Human-centric Computing and Information Sciences

SpringerOpen

WordBricks: a virtual language lab inspired by Scratch environment and dependency grammars

Human-centric Computing and Information Sciences 2013, **3**:5 doi:10.1186/2192-1962-3-5

Maxim Mozgovoy (mozgovoy@u-aizu.ac.jp)
Roman Efimov (romicher@gmail.com)

ISSN 2192-1962

Article type Research

Submission date 22 December 2012

Acceptance date 11 March 2013

Publication date 8 April 2013

Article URL http://www.hcis-journal.com/content/3/1/5

WordBricks: a virtual language lab inspired by Scratch environment and dependency grammars

Maxim Mozgovoy[1*]
[*] Corresponding author
Email: mozgovoy@u-aizu.ac.jp

Roman Efimov[1]
Email: romicher@gmail.com

[1] University of Aizu, Ikki-machi, Aizu-Wakamatsu, Fukushima, Tsuruga, Japan

Abstract

This paper explains design decisions forming a foundation of WordBricks — an intelligent computer-assisted language learning environment, recently initiated at our institution. WordBricks is intended to serve as a "virtual language lab" that supports open experiments with natural language constructions. Being based on dependency grammars, this instrument illustrates the use of modern natural language processing technologies in language learning. The latest prototypes of WordBricks also show how dependency-styled constructions can be represented in a more natural sequential form that facilitates easier user interaction.

Keywords

Computer-assisted language learning, Natural language processing, Dependency grammar

Introduction[a]

The use of computer-assisted language learning (CALL) instruments is now widespread and well recognized both by language teachers and language learners. Past decades brought more powerful and accessible computers and numerous CALL software packages; the level of technological awareness among teachers has also increased greatly. At this point, it seems natural that researchers are often more focused on the integration of existing technologies into language curricula and the development of well-balanced teaching methods that combine theory, technology, and pedagogy, rather than on purely technological advancements for CALL systems [1].

However, popular CALL systems still rarely incorporate modern achievements of natural language processing technologies. For example, language learning software packages, recently reviewed in PC Magazine [2], at best provide the following capabilities: lessons with multimedia content, word-based memory games, online tutoring, and pronunciation training. Some systems were characterized as being brilliantly designed, nicely organized (as a combination of traditional lessons, word drills, scenario-based lessons/dialogues, etc.), or based on innovative educational concepts, such as involving a learner into a real text translation project. Undoubtedly, these features are beneficial for a language learner, but in most cases they do not make use of recent research advancements (probably, the only exception is high-quality speech recognition).

The lack of intelligence in CALL systems is a well-known problem, clearly formulated at least as early as in 1992 [3]. It has been suggested that a hypothetical intelligent CALL (ICALL) system can be based on both technical (natural language processing, speech recognition, feedback generation) and theoretical (pedagogy, cognitive science) advancements. The review of ICALL instruments conducted in 2002 identified at least 40 systems that use artificial intelligence (AI) technologies to a certain extent [4]. The same paper admits that many capabilities of ICALL systems cannot be reliably addressed with state-of-the-art technologies. This is a likely reason for a low interest in AI technologies for CALL today. As noted in [5], "the development of systems using NLP technology is not on the agenda of most CALL experts, and interdisciplinary research projects integrating computational linguists and foreign language teachers remain very rare".

Examples of ICALL systems provided in [4] and [5] show that AI technologies are most commonly used for grammar checking, textual feedback generation, and automatic speech recognition. Still, these technologies rarely address one of the major flaws of today's CALL systems, lying in their strictly limited interactivity. Typically a student accesses learning materials in the same way as in case of traditional books and audiotapes, while having little or no ways to *experiment* with language. One can note a contrast between CALL instruments and educational software, available for natural sciences, such as physics or chemistry. For these subjects, in addition to browsing multimedia learning materials, a student can often perform numerous experiments in a "virtual lab" (such as, for example, The Virtual Physical Laboratory [6] and The ChemCollective [7]).

Theoretically, numerous language learning activities might benefit from students' unrestricted experimentation (checking the applicability of a certain construction in a certain context, finding the best translation for the given phrase, exploring word morphology and the rules of verb government). In practice, many of these options are still too challenging for today's speech and language processing technologies. Given these limitations, one might consider an alternative approach: instead of fulfilling pedagogical aims with immature technology, it makes sense to try to implement scenarios that are technologically doable, and still have pedagogical value.

The idea of a "virtual language lab" based on established natural language processing technologies is the starting point of a project recently initiated at our institution. In this paper, we will introduce this project, and discuss its expected advantages and drawbacks as well as possible research directions. The first version of our software will be English language-based, but in this paper we will also use examples from other languages to illustrate certain grammatical phenomena.

The basic concept of "Word Bricks"

We decided to devote our project to one specific type of language learning activities: to the process of constructing grammatically correct phrases. A student with initial vocabulary and some knowledge of grammar rules might want to practice them by creating simple sentences. At this stage, it is important to make sure that the sentences are built properly, and if not, the student gets necessary feedback. By creating sentences, the student in the simplest case can test hypotheses about the correctness of certain constructions. In more advanced scenario, the feedback might include hints on the proper use of words and word combinations. For example:

- A student can check whether a certain word is appropriate in a certain context. Suppose the student knows that one can *ride a horse*, but can one *ride a car*?
- A student can find the correct word form for the given syntactical context. In English, the verb form depends on the subject's person, so the student has to choose between the base form of the verb and the 3rd person singular form. For other languages these rules can be more complicated. For example, Russian verbs are conjugated according to the subject's person and number in the present tense, but to the subject's gender and number in the past tense.
- A student can find correct prepositions and/or grammatical cases for the given context. For example, in Finnish some verbs require that the object noun is always set into a certain form (so the verb "governs" the noun). This verb / noun form list has to be memorized.

The idea of incorporating a grammar checker into CALL software is not new. Such an automated feedback generation system was implemented, e.g., in Robo-Sensei Japanese tutoring system [8]. However, today's grammar checkers are not very helpful in open experiments with language constructions. As noted in [5], grammar checkers are usually aimed at native speakers, and do not provide sufficient feedback for language learners. One possible way to solve this problem is to restrict user input. This approach is implemented in Robo-Sensei: the system asks the student to answer a specific question, and then compares the response with an "answer schema" that specifies the pattern of the expected correct response.

We believe that free experiments with language constructions are possible without traditional grammar checking technologies. Consider the following analogy. A programmer, working with traditional programming languages, has to write plaintext code that is translated into low-level machine instructions. It is a job of a compiler or interpreter to parse the code, and to identify possible syntactic errors. Unlike them, *visual programming systems*, often used for teaching programming to kids, store programs in graphical flowcharts (see, e.g., Flowol [9]), thus eliminating the need of parsing and error checking. One can draw a flowchart that corresponds to a wrong algorithm, but the flowchart itself cannot be "syntactically incorrect", since the visual editor allows no illegal links between the elements.

In a sense, flowcharts represent "parsed" programs, stored in the form that directly reflects their syntactic and semantic structure. Natural language sentences also can be represented in a parsed tree-like form with *phrase-structure* grammars or *dependency grammars* [10]. Our idea is to let the students compose *parsed* sentences directly instead of traditional writing.

General design of the system

Currently, we are developing the system with the following image in mind. A student is given a number of "word bricks" that represent single words. The student can connect individual bricks to form phrases and sentences. Every brick has typed incoming and outgoing "connectors", ensuring that only grammatically correct links are possible.

We believe that dependency links are easy to understand, since they connect words of a sentence directly, and do not require additional non-word bricks, as in case of phrase-structure links. The dependency link from the word A to the word B can be informally explained as a question that contains A, and has B as an answer. For example, in the phrase *he likes apples* there is a dependency link from *likes* to *he*, since it is possible to construct a

question *who likes apples?*, having *he* as an answer. This idea is illustrated in the Figure 1 that shows the parse tree of the phrase *Tomorrow we go to Tokyo*.

Figure 1 The parsed phrase *Tomorrow we go to Tokyo*.

At this point it is important to note that the Figure 1 shows just a possible visualization of a parse tree, displaying a number of word bricks, connected with directed arrows. However, this is not the only way to draw a parse tree, and we will return to this question later.

From basic bricks to typed bricks

Let us recall that one of our aims is to restrict possible connections, so the student cannot produce ungrammatical sentences. The formalism of dependency grammars allows specifying the type of word-word relationship, such as verb-subject, verb-object, noun-modifier, and so on. If we know the type of this relationship, we can decide which restrictions should be applied in the given case.

Perhaps, the development of such word linking constraints for each relationship type is the most challenging part of our project. These rules vary greatly from language to language, and might require morphological or even semantic information about the words to be linked. We will not discuss here all possible types of grammatical relationships and all kinds of challenges that arise in the task of linking constraints declaration, let us consider several examples for the sake of illustration.

Noun-adjective link

In English, we can establish a link between any noun and any adjective (answering the *which?*-question). In Russian and Spanish this noun-adjective link can be established only if the adjective agrees in number and gender with the noun:

libro rojo *(red book)*
libros rojos *(red books)*
rosa roja *(red rose)*
rosas rojas *(red roses)*

Verb-object link

In English, normally any noun or pronoun in objective case can be used as an object of a verb:

I like cars.
I like her. *(‘her’ is an objective case of ‘she’)*

In Russian, we need to know whether the object represents something alive. For animate things the word form of the object is identical to the genitive case form, while for inanimate things the nominative case form should be used.

Verb government

The examples above describe general grammatical rules that hold for wide classes of word pairs. However, there are also verb-object relationships that depend on particular verbs. For example, the verb *to buy* can be used with an indirect object *place-of-purchase* with the preposition *in*:

I buy fish in a shop.

This fact is not as trivial as it might seem: in Finnish language one buys something *from* a shop (and this is expressed without any prepositions; the corresponding form of the word *shop* is used instead). So the choice of prepositions and word forms of verb objects is not obvious. It might depend on a particular verb, such as *to buy*.

Semantics-driven links

The discussed above link types can be used to ensure grammatical correctness of phrases. However, they do not prevent improper word use. Consider the following example. In English, one can *break the cup* and *break the law*. The student, familiar with English, might try to reproduce the same pattern in Russian, but this is incorrect: in Russian it is impossible to use the same verb in these two distinct contexts.

We believe this problem can be addressed with additional constraints on word types, as suggested in [11], though we did not decide yet whether we are going to implement this functionality, as it requires considerable amount of work. The idea is to introduce a hierarchy of word categories. By employing it, we can specify that one can break only breakable things, drive only drivable things, and so on. Several such ontologies are already available and can be used (see, for instance, the system of WordNet categories [12]).

Visualizing word bricks

In the first prototype of our system we have implemented a straightforward user interface: the students can arrange word bricks in the main window and connect them with lines to obtain graphs, similar to the one shown in the Figure 1. While this kind of visualization is the most natural way to show dependency trees, the process of drawing such diagrams hardly can be considered as the most intuitive way to build sentences. By drawing dependency trees, the students understand the tree-like structure of sentences, but such knowledge is beyond the curriculum of most language courses.

This problem has forced us to start designing an alternative visualization system for dependency trees, which would provide a more natural way for students to build sentences. The main source of inspiration for this work is the Scratch programming language learning environment [13]. In Scratch, the programs are created by connecting blocks, containing conventional elements of a programming language, such as assignments, input / output statements, branching and looping constructions. The blocks are shaped like jigsaw puzzle pieces, so it is impossible to create a syntactically incorrect program. For example, an IF-THEN-ELSE block contains three slots that have to be filled: the Boolean conditional expression, the THEN branch, and the ELSE branch (see the Figure 2). Unfortunately, natural language constructions are much more complex than the elements of a programming

language, and our first attempts to formalize them in form of jigsaw puzzle parts are far from being complete.

Figure 2 The IF-THEN-ELSE block in Scratch.

Such a puzzle-styled representation has one more important advantage: it provides a natural way to formalize word order rules (Figure 3). In the standard visualization of dependency trees (Figure 1) there is no means to specify the correct word order. However, it is unclear how puzzle parts can be effectively used in languages with relaxed word order.

Figure 3 The phrase *Tomorrow we go to Tokyo* represented with puzzle bricks.

Pros and cons

In the previous sections we have outlined specific techniques for addressing particular language phenomena. Now let us discuss the potential advantages and drawbacks of our "virtual language lab", affecting its pedagogical value.

At present, we see the following positive sides of our approach:

Supporting "virtual labs" in language learning. As mentioned above, the idea of open experimentation is supported in numerous educational software projects. However, in computer-assisted language learning this "virtual lab" approach is clearly underrepresented.

Formalized explanations. Typed word bricks provide a natural way to explain such language phenomena, as morphology, homonymy, cases and prepositions, verb government, and proper word use. Students can see how the choice of a word form affects brick type; how subjects and objects are linked to verbs, and so on.

Understanding underlying structures. Parse trees show the structure of sentences, thus contributing to deeper understanding of grammar rules and word-linking principles.

Contextualized assistance. Since the system knows internal structure of phrases, being constructed by students, it can provide numerous context-dependent hints. For example, it can automatically select the proper verb form for the given subject-verb word pair; it can provide a list of prepositions and grammatical cases, used with the given verb; it can display a list of breakable things used with the verb *to break* or a list of drivable things used with the verb *to drive*, and so on.

Our approach has also disadvantages, whose impact can be evaluated only in real-life experiments:

Unnatural constructions. Lucien Tesnière, who pioneered dependency grammars, distinguished the concepts of *words* as syntactic elements and of *nuclei* as complex elements carrying the same role as words [14]. For example, both the word *sees* and the word combination *will have been* seeing correspond to single nuclei. In a sense, "will have been seeing" is logically a single word, syntactically made up of separate tokens.

Dependency trees provide a convenient and natural way to link nuclei, but the situation becomes less obvious for the words inside a single nucleus. What kind of links connect the words *will*, *have*, *been*, and *seeing*? There are many such confusing sentence elements: complex objects that consist of and-, or-, or comma-separated elements; quotations;

prepositions and articles; proper names; punctuation marks. Researchers have developed consistent guidelines that assist constructing dependency trees (see, e.g., Stanford typed dependencies manual [15]), but the need of knowing these technicalities is an unnecessary burden for a language learner. One may argue that even the parsed representation itself is a burden, so these complications with word linking rules make the system impractical.

Dependency grammars were also criticized for little support of word ordering rules. There are attempts to address this defect (see, e.g., [16]), but currently it is unclear how to incorporate word ordering into our system in a natural, pedagogically sound way. The situation can be partially remedied by the system of jigsaw puzzle blocks that can incorporate complete nuclei and word order rules, but it is much harder to design a complete set of such blocks than to employ traditional parse trees.

Limitations of error prevention system. The proposed system is not bullet-proof. By design, it analyzes local contexts of words only, so it cannot detect errors that appear at paragraph level. For example, the system is generally unable to detect improper article use (except simple cases with precise phrase-level rules, such as "do not use articles with people's names"). The system is also unable to detect semantic errors, when the sentence is grammatically correct, but the meaning is wrong.

Technical difficulties. The complete set of word-linking rules, described in Section IV, is most probably too large for manual implementation. For the proof-of-concept system, we plan to limit ourselves with a small vocabulary, and to declare rules manually. However, for a full-sized system we will need to learn rules automatically from treebank data.[b] Currently, it is hard to estimate how challenging this process is.

Conclusion

In this paper we have outlined basic design ideas of WordBricks — a new virtual language lab project, recently initiated at the University of Aizu. We are trying to implement a tool for open experimentation with language constructions. Such ICALL instruments are still very rare today.

Throughout the paper, we have seen how various linguistic phenomena, such as word agreement, verb government, cases and prepositions can be handled, and how dependency parse trees can be used by students to construct phrases and sentences. We believe that such a visual representation of sentence structure is helpful for deeper understanding of human language grammar rules.

This year we are planning to conduct first experiments in a real classroom environment and to make grounded conclusions about the feasibility of our approach. We are aware of potential limitations and drawbacks, but many of them are caused with objective complications of human language, and there is no way to overcome them completely.

Endnotes

[a] This paper is based on: M. Mozgovoy. Towards WordBricks — a Virtual Language Lab for Computer-Assisted Language Learning. *The 2nd Int'l Workshop on Advances in Semantic Information Retrieval*, Wroclaw, Poland, 2012, p. 251-254

[b] A *treebank* is the collection of manually parsed sentences.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

MM created initial design of the WordBricks system, which allows the users to create sentences in form of dependency parse trees. RE is the main developer of the same system in the form of Scratch blocks, which should be a simpler choice for the users not familiar with dependency parsing. All authors read and approved the final manuscript.

References

1. Garrett N (2009) Computer-assisted language learning trends and issues revisited: integrating innovation. The Modern Language Journal 93:719–740

2. Duffy J The best language-learning software. PC Magazine, April 12, 2012. URL: http://www.pcmag.com/article2/0,2817,2381904,00.asp

3. Swartz M, Yazdani M (1992) Intelligent tutoring systems for foreign language learning. Springer Verlag

4. Gamper J, Knapp J, Gamper J, Knapp J (2002) A review of intelligent CALL systems. Computer Assisted Language Learning 15(4):329–342

5. Amaral L, Meurers D (2011) On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. ReCALL 23(1):4–24

6. The Virtual Physical Laboratory, URL: http://www.colpus.me.uk/vplabd/

7. The Chemistry Collective, URL: http://chemcollective.org/

8. Nagata N (2009) Robo-Sensei's NLP-based error detection and feedback generation. Calico Journal 26(3):562–579

9. Flowol, URL: http://www.flowol.com

10. Rambow O, Joshi A (1997) A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. Recent Trends in Meaning-Text Theory 39:167–190

11. Mozgovoy M, Kakkonen T (2009) An Approach to Building a Multilingual Translation Dictionary that Contains Case, Prepositional and Ontological Information. In: (ed) Proc. of the 12th Int'l Conf. on Humans and Computers., pp 135–139

12. Vossen P et al (1998) The EuroWordNet Base Concepts and Top Ontology. Technical report 1998-TR-004, Centre National de la Recherche Scientifique, France

13. Silver J, Silverman B, Kafai Y (2009) Scratch: programming for all. Communications of the ACM 52(11):60–67

14. Kahane S (1996) If HPSG were a dependency grammar…. In: (ed) Proc. of the 3rd TALN Conf., pp 45–49

15. de Marneffe M-C, Manning C (2008) Stanford Typed Dependencies Manual. Stanford University

16. Duchier D, Debusmann R (2001) Topological Dependency Trees: A Constraint-Based Account of Linear Precedence. In: (ed) *Proc. of the 39th Annual Meeting on Association for Computational Linguistics*. pp 180–187

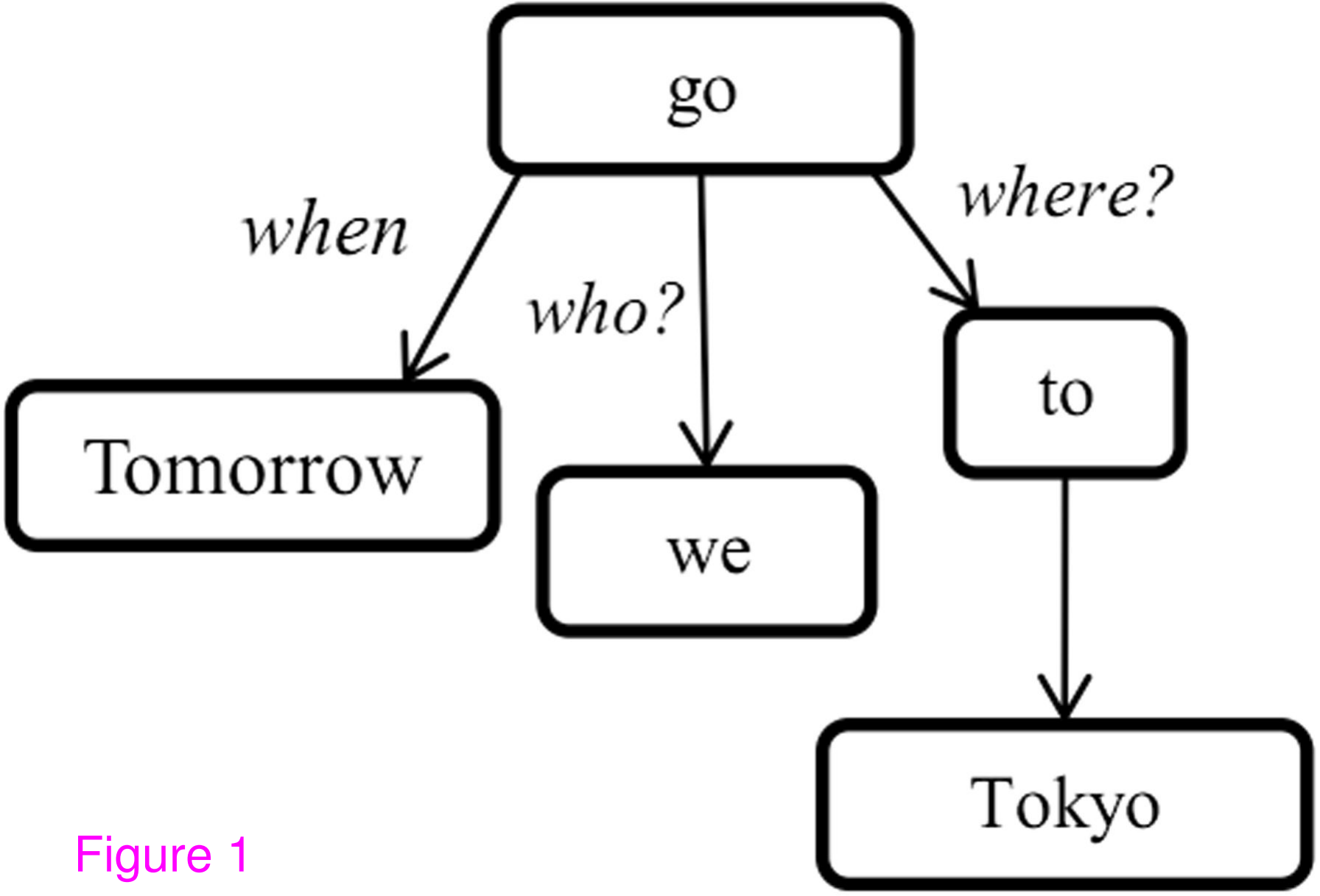

Figure 1

Figure 2

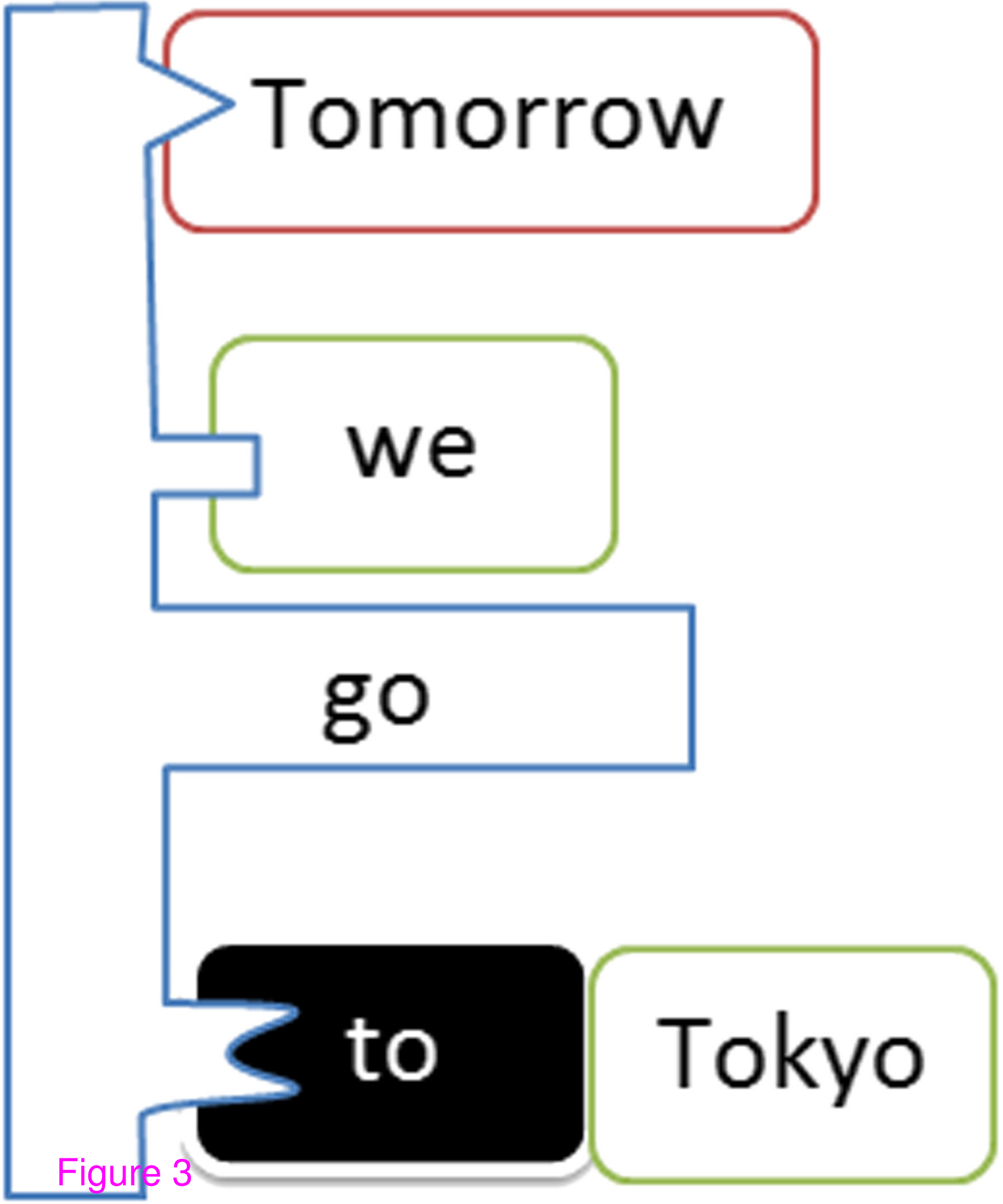

Figure 3