on their text form, a relational database is used. For every morpheme term, it

contains further information, such as the text form, the identifier from the dictionary, and the name of the user who annotated the term as well as the time stamp of the annotation.

4.2 Annotation Rules

Making use of specific systematic language rules to achieve solutions automatically was one task, treating exceptions and allowing the user to interfere, modify existing rules, or add new ones, another. Using the declarative programming language PROLOG, we have implemented a set of fifty word formation rules that – in combination with exclusion rules – allow for the automatic detection of the part of speech and the underlying formation process. As PROLOG's data structures are formed by terms, it is suited excellently for the representation of morpheme structures and enables flexible encoding and expanding.

The processing of the morpheme terms is completely done on the PROLOG side. In the background, a set of PROLOG rules is managed, which can automatically infer further annotations from the annotations and decompositions given by the user. With the following logical annotation rule, the term `((craft+s)+man)*noun+ship` is recognized as a noun and can be further annotated to `(((craft+s)+man)*noun+ship)*noun`:

```
has_word_class(X, noun) :-
   mc(X, A, B), has_word_class(A, noun), text_form(B, [ship, ...]).
```

Moreover, we have established a hierarchy of rules, in which some are applied automatically and others result in suggestions to the user, who can confirm their validity. Confirmed rules can climb in hierarchy and will then be applied automatically.

4.3 The Graphical User Interface

The morpheme annotation tool supports the user in checking all pre–annotated morpheme decompositions. The user can annotate parts of a morpheme decomposition. Based on a set of annotation rules – which can be extended dynamically – the morpheme decomposition is further annotated as detailed as possible. The graphical user interface, which is also implemented in PROLOG, consists of the morpheme editor, a visualization of the morpheme terms as tree structures, and a database browser, cf. Figure 3.

To keep all processes transparent within the tool and to encourage trans–disciplinary bridging, we have implemented suitable features in the graphical user interface, such as a combined expanding and updating mechanism, abbreviated annotation tags, and an update function that facilitates minor changes.

The database window at the bottom of Figure 3 shows, that the lemma `gold` has the annotation `gold*lm*noun`. For the text form `en`, the database contains three different annotations: `en*da*suffix`, `en*le`, `en*ia*suffix`. This information can be transferred to word formations containing `gold` or `en`. Without any expansions or updates performed, the adjective `golden` appears as `(gold+en)*adj`; the segmentation has been obtained using Morfessor. Using the `Expand&Update` function opens a database update window (in the middle, on top), where the user can choose a suitable annotation for `en` and complete

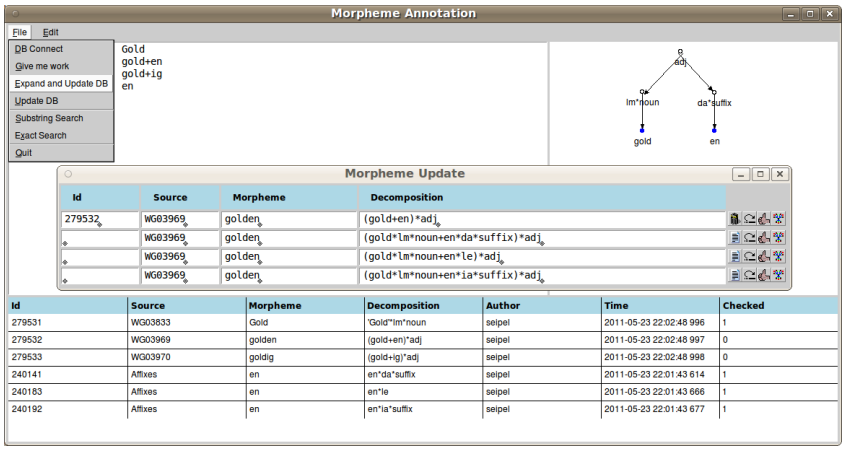

Figure 3: The Graphical User Interface of the Morpheme Annotation Tool.

the term to `(gold*lm*noun+en*da*suffix)*adj`. The selection could be supported by suitable exclusion rules. Currently, the kind of word formation is not yet determined in this step; this will be automatized in a later stage. But the annotation `derv` could also be added manually to make it available immediately.

The tree structure of the annotated morpheme terms can be visualized (see top right window). The annotated morpheme terms are stored in a relational database. It is possible to efficiently search for individual annotated morpheme terms (`Exact Search`) or for all morpheme terms containing a search string (`Substring Search`).

4.4 XML Export of Morpheme Terms and Annotation Rules

The morpheme decompositions can be exported to the Web Ontology Language (OWL) and imported into standard tools, such as the collaborative ontology editor Protégé [12].

XML Export of Morpheme Terms. Following its tree structure, the annotated morpheme term `(gold*lm*noun+en*da*suffix)*adj` is represented in XML as follows:

```
<term>
   <term>
      <term text="gold">
         <annotation>lexical morpheme</annotation>
         <annotation>noun</annotation>
      </term>
      <term text="en">
         <annotation>derivational affix</annotation>
         <annotation>suffix</annotation>
      </term>
   </term>
   <annotation>adjective</annotation>
</term>
```

This XML representation can be transformed to OWL and imported into Protégé, which enables supplementary features utilizing some of the strengths provided by Protégé, namely the visualization and analysis of hierarchic structures.

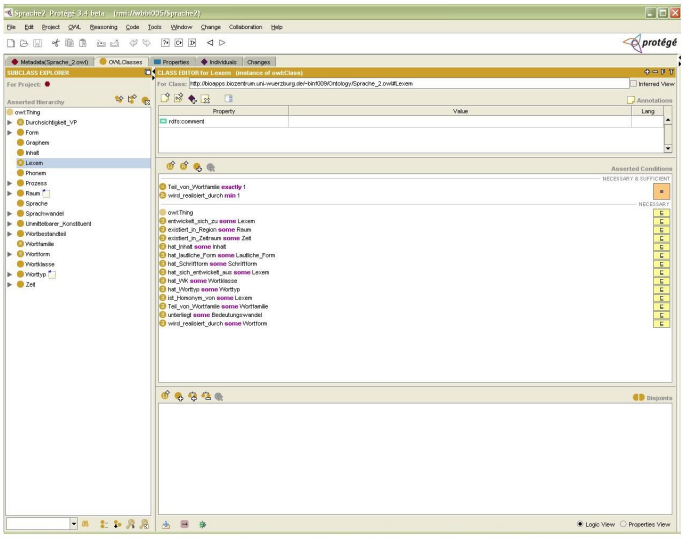

Figure 4: Protégé.

XML **Export of Annotation Rules.** The Semantic Web Rule Language (SWRL) was developed as a simple rule syntax in connection with OWL. Thus, apart from Built–Ins, SWRL can only represent unary or binary structures, like `has_word_class(A, noun)`, and no function symbols, like in list structures `[ship, ...]`. Thus, we export a rule

```
has_word_class(X, noun) :-
   mc(X, A, B), has_word_class(A, noun), text_form(B, [ship, ...]).
```

into a more powerful XML representation, that is inspired by SWRL:

```
<rule>
   <head>
      <atom predicate="has_word_class" arity="2">
         <var name="X"/> <const name="noun"/> </atom>
   </head>
   <body> ...
      <atom predicate="text_form" arity="2">
         <var name="B"/>
         <list> <const name="ship"/> ... </list> </atom>
   </body>
</rule>
```

A documentation of our rule system will be made fully available and may serve as a matter of examination itself.

Analysis of Morpheme Terms and Annotation Rules. Based on the XML/ OWL export of the annotated dictionaries, which will be made publicly available for other researchers, it becomes possible to perform more refined statistical analyses. We have started research into word formation models, that could analyse how often a pattern appears on a language stage, and how its usage changes over time. There can, e.g., be purely structural patterns, such as `X = (Y*lm*noun+Z*da*suffix)*adj*derv`, to investigate how often a lexical morpheme `Y` that is also a noun is combined with a word formation morpheme `Z` to derive an adjective `X`. Other patterns can also depend on the text forms (e.g., `X*noun+bar`). Finally, even the logical annotation rules themselves can be analysed.

5. Flexibility and Usability of the Extended Annotation Tool

By extending and shaping the rules in our annotation tool, we have achieved a fine–grained annotation of morphemes and their combinations. Dealing with natural language requires an amount of flexibility and adaptability. Exceptions and doubtful cases occur and need to be treated within the same environment. The solutions we have embedded to take this into consideration also make it possible to extend the tool for further questions. From the background of our initial research question, we have chosen dictionary input and could therefore focus on word formation rather than, e.g., inflection. However, the design of the tool allows for implementing corresponding rules analogously. Rules can be kept in a try–out–mode, where their functionality is presented to a user in the form of suggested annotations. Only when the systematic quality is approved, the rule obtains a fully automatic status.

In terms of usability, we have introduced new features to the graphical user interface. These include a function that provides the user with entries that require manual confirmation (`Give me work`) and the already described `Expand&Update` function. An additional `Update` function enables the user to submit slight changes within an annotated context without having to go through the complete process of suggestions for annotations.

The extended tool offers a great potential for *search queries*. In addition to scanning written forms, customized search queries also make it possible to track down annotations or specific patterns. A query designed for a specific research question leads to results that contribute to their identification and definition. Moreover, these insights may lead to further extensions that continue to advance the tool.

6. Conclusions

We have substantially extended the functionality of the morpheme annotation tool, and thus introduced essential innovations to our *generic e–infrastructure* for analyzing language data. The original structure, that was aimed at the detection and analysis of variance, has been enriched by implementing word formation rules. In addition, usability could be improved and mechanisms that facilitate customization and further expansion have been integrated. Besides the extensions described in this paper, we have outlined how

the new design enables a flexible access for further user–oriented approaches. Thus, not only is existing data continually enriched, but also new perspectives become visible that researchers can benefit from.

We have developed an interoperable tool that is fully compatible and ready for *collaborative research*. Its design makes it easily adaptable to specific tasks. We are aiming at an integration into the virtual research environment TextGrid. Making use of this generic e–infrastructure will support the acceptance of the tool and enable specific studies as well as further participation. The annotated dictionary data are stored in Xml data formats. We will also provide a format conformant with the Tei P5 Guidelines for Electronic Text Encoding and Interchange [11]. The exported dictionary data will be publicly available for other researchers, e.g., for researching into variance comparable to genome structures.

References

[1] Campe, Joachim Heinrich: *Wörterbuch der deutschen Sprache.* 5 Vol., 1807–1811.

[2] Canoo Engineering AG. *Canoo LanguageTools.* `http://languagetools.canoo.com`.

[3] Fleischer, Wolfgang; Barz, Irmhild: *Wortbildung der deutschen Gegenwartssprache.* 4th revised edition. Niemeyer, Tübingen, 2012.

[4] Gazdar, Gerald; Mellish, Chris: *Natural Language Processing in* Prolog. *An Introduction to Computational Linguistics.* Addison–Wesley, 1989.

[5] Gouws, Rufus et. al.: *Wörterbücher / Dictionaries / Dictionnaires. ... / An International Encyclopedia of Lexicography / ... Vol. 2*, de Gruyter, Berlin / New York, 1990.

[6] *GOLD (General Ontology for Linguistic Description).* `http://www.isocat.org`.

[7] Klappenbach, Ruth; Steinitz, Wolfgang (Eds.): *Wörterbuch der deutschen Gegenwartssprache.* 6 Vol., Akademie–Verlag, Berlin, 1961–1977.

[8] Michel, Jean–Baptiste, et al.: *Quantitative Analysis of Culture Using Millions of Digitized Books.* Science 14, January 2011, pp. 176–182.

[9] Schneiker, Christian; Seipel, Dietmar; Wegstein, Werner; Prätor, Klaus: *Declarative Parsing and Annotation of Electronic Dictionaries.* Proc. 6th International Workshop on Natural Language Processing and Cognitive Science (NLPCS), 2009.

[10] Seipel, Dietmar; Wegstein, Werner: *metaDictionary. Towards a Generic e–Infrastructure for Detecting Variation in Language by Exploiting Dictionaries.* Proc. International Symposium on Grids and Clouds (ISGC), 2011.

[11] *P5: Guidelines for Electronic Text Encoding and Interchange.* `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html`.

[12] *The Protégé Ontology Editor.* `http://protege.stanford.edu`.

[13] TextGrid*: A Modular Platform for Collaborative Textual Editing – a Community Grid for the Humanities.* 2009, `http://www.textgrid.de`.

[14] *Trierer Wörterbuchnetz – Network of Electronic Dictionaries of the University of Trier.* `http://www.woerterbuchnetz.de`.

[15] *Variance in Language and Genome*: Project funded by the German Ministry of Education and Research (BMBF) since 2008. `http://www.sprache-und-genome.de`.