Developing and Exploiting a Multilingual Grammar for Human-Computer Interaction

Xian Zhang, Rico Andrich, and Dietmar Rösner

Department of Knowledge Processing and Language Engineering,
Otto-von-Guericke-University Magdeburg,
P.O. Box 4120, D-39106 Magdeburg, Germany
{xzhang,roesner}@iws.cs.uni-magdeburg.de,
ricoandrich@gmx.de

Abstract. How to build a grammar that can accept as many as possible user inputs is one of the central issues in human-computer interaction. In this paper, we report about a corpus-based multilingual grammar, which has the aim to parse naturally occurring utterances that are used frequently by subjects in a domain-specific spoken dialogue system. The goal is achieved by the following approach: utterance classification, syntax analysis, and grammar formulation.

Keywords: NLU, HCI, grammar, multilinguality, GF.

1 Introduction

In human-computer interaction (HCI), the problem of grammatical coverage, which means how to build a grammar that can accept as many as possible user inputs, is one of the central issues in spoken dialogue systems.

In this work we are thus interested in the problem of developing a grammar that is powerful enough to parse as many as possible naturally occurring utterances that are frequently used by subjects in a domain-specific spoken dialogue system. The idea is to first classify user utterances in order to detect the commonly used patterns and concepts from a corpus, and then find out the syntactic structures and summarize them into rules to build up the grammar.

The development was based on the transcripts of the NIMITEK corpus [1] that collects recordings of affected German speech in human-computer interaction (HCI) gathered in a Wizard-of-Oz (WOZ) [2] experiment that was conducted by Gnjatovic and Rösner [3] since 2006. The NIMITEK corpus is a multimodal corpus of affected behavior in human-machine interaction [4]. It contains 15 hours of audio and video recordings that are produced during the refined WOZ experiment, which was designed to induce emotional reactions of the human participants. During the experiment, the language of the subjects was very natural because the subjects were only allowed to give speech based instructions to the system and no language restrictions were given to them.

In this work, we use one of the tasks in the NIMITEK corpus named TANGRAM as the sample. Tangram is a popular match game. Of seven stones, namely, five triangles, a square and a parallelogram, we can put figures together. All pieces must

J.A. Jacko (Ed.): Human-Computer Interaction, Part II, HCII 2011, LNCS 6762, pp. 396–405, 2011.

be used. They can be moved, rotated, or reflected; they have to touch each other, but must not overlap. Fig. 1 shows the seven stones that are used in Tangram, and an example of Tangram figure.

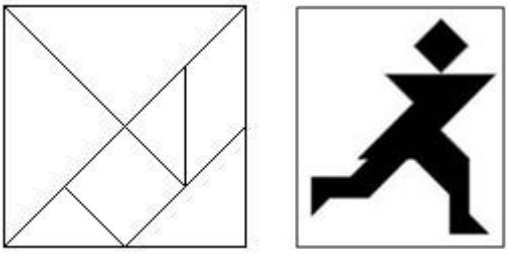

Fig. 1. Tangram stones and an example of Tangram figure

To illustrate how natural utterances of the subjects in solving of Tangram puzzles look like, let us observe two dialogue fragments translated into English from the corpus shown in Fig. 2. The example includes two sequences of commands produced by the subjects, whereas the corresponding system's actions are not explicitly given.

In the first sequence, from the subject's point of view, the square should be put on the position of the head in the figure (see Fig. 1), but unfortunately the machine, i.e. WOZ pretended that it did not understand this natural speech (e.g. "*the square is the head*"), so he/she has to give up this instruction and express it in another way ("*the square to the top*"), then it is accepted. In the second sequence, the user even tries to precisely define a certain amount like "*2cm is as long as a leg of the triangle*".

1. The square is the head ... the square is the head of the figure ... the big triangle to the right ... stop ... upwards ... stop ... the square to the top ... stop ...

2. The second big triangle rotates by 45 degrees to the left ... move to the right ... stop ... 2 cm to the top ... 2 cm is as long as a leg of the triangle ...

Fig. 2. Two sequences of subjects' commands

The development of the grammar was based on the Grammatical Framework (GF) [5], which is a grammar formalism for developing multilingual grammar applications.

Section 2 gives an introduction to the fragment analysis, including the utterance classification and the syntax analysis. Section 3 explains the formulation of the grammar, which includes the implementation of multilingualism and the composition of the grammar. Section 4 presents experimental results with corresponding discussion from two tasks: parsing and linearization. Section 5 concludes the paper.

2 Utterance Analysis

To get a better view of the utterances that appeared in the corpus, we decided to first classify the utterances in order to find out the common patterns and concepts used by

the subjects, and then through syntax analysis gather helpful information to find a good way to design the grammar accordingly.

From the 10 available transcripts of the NIMITEK corpus, 15 samples (one transcript may have one or two Tangram tasks) of Tangram were collected, which contain 2494 utterances. The Utterances were already manually annotated with three labels: command, comment, and question. In this work only 'command' utterances were considered, because more than 80% (2070) of the utterances in the collected samples are labeled with 'command'.

2.1 Utterance Classification

The process of classifying utterances helped us to detect common and non-common language patterns that were used by subjects. The utterances were therefore classified into three groups: simple utterances, complex utterances, and others.

1. Simple utterances

Utterances are simple if they have a simple syntactic structure, particularly if they are short and elliptic, and their referenced concepts of the task domain can be identified directly. For example, "move the triangle to the left" has a simple syntax: a verb phrase built up from a noun phrase and a prepositional phrase. The referenced concepts have no modifications (e.g. size and position for object, speed for movement), and thus can be identified directly: an action "move", an object "triangle", and a direction "to the left".

2. Complex utterances

Complex utterances also have a simple syntactic structure, but they demand more interpretation effort to identify the referenced concepts. This is usually due to the usage of modifications. For example, the utterance "the big triangle on the left" has also a simple syntax: a noun phrase built up from an adjective, a noun, and a prepositional phrase. However, there are two modifications of the noun "triangle": the adjective that specifies a size "big", and the prepositional phrase that specifies a position "on the left". We have to consider these two modifications to identify the target object: the triangle, which is big and is on the left.

3. Others

Utterances that cannot be classified to either of the two types above are collected in this group. They have one or more of the following properties:

- too complex syntactic structures. For example, "Jetzt nimm bitte das Dreieck, was von den beiden das größte ist und markier das erstmal (now please take the triangle, which is the largest of the both and label it first)".
- too complex content or rarely used concept. For example, "das Quadrat ist der Kopf (the square is the head)". The concept "Kopf (head)" appears only once in the corpus, apparently, it is uncommon in the task.
- too little or no information to be used to get any useful interpretation. For example, "Ich befinde mich mit meinen Anweisungen wieder auf der linken Abbildung (I want to locate my position in the left side of the figure again)". This utterance was

manually labeled as 'command' in the corpus. However, for the system it is not a real command. This is because it contains no explicit knowledge about the task, which is useful for the system to give its next action.

Through this classification we got a basic idea of the construction of the commonly used utterances. The semantic structure of most commands can be captured by a rule like: *object* + *direction* + *action*. Additionally, a word list for Tangram, which contains all words used for each semantic category and modifications was also collected from the corpus.

2.2 Syntax Analysis

Part-of-speech (POS) tagging was done to help us understanding the syntax of sentences. Context free grammar (CFG) analysis was done to prepare the extraction and construction of rules. Utterances selected from the NIMITEK corpus were tagged automatically by a regular expression tagger from the natural language toolkit (NLTK) [6] for German. For example:

- Das/DT oberste/JJ Dreieck/NN

  (the top triangle)

- Bewegen/VB nach/IN unten/RB

  (move down)

Here, *DT* means determiner, *JJ* means adjective or numeral, ordinal, *NN* means noun with singular form, *VB* means verb with base form, *IN* means preposition or conjunction, *RB* means adverb.

Through a bottom-up induction method, the corresponding context free grammar (CFG) [7] was built manually from POS tags to constituents. For example, for the two sentences above, the related CFG derivations are as follows:

- DT JJ NN → NP → S
- VB IN RB → VB PP → VP → S

From this experimental approach, we generated a basic command structure with three basic parts for designing a grammar for the typical Tangram commands from the NIMITEK corpus: *object, direction,* and *action*. In addition, the *object* could have modifications such as *size, position*, and *color*. The modification *position* has two possible types: object related position and global position. For example, "the triangle near the square" represents an object related position; "the left triangle" represents a global position. Object related position is used rather uncommon by subjects in contrast with global position, so we decided not to take the object related position into account for our grammar. Actions like movement could also have modification such as *speed*. For example, "go slowly", "move fast".

3 Grammar Formulation

Based on the collected syntactic structures, i.e. the rules for the naturally occurred utterances that were summarized from the corpus, we formulated the grammar.

3.1 Multilingualism

Besides the requirement of parsing commonly used user inputs, another major requirement of our grammar is multilingualism. In particular, we want to use one task-specific grammar for multiple languages. This is a specific problem of grammar engineering and can practically be solved by separating language dependent and language independent grammar components. The language independent part is called abstract syntax. It comprises concepts and rules of the task domain. A concrete syntax on the other hand, combines information about how to build up syntactical constituents from words for one specific language.

We chose GF [8] to implement our grammar, because it has many advantages for multilingual grammar applications. GF brings the Resource Grammar Library (RGL) [9] with it, which mainly provides language dependent and independent functions to form syntactical constituents. A wide range of languages is supported, including English and German. By using these functions, morphological aspects (e.g. inflection, gender and number agreement, long distance dependency...) are handled automatically, which eases the process of grammar engineering.

Additionally, GF uses a module system, which allows different languages to share declarations for common concepts (like noun-phrases) and building functions for them, while each uses its own definitions. This allowed us to use generic building functions to formulate the concrete syntax and outsource the dictionary of words (including knowledge about features like gender etc.). A concrete syntax is then called incomplete concrete syntax. It ensures two things, which also ease grammar engineering: first, changes of the concrete syntax will automatically affect all languages; second, adding new languages is achieved by simply adding an appropriate word lexicon.

To illustrate it more clearly we give examples for the particular syntaxes for Tangram:

1. Abstract syntax: The abstract syntax gives a set of declarations and functions for categories and production rules. Fig. 4 shows a fragment of the abstract Tangram syntax. **cat** means categories, **fun** means functions, i.e. grammar rules. The syntax defines four categories and one rule to form a command. The fragment in **fun** formalizes a rule that a command is made of an action (e.g. “flip”, “select”), an object (e.g. “the small triangle”), and a direction (e.g. “to the left”).

```
cat
  Command;
  Object;
  Action;
  Direction;

fun
  CmdPhrase: Object -->Action --> Direction --> Command
```

Fig. 4. Example: abstract syntax for Tangram

2. Concrete syntax: Concrete syntax maps abstract syntax trees into linguistic objects. The objects can be simply strings or records that contain more strings, agreement features, etc. [10]. For instance, a part of the concrete syntax for Tangram is shown in Fig.5. **lincat** defines the linearization types, and **lin** defines the combination

rules for linearization, which state e.g. that *Command* should be an imperative verb phrase or that the object is a noun phrase. These categories are combined by using type conform building functions (e.g. mkVP …) to form a command.

```
lincat
  Command = Imp ;
  Object = NP ;
  Action = V ;
  Direction = Adv ;

lin
  CmdPhrase o a d = mkImp (mkVP (mkVP a o)  d) ;
```

Fig. 5. Example: concrete syntax for Tangram

Furthermore, different languages have different parameter systems and lexicons. This ensures the independence of the grammar and the expansibility of the language.

3.2 Grammar Formalism

Following Ranta [11], the grammar in our work is composed by the following parts:

1. Abstract syntax: declare categories used by the user and functions in Tangram, i.e. semantic grammar rules.
2. Incomplete concrete syntax: define the declared categories and functions by using the language independent syntax building functions of the RGL, and uses a concrete syntax in 3.
3. Concrete syntax: declare the language dependent resources, which includes separate language syntax and word lexicon for German and English.
4. Lexicon interface: declare words, i.e. word symbols, and their types. The interface is language independent.
5. Lexicon instances: define words together with their morphological features (e.g. case, gender) for both German and English.

To get a better understanding of the parsing process, let us see a parsing example taken from the command line interface of GF shown in Fig. 6.

```
>parse "bewege das kleine Dreieck nach links" ["move the small triangle
to the left"]

Command PositivePol (Translate Simple (mkObjectNP PieceK
DefiniteSg (mkObjectCN PieceK (Size Small) Triangle)) (DirectionS
LeftAdv))
```

Fig. 6. Example of parsing

Fig. 7 shows a simplified syntax tree, which illustrates the parsing result above and gives a very simple and clear interpretation. Every node in the tree represents a syntactic constituent and is formed by the constituents of its child nodes. One such

building step is represented by a specific rule in the grammar. For example, **Move** represents a verb phrase, which is built of a noun phrase **Object** and a prepositional phrase **Direction**, with "move" as its central verb.

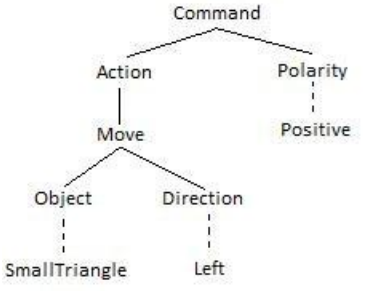

Fig. 7. Example of the syntax tree

4 Results and Discussion

We used 2070 utterances (cf. above) in the samples of Tangram in the NIMITEK corpus to test our grammar. The evaluation includes parsing, i.e. language analysis, and linearization, i.e. language generation.

4.1 Parsing

Since the grammar is currently not sensitive with respect to punctuation, we get the parsing results (shown in Table 1) by omitting all punctuations in the utterances.

Table 1. Results of utterance parsing

Command utterance	Number
Parsable	1248
Non-parsable	822

Parsable means everything that is acceptable by the grammar and thus by the parser. Utterances that are parsable include the two groups that we have mentioned earlier in utterance classification: simple utterances and complex utterances.

Non-parsable means everything that is not acceptable by the grammar. This includes utterances of the third group of the utterance classification that was mentioned in section 2.1. We decided not to build rules for them because they are hard to process (according to the three properties in this group) and bring little gain in our scenario. Otherwise, we found non-parsable utterances in the corpus, which could be turned into parsable utterances with little effort. For example, the parser fails to build a syntax tree, when there is an unknown word, non-agreeing word endings or wrong spellings. This could be repaired easily in most cases. However, we did not test this approach so far.

4.2 Linearization

Linearization means language generation, which generates language strings from syntax trees (according to the abstract syntax). It is the reverse process of parsing. A syntax tree can be rendered in any of the available languages (in our work: German and English) through concrete syntax. Fig. 8 shows a result of language generation for both English and German.

```
>linearize (Command PositivePol (Select (mkObjectNP PieceK
DefiniteSg Triangle)))

take the triangle
nimm das Dreieck
```

Fig. 8. Example of generation

Another important usage of linearization is translation. Different languages can be translated to each other. This is achieved by using a parse-linearize pipe: the input language will first be parsed with its grammar and lexicon, then linearized back to the target language with the target language grammar and its related lexicon. Fig. 9 shows a result of language translation from German to English.

```
>parse –lang=TangramGer "bewege das kleine Dreieck nach
links" | linearize –lang=TangramEng

move the small triangle to the left
```

Fig. 9. Example of translation

Here *TangramGer* is the German concrete syntax, and *TangramEng* is the English concrete syntax.

However, in our application, the advantage of the grammar is to support multilingual speech input and output, so we use it mainly for utterance parsing and generation.

5 Conclusion

In this work we developed a corpus-based multilingual grammar with the ability of both parsing and linearization. The grammar used self-defined semantic categories (e.g. object, action, direction) instead of syntactic categories (e.g. NP, VP). It was expected that the grammar can parse as many as possible naturally occurring utterances with simple syntax when solving Tangram puzzles interactively, and the goal was achieved by applying the following approach: First, classifying utterances to find out commonly used patterns and generating a word list of the task. Second, analyzing syntactic structures by applying POS tagging and CFG analysis. Third, formulating the grammar in GF, and synchronizing the syntax and lexicon in both German and English for multilingualism. The parser was integrated now in a Tangram demonstrator of the NIMITEK project with both speech and textual input interface.

The development was based on the transcripts of the NIMITEK corpus and the interface of the grammatical framework (GF). The transcripts of the NIMITEK corpus are in German, English transcripts are not available.

Additionally, during the analysis of the utterances, we found out that seven in ten subjects in the recordings of the NIMITEK corpus used simple, short instructions, while the other three used more complex, clause-based instructions. This may show a fact that most of the people intuitively choose to use simple or elliptical instructions, which on one side reduces the complexity of grammar development, but on the other side increases the degree of ambiguity, i.e. what users actually want or want to do. For instance, there are five triangles in Tangram, for user commands like "*the triangle*", which one did the user really mean? Therefore, a corresponding dialogue strategy was planned and used for a backup. The strategy applied different types of questions for the system to guess and confirm the intention of users. For instance, giving a command like "*the triangle to the right*", system will first guess the most possible triangle according to the state history of the task, then mark it and ask for user's confirmation with a question like "*do you mean this one?*" In this way the system's ability of solving ambiguity in natural language understanding was improved. Furthermore, if the user commands from the results of automatic speech recognition (ASR) are not understandable, according to the dialogue strategy the system outputs will be like "*pardon?*", "*sorry, I do not understand you.*", or "*I did not get it, could you please repeat it again?*".

Acknowledgment. The presented study is based on work within the NIMITEK project (http://wdok.cs.uni-magdeburg.de/nimitek), and within the SFB TRR 62 ('Companion Technology for Cognitive Technical Systems') funded by the German Research Foundation (DFG). The responsibility for the content of this paper lies with the authors.

References

1. Gnjatovic, M., Rösner, D.: Inducing Genuine Emotions in Simulated Speech-Based Human-Machine Interaction: The NIMITEK Corpus. IEEE Transactions on Affective Computing, 132–144 (2010)
2. Fraser, N., Gilbert, G.N.: Simulating speech systems. Computer Speech and Language 5, 81–99 (1991)
3. Gnjatovic, M., Rösner, D.: Gathering Corpora of Affected Speech in Human-Machine Interaction: Refinement of the Wizard-of-Oz Technique. In: Proceedings of the International Symposium on Linguistic Patterns in Spontaneous Speech (LPSS 2006) (2006)
4. Gnjatovic, M., Rösner, D.: The NIMITEK Corpus of Affected Behavior in Human-Machine Interaction. In: Processing of the Second International Workshop on Corpora for Research on Emotion and Affect (satellite of LREC 2008), pp. 5–8 (2008)
5. Ranta, A.: Grammatical Framework: A Multilingual Grammar Formalism. Language and Linguistics Compass 3 (2009)
6. `http://www.nltk.org`
7. Knuth, D.E.: Semantics of Context-Free Languages. Theory of Computing Systems 2(2), 127–145 (1968)

8. Ranta, A., Angelov, K., Bringert, B.: Grammar Development in GF. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session (EACL 2009), pp. 57–60 (2009)
9. Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology 2 (2009)
10. Khegai, J., Nordström, B., Ranta, A.: Multilingual syntax editing in GF. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 453–464. Springer, Heidelberg (2003)
11. http://www.grammaticalframework.org/doc/gf-tutorial.html