# ICALL Activities for Gerunds vs. To-infinitives: A Constraint Grammar-based Extension to the New WERTi System

**Niels Ott & Ramon Ziai**
Seminar für Sprachwissenschaft
Universität Tübingen
72074 Tübingen, Germany
{nott,rziai}@sfs.uni-tuebingen.de

## Abstract

This paper describes the design and implementation of automatically generated ICALL activities on authentic texts. These activities allow ESL learners to practice the proper use of the gerund vs. the to-infinitive. Results show that the automatic creation of such activities is feasible but further research is necessary.

## 1 Introduction

Computer-Aided Language Learning (CALL) allows students to do language learning exercises or lessons on a computer. Usually these exercises and lessons are prepared in advance by instructors. Intelligent Computer-Aided Language Learning (ICALL) adds Natural Language Processing (NLP) to CALL. In the subfield of Authentic Text ICALL (ATICALL), the preparation on exercises is conducted on the fly on text material chosen by the learner. Amaral et al. (2006) present WERTi, a system providing language awareness activities to learners in of English as a Secondary Language (ESL). Those activities include the color highlighting of targeted language phenomena (receptive presentation), and cloze tests having the learner fill in the blanks (controlled practice).

Dimitrov (2008) presents a re-implementation of the original Python-based WERTi system using Java-based enterprise technology. The presented work describes the extension of this new WERTi system by three new activities. These activities focus on the use of the gerund vs. the to-infinitive in English, allowing the learner to practice when to use which. Depending on the context, it is appropriate to use either the gerund verb form or the to-infinitive. In some situations, they can be used interchangeably, in others they cannot. The proper use therefore must be practiced by learners of English.

We discuss the linguistic grounds of gerunds and to-infinitives. Furthermore, we put the focus on clue phrases. These are little words and phrases that help the learner in choosing the right option (gerund or to-infinitive). To implement the computational analysis of the linguistic phenomenon, Constraint Grammar is employed (Karlsson, 1990). Last but not least we report on the integration of Constraint Grammar in the enterprise architecture of the new WERTi, including the realization of the named activities using the Google Web Toolkit (GWT).

## 2 Linguistic Patterns

### 2.1 The To-infinitive

The to-infinitive is easy to recognize on the surface. As the name says, it consists of the word *to* and the uninflected form of a verb. However, the to-infinitive is seen in a more restricted sense in grammar teaching. Grammar books such as Ungerer et al. (1989) distinguish between the to-infinitive used in constructions such as the going-to-future and other uses. The variant we are mainly concerned with is the one occurring as subject or object argument to other verbs. A few example cases:

He wants *to attend* the lectures today.

Will the teacher be able *to explain* these facts?
Students are *to drink* beer.

### 2.2 The Gerund

What is a gerund? This question is easily answered by any learner grammar book, but after a closer look, the situation turns out to be more complicated. Obviously, not every verb form ending in *ing* is a gerund. The simpler cases are the progressive forms and the going-to future such as those in:

He was *writing* a morphological analyzer.
What are you *doing* here?
They are *going* to do their homework.

Linguistically more debated is the distinction between the participle and the gerund. Huddleston and Pullum (2006, p. 1120) refute the existence of a basis for this distinction: "We call this form gerund-participle to reflect the fact that it covers the ground of both gerunds and present participles in other languages." Richardson (1991) points out that the participle can occur in a "adverbial function", adding information to the verb, as well as in a "nominal function", while the gerund can only can function in the nominal sense.

She started to run, *leaving the homework undone*.

While in the above example, the phrase holding the -ing form is a participle providing a closer specification to the verb, the very same phrase could be used as a gerund:

He imagined *leaving the homework undone*.

Richardson concludes that "distinguishing a gerund -ing form and a participle -ing form with distinct paradigms adds unnecessary complexity (i.e. to the grammar)." The production of gerunds and participle certainly works with the same mechanics. Linguists are debating about whether the distinction makes sense at all.

For practical reasons in ICALL, we need this distinction: the gerund -ing form can be used exchangeable with the to-infinitive, while the participle -ing form cannot. Hence if the learner is to decide whether to use the to-infinitive or the gerund in an activity, we must ensure that the task does not mutate to an to-infinitive vs. participle activity. Therefore we must make the distinction in the very same way as learner grammars such as Ungerer et al. (1989) are making it; in the given scenario there is no such thing as the "gerund-participle" of Huddleston and Pullum. Instead, it is assumed that there is a clear-cut line between the gerund and all other forms, including the participle and the progressives.

### 2.3 Clue Phrases

#### 2.3.1 Towards a Definition

Clue phrases play a role in language learning and in prescriptive grammar[1]. Clue phrases work in this manner: the question whether to use the gerund or the to-infinitive is often answered by certain little words or phrases being present, entailing the answer. These phrases simply are patterns of usage. Furthermore, they are irrelevant for most descriptive views on the gerund and the infinitive. In the work presented, we distinguish two basic types of clue phrases, with the first one exhibiting four subtypes:

- Verbs that take as an argument either to-infinitives or gerunds, in some cases both.
  - Always gerund.
  - Always to-infinitive.
  - Both, the meaning is the same.
  - Both, but the meaning differs.
- Fixed expressions that are always followed by the gerund.

Most learner grammars include lists, e.g. Alexander (2007, p. 315). The learner can infer rules from this information, such as 'always use the gerund after *appreciate*' or 'always use the to-infinitive after *want*'.

Although we need to make the fine-grained distinction presented above for computational processing, it is not represented on the user interface level. We consider it to be sufficient to highlight clue

[1] Huddleston and Pullum (2006, p. 5) use the term "usage manuals" instead. Prescriptive grammars are meant to give advise to those who are uncertain about language use, while descriptive grammars in linguistics are aiming to describe the language use of those who are certain about it.

phrases as such, letting the learners make the connection to the intended usage themselves.

### 2.3.2 Grammar Books vs. Language Use

Prescriptive grammars run the risk of being victims of "taste tyranny" (Huddleston and Pullum, 2006, p. 7). Is the grammar book's advise of using the gerund vs. the to-infinitive really compatible with the English of the real world? For learners writing texts, this might be irrelevant as they simply need something to hold on to. But for processing free text, as the described piece of software is to do it, it is questionable whether clue phrases can be relied on if their usage is not confirmed by real world usage patterns.

In a corpus-based experiment, we tested 71 verbs taken from grammar books listing them as clue phrases. The classification into the four subtypes discussed in the previous section was applied as stated in the books. We then performed automated queries to the British National Corpus (BNC, Burnard 2007). Four patterns were queried for each verb:

1. *verb* + to + infinitive
2. *verb* + preposition to + infinitive
3. *verb* + -ing form
4. *verb* + preposition + -ing form

The first category produces virtually no hits in the corpus, so it can be disregarded. Furthermore, some verbs are low-frequency words. All verbs occurring less than 25 times for all three remaining cases together were dropped, leaving 63 observations. The results are depicted in figure 1; the dots accumulating in the corners indicate that there actually is a number of verbs being used with either the gerund or the to-infinitive. This insight of course is restricted to the language use as it is represented by the BNC.

Equipped with these results, we tried to reclassify all verbs on the basis of their number of occurrence. If a *verb*+to+infinitive pattern occurred more than 90% of the patterns 1–3 given above and if the corresponding -ing form patterns occurred less than 25 times, we classified the verb as 'always with to-infinitive'. Seen from the other direction, we attached the 'always with gerund' attribute. Verbs being situated between the 90% lines were labeled as 'both gerund and to-infinitive'. The rest was declared 'undecidable'.

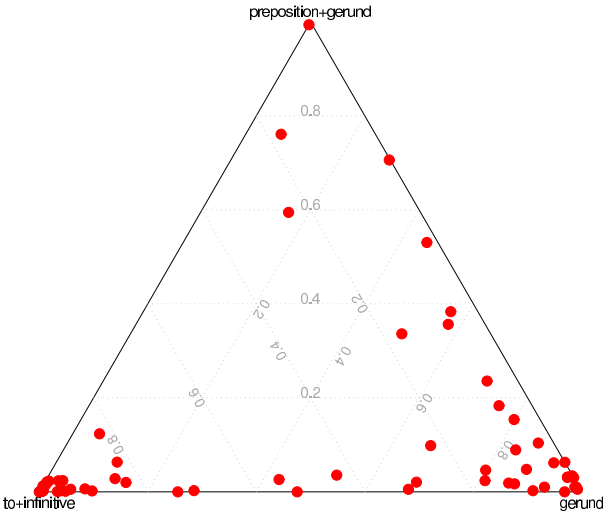


Figure 1: 63 verbs used with to-infinitive vs. gerund in the British National Corpus.

From our classification and the classification given in the grammar books, we computed Cohen's Kappa statistics for inter-rater agreement. It computes to $\kappa = 0.314$, a value indicating only debatable agreement between the (descriptive) corpus observations and the (prescriptive) grammars.

From these results we conclude that for retrieving reliable verbs usable as clue phrases, more research must be done. For now, we retreat to the ideal world of 'should-be language' found in learner grammars.

## 3 Natural Language Processing

### 3.1 Standard Tagsets provide only -Ing Forms

The new WERTi system uses a POS-tagger with a statistical model trained using the Brown Corpus (Francis and Kucera, 1979). Consequently, the analysis available is based on the Brown Corpus tagset. Neither this tagset nor the CLAWS tagset nor the Penn Treebank Tagset distinguish between the -ing forms of the verb (Leech et al., 1994; Santorini, 1990). May this fact be due to the linguistic convictions of those who created the tagsets or may it be a technical choice in favor of simplicity, the endeavor described in the presented paper supports the

design decision to leave aside the difficulties of distinguishing -ing forms in corpus annotation.

Nevertheless, the distinction needs to be made for the given purpose. We decided to implement a rule-based approach on the basis of Constraint grammar.

### 3.2 Constraint Grammar

#### 3.2.1 The Original

Parsing always involves dealing with ambiguous sentences. Traditional parsers produce the more parse trees per sentence, the more rules the grammar includes. Constraint Grammar, initially described by Karlsson (1990), turns the entire affair upside down: a Constraint Grammar parser starts out with all ambiguities that are known, subsequently eliminating as many of them as possible. In an ideal world with an ideal grammar, there is only one single analysis remaining per sentence in the end. There is another point to mention: Constraint Grammar fosters the use of partial or shallow analysis. If the annotation of a few words or phrases in a sentence is enough to produce the required analysis, no full parse and no large-coverage grammar of the language in question is required.

#### 3.2.2 Introducing Ambiguity only to Resolve It

Our approach differs slightly from the procedure described by Karlsson (1990). While he includes morphological analyses and their disambiguation, we start out with readily available disambiguated POS-tagger output. However, since the tagset does not distinguish -ing forms, ambiguity is introduced for each -ing form. Initially, the parser is confronted with three possible readings: progressive form, gerund, participle. For the word form *doing*, we additionally introduce readings for going-to future and going-to future in the past.

The to-infinitives are trivial to handle: uninflected forms are tagged as such and the infinitival *to* is assigned its own private tag.

It is important to keep in mind that in ICALL, precision must be focussed. It is not elegant to miss occurrences of gerunds in the analysis but it is inexcusable to declare an -ing form as a gerund that in fact is something else. So if it is possible to safely identify the going-to future as a first step, this reduces chances to do wrong later on. Hence our rules are ordered to detect linguistic phenomena with ascending difficulty: going-to future (in the past), progressive forms, participles, gerunds. If all stages fail, we know that no analysis of the given -ing form was possible with these rules–which is still better than casting a false positive gerund.

```
SUBSTITUTE ("" VBG) ("---remove---"
    VBG) (VBG);
APPEND ("" VBG GERU)   (VBG);
APPEND ("" HVG GERU)   (HVG);
APPEND ("" BEG GERU)   (BEG);
APPEND ("" VBG PROG)   (VBG);
APPEND ("" HVG PROG)   (HVG);
APPEND ("" BEG PROG)   (BEG);
APPEND ("" VBG PART)   (VBG);
APPEND ("" HVG PART)   (HVG);
APPEND ("" BEG PART)   (BEG);
REMOVE ("---remove---" VBG);
"<going>" APPEND ("" VGB GOFU)
    (VBG);
"<going>" APPEND ("" VGB GOFUPA)
    (VBG);
```

Figure 2: Introducing multiple readings[3] for -ing forms .

The analysis conducted is kept as shallow as possible. For example, an argument of a verb is simply represented by 'everything that is not a verb and not a clause delimiter.'[2] After being run through the disambiguation section, there may still be words with more than one reading. For these ambiguous cases, all readings are removed and a simple reading saying 'this is still ambiguous' is introduced. While the grammar consists of 98 rules altogether, the disambiguation rules for -ing forms sum up to only 32. Example pieces from the grammar are shown in figures 2 and 3.

#### 3.2.3 Clue Phrase Detection

Since clue phrases are to be presented by the learner, they must be detected by the program. This is done using the verbs named in section 2.3.2 as a sure-fire list. In English, these verbs are always situated preceding the occurrence of the gerund or to-infinitive. However, there can be additional modifiers in be-

[2] Pragmatic definition of a clause delimiter: comma, period or a word tagged as a subordinate conjunction.

[3] The empty string ("") denotes an empty lemma which is required for technical reasons. Since we are not using any lemmatization at that stage of processing, they are all empty.

```
LIST _3P_PRES = VBZ BEZ DOZ VHZ;
LIST _3P_PAST = BEDZ DOD HVD VBD;
SELECT (GERU) ( +1* _3P_PRES BARRIER
    _COMMA_OR_SUBORD OR
    _ANYVERB_ANYFORM);
SELECT (GERU) ( +1* _3P_PAST BARRIER
    _COMMA_OR_SUBORD OR
    _ANYVERB_ANYFORM);
SELECT (GERU) ( +1* (MD) BARRIER
    _COMMA_OR_SUBORD OR
    _ANYVERB_ANYFORM LINK +1 _UNINF);
```

Figure 3: Example rules for selecting gerunds from ambiguous readings: the gerund with possible arguments of its own as a subject 3rd person argument to verbs.

tween. This is shown in the following example with the clue phrase *kept* suggesting the use of the gerund *waiting*:

> He *kept* her *waiting* for too long.

It turns out that these long-distance clue phrases are problematic in some situations. The 'everything that is not a verb and not a clause delimiter'-strategy (cf. section 3.2.2) for identifying arguments of verbs fails in the case of gerunds or to-infinitives being arguments to nouns or other POS. In the following example, the occurrence of *denies* is analyzed as a clue phrase. This is wrong, because the gerund construction *(of) writing* actually attaches to *habit*.

> The author *denies* his habit of *writing* morbid books.

For fixed expression clue phrases, we use a simple list of contexts taken from Ungerer et al. (1989). Elaborate information on these expressions is rare in learner grammars so further research should take into account harvesting such expressions by cautious corpus exploration.

```
ADD (GER-B) (GERU) (0C (GERU));
ADD (GOI-B) (GOFU) (0C (GOFU));
ADD (GOP-B) (GOFUPA) (0C (GOFUPA));
ADD (PRO-B) (PROG) (0C (PROG));
ADD (PAR-B) (PART) (0C (PART));
```

Figure 4: Marking chunks for all disambiguated -ing forms: unlike for the to-infinitive, these are all single-token chunks.

#### 3.2.4 Marking Chunks

After the disambiguation step, our constraint grammar inserts chunk tags marking ranges that are eventually used for presenting the linguistic phenomena to the user. These chunk tags consist of 'begin' and 'inside' tags such as `INF-B` and `INF-I` for the to-infinitive and `CLU-B` and `CLU-I` for clue phrases.[4]

There are three basic types of chunks: to-infinitives, gerunds, and clue phrases. The classification of other -ing forms plus ambiguous -ing forms is marked with chunks as well. It can be used for debugging purposes.

Ranges spanning a to-infinitive and a clue phrase or a gerund and a clue phrase are marked as `RELEVANT` chunks as those are to be presented to the user in all three activities. Although this type of annotation allows overlapping spans, we stick to the convention that a `RELEVANT` span must contain exactly one `CLU` (clue phrase) span and either one `INF` (to-infinitive) or `GER` (gerund) span. Apart from that there are no other nested spans.

With this strategy, the entire linguistically motivated application logic is loaded into the constraint grammar.[5] An example for marking chunks is shown in figure 4.

## 4 Processing in WERTi

### 4.1 The Architecture of WERTi

The new WERTi as developed by Dimitrov (2008) makes use of three enterprise technologies: Java Servlets together with the Apache Tomcat Servlet Container[6] provide the basis on which Apache UIMA (Ferrucci and Lally, 2004) and the Google Web Toolkit (GWT)[7] operate. The required NLP pipeline is constructed as UIMA components, while the user interface as well as the client-server communication is implemented using GWT.

Figure 5 depicts the data flow in WERTi, omitting the details of client-server communication. The URL Fetcher retrieves the web page to process, the NLP pipeline in UIMA produces all analyses re-

[4] This strategy is inspired by the IOB (inside, outside, begin) chunk tagging introduced by Ramshaw and Marcus (1995).

[5] Apart from POS tagging as a given step of preprocessing and the lemmatization of relevant -ing forms, see section 4.4.

[6] `http://tomcat.apache.org/`

[7] `http://code.google.com/webtoolkit/`

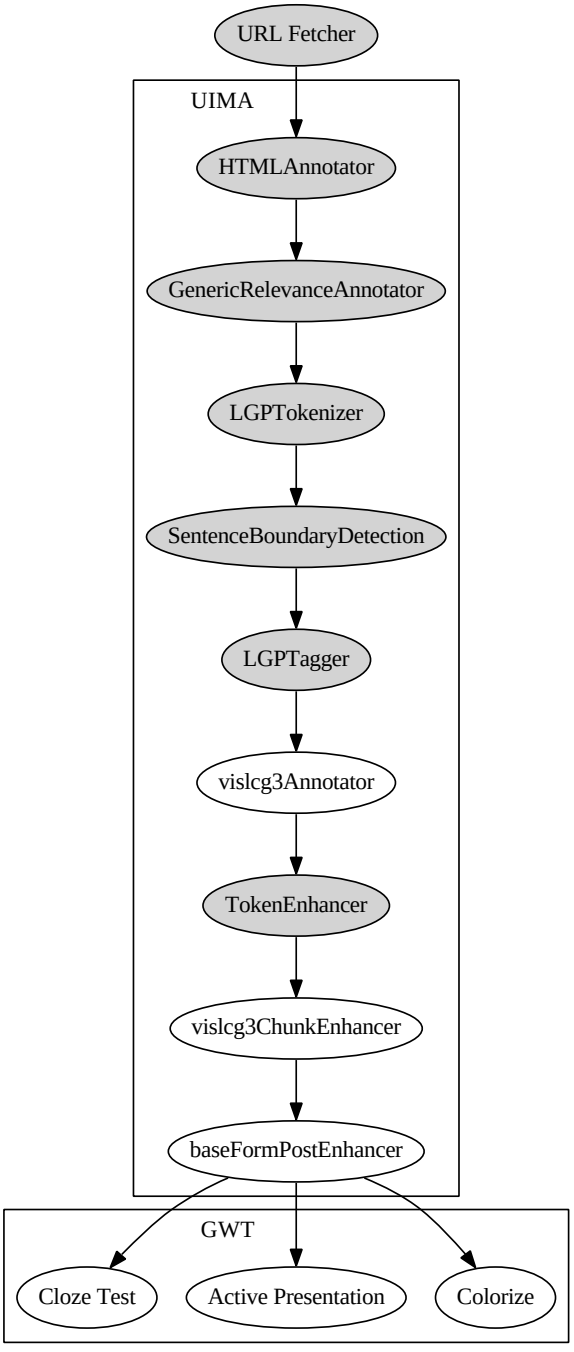


Figure 5: Simplified representation of the data flow in WERTi, gray elements are part of the base system introduced by Dimitrov (2008).

quired for the desired ICALL activities, which then are on the user interface side implemented as GWT modules. Elements present in the base system are shown in gray.

The initial modules *HTMLAnnotator* and *GenericRelevanceAnnotator* ensure that only those parts of the web page that contain the actual text are processed. This is followed by tokenization, sentence boundary detection and POS tagging with the Lingpipe Tagger.[8]

The *vislcg3Annotator* is a generic wrapper around the vislcg3 Constraint Grammar parser[9]. The *vislcg3ChunkEnhancer* is more specific: it expects chunk tagging as described in section 3.2.4 to have happened before and prepares the insertion of GWT modules for the desired activities. The *baseFormPostEnhancer* extends the output of the previous module by lemmata where needed.

Last but not least, three GWT modules implement the actual ICALL activities. In our scenario, all three modules make use of the very same linguistic analysis and the very same preparation conducted by the enhancer modules. Other activities may require to branch the data flow at a much earlier step, e.g. by making use of a different vislcg3 grammar that entails the use of another specific enhancer module.

### 4.2 A VislCG3 Wrapper for Preprocessing

*vislcg3* is a constraint grammar compiler developed by Tino Didriksen and Eckhard Bick at the Syddansk Universiteit. It was fashioned after the older CG-2 compiler presented in Tapanainen (1996) and is currently used as the computational core of the Visual Interactive Syntax Learning project (Bick, 2001).

For our purposes, we needed a UIMA wrapper around the standalone *vislcg3* program. It converts UIMA token annotations into the text format expected by *vislcg3* and feeds it to the program. The output of *vislcg3* is then parsed and put into UIMA data structures so that further analysis components can draw on it. Multiple readings are realized as a

[8] Dimitrov (2008) reports using the tokenizer of Stanford Tagger and TreeTagger as tagger. Concerning the current status of the system, this information is outdated. Both tagger and tokenizer are currently those of the Lingpipe toolkit. `http://alias-i.com/lingpipe/`

[9] `http://beta.visl.sdu.dk/cg3.html`

feature of a token in the UIMA type system and represented as lists of strings where one element corresponds to what is called a 'tag' in constraint grammar.

### 4.3 Enhancing HTML Pages

The re-implementation of WERTi by Dimitrov provides the annotation type of generic enhancements. This type ranges over a certain span of the HTML document in question and further allows the association of arbitrary HTML tags with a certain enhancement.

To stick with this approach, we implemented the *vislcg3ChunkEnhancer* which relies on annotations made by the wrapper described above. In our vislcg3 grammar we insert chunk tags as described in section 3.2.4. These chunks are identified by the chunk enhancer and for every chunk a new enhancement is constructed and added to the annotations.

We insert generic HTML `span` tags that are associated with special IDs in order to be able to identify them later on the client side. The client module can then decide on a visualization strategy that suits the intended activity.

### 4.4 Lemmatization as a Step in Post-processing

Activities such as cloze tests may additionally require the base form of the verb, e.g. in order to display it to the user as instruction. Consequently, lemmatizing functionality was needed which we implemented with the help of the *morpha* analyzer described in Minnen et al. (2001).

In the case of the to-infinitive, deriving the base form is unneeded as it is already present. For the gerunds, we feed the -ing form to *morpha* and add the output to the enhancement created by the chunk analyzer. As a result, the activity module can draw on this information.

### 4.5 Activities implemented using the Google Web Toolkit

Three activities were realized using the same underlying analysis:

- A *Colorize* activity that marks occurrences of gerunds and to-infinitives and their respective clues.
- An *Active Presentation* that requires the learner to identify clues by clicking on them after being presented with gerunds and to-infinitives.
- A *Cloze* (fill-in-the-blank) exercise that highlights the clues and prompts the learner for the correct forms.

In each case the same HTML document is offered to the client browser but the JavaScript Code generated by GWT differs for each activity. As the analysis is done on the server side, all the client code needs to do is substitute certain tags in the HTML for the desired visualization. For example, in the *Colorize* activity, we just added a certain style to the `span` tags already there in order to make the tokens appear in a different color.

The most complicated activity is *Cloze* as it needs to check the user input and provide visual feedback. Here we built on a reusable component already provided in WERTi which encapsulates the functionality of a fill-in-the-blank box.

## 5 Development and Testing

While we constructed the grammar we worked with experimental sentences that were constructed manually in order to present typical cases that our grammar had to deal with. The grammar grew in parallel to the software components that were needed to integrate the approach in WERTi. Since all the linguistic intelligence is in the grammar, the other components could be developed independently and would then profit from the grammar improvements.

We did not conduct a full-scale evaluation experiment on our experiments. For a meaningful evaluation, one would need to deploy the system in a real English Language Teaching environment which we hope to do in the near future.

Nevertheless, we naturally tested our system in the process of development. A problem one immediately notices is that not all sites contain sufficient occurrences of the phenomena, so one needs to select specific sites known to contain gerunds[10] (the to-infinitive is far more frequent).

[10] We ran most of our tests on texts from the 'Young Readers' section of classicreader.com, see `http://www.classicreader.com/browse/3/title/`

Our tests show that the system is able to deal with most typical occurrences, such as "I've kept her *waiting*". It fails if the tagger (which we rely on) misses fixed expressions, as in "*bathing* machines" and mistakenly marks them as possible gerunds. In general, tagging accuracy highly influences the quality of our analysis.

There is also the problem of limited context we use in our grammar rules. Then again, if one is to cover the harder cases, the grammar becomes significantly more complicated because long-distance dependencies within a sentence need to be taken into account and the functions of subject and object would need to be assigned. Within the scope of this project, such a thorough analysis was not possible.

## 6 Conclusion

We have discussed the design and implementation of ICALL activities on authentic texts concerning the contrast between gerunds and to-infinitives. Our preliminary tests done during development show promising first results but a full-scale empirical evaluation is needed. Without deployment in a real learning environment with a sufficient number of users, meaningful analysis of the results is hardly possible.

As detailed in section 2.3.2 some clue phrases cannot really be justified as reliable indicators of either to-infinitive or gerund. Consequently, more data-driven (e.g. corpus-based) research is needed in order to end up with better clues. The mismatch between grammar books and real-world language is a hindrance in developing consistent ICALL activities for language learners.

Another problem is the limited frequency of the gerund. In many texts, gerunds are simply not used, for stylistic or other reasons. Such texts will then produce disappointing results for learners. Further research should include pre-selecting texts based on occurrences of certain constructions, such as gerunds, from the domain of interest to the learner.

Nevertheless, several important points can be drawn already from this project. Constraint Grammar has proven to be well-suited to ICALL activities with authentic texts because it does not attempt to cover a language (fragment) with regard to completeness. As a result, one can implement exactly the rules one needs for an activity and no others.

Furthermore, we have seen that it is feasible to implement ICALL activities on authentic texts within a reasonable time span using the new WERTi system.

## Acknowledgements

We would like to thank Aleksandar Dimitrov for the invaluable and constant support he provided during the development of our software.

## References

L. G. Alexander, 2007. *Longman English Grammar*. Longman, London.

Luiz Amaral, Vanessa Metcalf and Detmar Meurers, 2006. Language Awareness through Re-use of NLP Technology. In *Pre-conference Workshop on NLP in CALL – Computational and Linguistic Challenges*. CALICO 2006, University of Hawaii.

Eckhard Bick, 2001. The VISL System: Research and applicative aspects of IT-based learning. In *Proceedings of NoDaLiDa (Uppsala)*.

Lou Burnard, 2007. *Reference Guide for the British National Corpus*. Oxford University Computing Services, Oxford.

Aleksandar Dimitrov, 2008. *Rebuilding WERTi: Providing a Platform for Second Language Acquisition Assistance*. Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.

David Ferrucci and Adam Lally, 2004. UIMA: An architectural approach to unstructured information processing on the corporate research environment. *Natural Language Engineering*, 10(3–4), 327–348.

W. N. Francis and H. Kucera, 1979. *Brown Corpus Manual*. Department of Linguistics, Brown University, Providence, Rhode Island.

Rodney Huddleston and Geoffrey K. Pullum, 2006. *The Cambridge grammar of the English languag*. Cambridge University Press, Cambridge, 4 edition.

Fred Karlsson, 1990. Constraint grammar as a framework for parsing running text. In *Proceedings of the 13th conference on Computational lin-*

*guistics*. Association for Computational Linguistics, Morristown, NJ, USA, pp. 168–173.

Geoffrey Leech, Roger Garside and Michael Bryant, 1994. CLAWS4: the tagging of the British National Corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*. Kyoto,Japan, pp. 622–628.

Guido Minnen, John Carroll and Darren Pearce, 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3), 207–223.

Lance A. Ramshaw and Mitchell P. Marcus, 1995. Text Chunking Using Transformation-Based Learning. In David Yarovsky and Kenneth Church (eds.), *Proceedings of the Third Workshop on Very Large Corpora*. Association for Computational Linguistics, Somerset, New Jersey, pp. 82–94.

Andrew Richardson, 1991. A Grammatical Description of the English Gerund and Related Forms. Ph.D. thesis, University of Essex.

Beatrice Santorini, 1990. *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*. Technical Report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania.

Pasi Tapanainen, 1996. *The Constraint Grammar parser CG-2*. Number 27 in Publications of the Department of General Linguistics. University of Helsinki.

Friedrich Ungerer, Peter Pasch, Peter Lampater and Rosemary Hellyer-Jones, 1989. *Grundgrammatik: Ausgabe für Gymnasien*. Learning English. Ernst Klett Schulbuchverlag, Stuttgart, Germany, 1 edition.